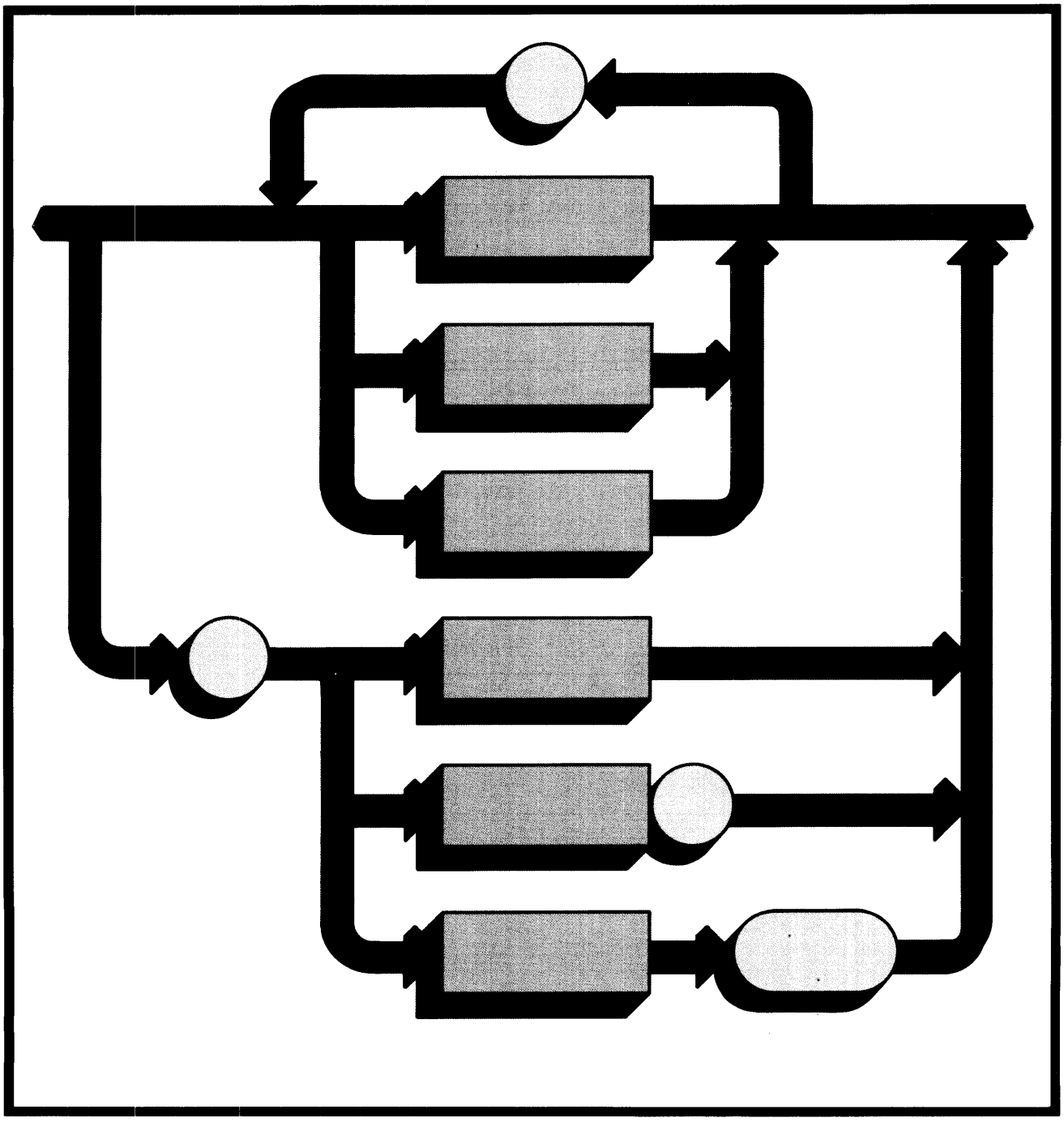
HP 9000 Series 200 Computers

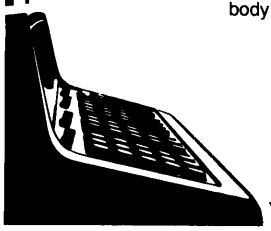
HEWLETT PACKARD

Pascal Source Code Listings
Volume II-Pascal



Pascal 3.0 Source Code Listings
Volume II-Pascal
for the HP 9000 Series 200 Computers

Manual Part No. 98615-90074



Hewlett-Packard Company
3404 East Harmony Road, Fort Collins, Colorado 80525

Printing History

New editions of this manual will incorporate all material updated since the previous edition. Update packages may be issued between editions and contain replacement and additional pages to be merged into the manual by the user. Each updated page will be indicated by a revision date at the bottom of the page. A vertical bar in the margin indicates the changes on each page. Note that pages which are rearranged due to changes on a previous page are not considered revised.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

February 1985...Edition 1

Table of Contents

File →Module Map

The title pages in the volumes of listings refer to *file* names, not *volume* names. The compilation of a single file may include multiple modules, and the module names many times bear no resemblance to the file names in which they appear. To help you determine which modules appear in each file of this listings volume, the following file:module cross-reference has been created. In the left column are the file names, and in the right column, in alphabetical order, are the modules which are declared in each file. The map only includes files whose listings are in this volume of listings.

A804XDVR	a804xdvr
AMIGO	amigodvr, csamigo
ASCII	asciimodule
BAT	bat
BUBBLES	bubble
CLOCK	clock
CONVERT	convert_text
CRT	crt
CS80	cs80, cs80dsr, cs80dvr, tapebuf
CTABLE	brstuff, ctr, options, scanstuff
C_HOOK	(no explicit modules)
DC_DRV	extdc, init_dc, intdc
DGL_C_IN	dgl_confg_in
DGL_C_OUT	dgl_confg_out
DGL_HPGL	dgl_hpgl
DGL_HPGLI	dgl_hpgli
DGL_INQ	dgl_inq
DGL_KNOB	dgl_knob
DGL_POLY	dgl_poly
DGL_RAS	dgl_raster
DGL_TOOLS	dgl_tools
DGL_VARS	dgl_vars
DISCHPIB	bkgnd, dischpib
DI_DRV	extdi, init_discint
DMA_DRV	init_dma
EDRIVER	edriver
EPROMS	eproms
ETU	(no explicit modules)
F9885	f9885dvr
GCRT	crtb
GEN	dgl_gen
GLE_FILE	gle_file_io
GLE_GEN	gle_gen
GLE_GENI	gle_geni
GLE_HPGL	gle_hpgl_out
GLE_HPGLI	gle_hpgl_in
GLE_HPIB	gle_hpib_io
GLE_KNOB	gle_knob_in
GLE_RGL	gle_ras_out

GLE_SCLIP	gle_sclip
GLE_SMARK	gle_smark
GLE_STEXT	gle_stext
GLE_TYPES	gle_types
GLE_UTLS	gle_utls
G_DRV	extg, init_gpio
HEAPT	hpm, sysglobals
HPHIL	hphil
H_DRV	exth, init_hpib
INIT	fs, initunits, isr, ldr, misc
INITLOAD	bootdammodule, loader, mini, sysglobals
IOLIB	general_1, general_2, general_3, general_4, hpib_0, hpib_1, hpib_2, hpib_3, serial_0, serial_3
KERNEL	general_0, iocomasm, iodeclarations
KEYS	keys
LIB	dgl_lib
LIBRARIAN	(no explicit modules)
LIFDAM	lifmodule
LOCKMOD	lockmodule
M68KSYS	ci
MINIT	asmr, bminit, cs80ir, hminit, iramigo, midecs, mminit, qminit, xminit,
MIUI	(no explicit modules)
MOREFSYS	mfs
MOUSE	mouse
NONUSKBD1	non_us_kbd1
NONUSKBD2	(no explicit modules)
PRINTER	prtdvr
RS_DRV	init_rs, rs
SEGMENTER	asm, segmenter
SRMAM	srmammodule
SRMDAM	srmdammodule
SRM_DRV	srm
SWVOL	(no explicit modules)
SYSDEVS	sysdevs
TAIL	(no explicit modules)
TAPEBKUP	cs80tbdvr, cs80tbr

Module →File Map

The following cross-reference map is the inverse of the one in the preceding section. This map will tell you which listing file a particular module is in.

A804XDVR	
amigodvr	AMIGO
asciimodule	ASCII
asm	SEGMENTER
asmr	MINIT
bat	BAT
bkgnd	DISCHPIB
bminit	MINIT
bootdammodule	INITLOAD
brstuff	CTABLE
bubble	BUBBLES
ci	M68KSYS
clock	CLOCK
convert_text	CONVERT
crt	CRT
crtb	GCRT
cs80	CS80
cs80dsr	CS80
cs80dvr	CS80
cs80ir	MINIT
cs80tbdvr	TAPEBKUP
cs80tbr	TAPEBKUP
csamigo	AMIGO
ctr	CTABLE
dgl_confg_in	DGL_C_IN
dgl_confg_out	DGL_C_OUT
dgl_gen	GEN
dgl_hpgl	DGL_HPGL
dgl_hpgli	DGL_HPGLI
dgl_inq	DGL_INQ
dgl_knob	DGL_KNOB
dgl_lib	LIB
dgl_poly	DGL_POLY
dgl_raster	DGL_RAS
dgl_tools	DGL_TOOLS
dgl_vars	DGL_VARS
dischpib	DISCHPIB
edriver	EDRIVER
eproms	EPROMS
extdc	DC_DRV
extdi	DI_DRV
extg	G_DRV
exth	H_DRV
f9885dvr	F9885

fs	INIT
general_0	KERNEL
general_1	IOLIB
general_2	IOLIB
general_3	IOLIB
general_4	IOLIB
gle_file_io	GLE_FILE
gle_gen	GLE_GEN
gle_geni	GLE_GENI
gle_hpgl_in	GLE_HPGLI
gle_hpgl_out	GLE_HPGL
gle_hpib_io	GLE_HPIB
gle_knob_in	GLE_KNOB
gle_ras_out	GLE_RGL
gle_sclip	GLE_SCLIP
gle_smark	GLE_SMARK
gle_stext	GLE_STEXT
gle_types	GLE_TYPES
gle_utls	GLE_UTLS
hminit	MINIT
hphil	HPHIL
hpib_0	IOLIB
hpib_1	IOLIB
hpib_2	IOLIB
hpib_3	IOLIB
hpm	HEAPTHEAPT
init_dc	DC_DRV
init_discint	DI_DRV
init_dma	DMA_DRV
init_gpio	G_DRV
init_hpib	H_DRV
init_rs	RS_DRV
initunits	INIT
intdc	DC_DRV
iocomasm	KERNEL
iodeclarations	KERNEL
iramigo	MINIT
isr	INIT
keys	KEYS
ldr	INIT
lifmodule	LIFDAM
loader	INITLOAD
lockmodule	LOCKMOD
mfs	MOREFSYS
midecs	MINIT
mini	INITLOAD
misc	INIT
mminit	MINIT
mouse	MOUSE
non_us_kbd1	NONUSKBD1

options	CTABLE
prtdvr	PRINTER
qminit	MINIT
rs	RS_DRV
scanstuff	CTABLE
segmenter	SEGMENTER
serial_0	IOLIB
serial_3	IOLIB
srm	SRM_DRV
srmammodule	SRMAM
srmdammodule	SRMDAM
sysglobals	HEAPT
sysglobals	INITLOAD
sysdevs	SYSDEVS
tapebuf	CS80
xminit	MINIT

```
a
  AMIGO(csamigo)                ***    108d
  CTABLE(ctr)                          395c   398c   399c   400c   403c
  DGL_POLY(dgl_poly)                 20252d 20259c
  GEN(dgl_gen)                         3195d  3198c  3200c  3205d  3208c  3210c
  INIT(fs)                              809d   810d   829d   831d
  INIT(ldr)                            2375c  2377c  2395c  2430c  2431c  2440c  2460c  2468c  2469c  2471c  2478c
  INITLOAD(loader)                      888d  1114c  1115c  1142c  1143c  1148c  1151c  1153c  1158c  1235c  1259c  1262c  1269c  1272c
                                       1274c  1286c  1332c  1392c  1428c  1432c  1433c  1441c  1442c  1452c  1453c  1456c  1457c  1458c
                                       1464c  1467c  1471c  1484c  1486c  1515c  1516c  1523c  1524c  1526c  1530c  1534c  1535c  1536c
                                       1542c  1546c  1549c  1570c  1571c  1574c  1598c  1599c  1602c  1604c  1625c  1631c  1632c  1634c
                                       1641c
  LIBRARIAN                             127c   128c   326c   355c   397c   398c   407c   415d   521c   572c   582c   607c   631c   857c
                                        954c   957c   962c  1090c  1128c  1130c  1168c  1178c  1277c  1279c  1280c  1291c  1295c  1298c
                                       1302c  1327c  1328c  1332c  1356c  1382c  1399c  1402c  1403c  1499c  1501c  1506c  1513c  1515c
                                       1517c  1528c  1529c  1533c  1535c  1543c  1544c  1549c  1559c  1560c  1577c  1581c  1592c  1597c
                                       1599c  1609c  1624c  1628c  1638c  1639c  1643c  1644c  1645c  1688c  1695c  1702c  1736c  1746c
                                       1767c  1783c  1805d  1812c  1815c  1816c  1817c  1818c  1821c  1822c  1828c  1834c  1836c  1838c
                                       1839c  1841c  1875c  1876c  1894c  1896c  1920c  1922c  1938c  1939c  1941c  1945c  1948c  1965c
                                       2022c  2032c  2035c  2043c  2052c  2053c  2074c  2076c  2101c  2102c  2114c  2181c  2186c  2190c
                                       2204c  2206c  2209c  2211c  2216c  2234c  2235c  2237c  2238c  2244c  2246c  2250c  2251c  2261c
                                       2269c  2273c  2274c  2275c  2283c  2288c  2289c  2290c  2292c  2297c  2298c  2299c  2318c  2319c
                                       2320c  2321c  2326c  2327c  2329c  2339c  2341c  2371c  2382c  2385c  2393c  2403c  2404c  2405c
                                       2410c  2411c  2412c  2455c  2479c  2491c  2493c  2502c  2513c  2522c  2524c  2527c  2528c  2531c
                                       2547c  2548c  2556c  2557c  2565c  2567c  2568c  2569c  2576c  2578c  2580c  2581c  2585c  2595c
                                       2596c  2598c  2599c  2601c  2612c  2619c  2621c  2627c  2639c  2647c  2652c  2654c  2707c  2709c
                                       2714c  2715c  2752c  2753c  2759c  2762c  2770c  2772c  2774c  2778c  2781c  2794c  2795c  2800c
                                       2801c  2806c  2810c  2813c  2816c  2838c  2841c  2842c  2843c  2849c  2850c  2961c  2964c  3093c
                                       3094c  3096c  3152c  3221c  3240c  3722c
  SEGMENTER(asm)                        123c   126c   127c   128c   135c   189c   235c   236c   241c   242c   329c
  SYSDEVS(sysdevs)                      374d   492d   496c   500c   505c
a5
  INIT(ldr)                            2469c  2470c
  INITLOAD(loader)                     1048d  1632c  1633c  1657c
  M68KSYS(ci)                   ***     190c
  SEGMENTER(asm)                ***     205c
a804xdvr
  A804XDVR                      ***     414d
  A804XDVR(a804xdvr)            ***      29d
  HPHIL(hphil)                  ***      30d
a804xinit
  A804XDVR                      ***      28d
a804xisr
  A804XDVR(a804xdvr)                     89d    122c   386c
a_max
  DGL_POLY(dgl_poly)                 20521d 20535c 20541c
a_min
  DGL_POLY(dgl_poly)                 20520d 20534c 20538c
aa
  KEYS(keys)                           144d   146d   147d   148d   149d   150d   151d   152d   153d   154d   155d
aaa
  CS80(cs80)                    ***     217d
abort_bkgnd_process
  AMIGO(amigodvr)                      645c   758c   770c   780c   969c
  CS80(cs80dvr)                       1009c  1133c
  DISCHPIB(bkgnd)                       64d   146d
  MINIT(hminit)                 ***     499c
  MINIT(qminit)                 ***    1275c
  TAPEBKUP(cs80tbdvr)           ***     356c
abort_bkgnd_process_proc
  CTABLE(scanstuff)                    1072d  1129c  1157c  1167c
abort_hpib
  IOLIB(hpib_2)                        1778d  1810d
```

```
abort_serial
  IOLIB(serial_3)                      2517d  2835d
abort_transfer
  IOLIB(general_4)                     1196d  1327d
abs_position
  CS80(cs80dvr)                        1215d  1244c  1245c  1270c  1271c  1297c  1298c  1309c  1310c
abscode
  LIBRARIAN                             298d  1145c  1209c  1213c  1460c  1462c
abscoords
  HPHIL(hphil)                  ***     110c
  SYSDEVS(sysdevs)              ***     315d
absolute
  INITLOAD(loader)                      858d  1460c
  LIBRARIAN                             179c  1551c  1610c  2377c  2398c  2696c
ac
  SRMDAM(srmdammodule)                  213d   317c   318c   320c   578d   614c   615c   616c   618c  1034d  1071c  1072c  1073c
ac_char_arr
  SRMDAM(srmdammodule)                  567d   569d
ac_chars
  SRMDAM(srmdammodule)                  569d   616c
ac_createlink
  SRMDAM(srmdammodule)                  203d   213d   317c   567d   613c   614c
  SRM_DRV(srm)                  ***     186d
ac_execute
  SRM_DRV(srm)                  ***     187d
ac_generic09
  SRM_DRV(srm)                  ***     188d
ac_generic1
  SRM_DRV(srm)                  ***     188d
ac_generic10
  SRM_DRV(srm)                  ***     189d
ac_generic11
  SRM_DRV(srm)                  ***     190d
ac_generic12
  SRM_DRV(srm)                  ***     191d
ac_generic13
  SRM_DRV(srm)                  ***     192d
ac_generic14
  SRM_DRV(srm)                  ***     193d
ac_generic15
  SRM_DRV(srm)                  ***     194d
ac_generic16
  SRM_DRV(srm)                  ***     195d
ac_generic17
  SRM_DRV(srm)                  ***     188d
ac_generic18
  SRM_DRV(srm)                  ***     189d
ac_generic19
  SRM_DRV(srm)                  ***     190d
ac_generic2
  SRM_DRV(srm)                  ***     189d
ac_generic20
  SRM_DRV(srm)                  ***     191d
ac_generic21
  SRM_DRV(srm)                  ***     192d
ac_generic22
  SRM_DRV(srm)                  ***     193d
ac_generic23
  SRM_DRV(srm)                  ***     194d
ac_generic24
  SRM_DRV(srm)                  ***     195d
ac_generic25
  SRM_DRV(srm)                  ***     196d
```

```
ac_generic3
  SRM_DRV(srm)                    ***    190d
ac_generic4
  SRM_DRV(srm)                    ***    191d
ac_generic5
  SRM_DRV(srm)                    ***    192d
ac_generic6
  SRM_DRV(srm)                    ***    193d
ac_generic7
  SRM_DRV(srm)                    ***    194d
ac_generic8
  SRM_DRV(srm)                    ***    195d
ac_manager
  SRMDAM(srmdammodule)                   203d   213d   317c   567d   614c  1034d  1071c  1105c
  SRM_DRV(srm)                    ***    181d
ac_purgelink
  SRMDAM(srmdammodule)                  1034d  1071c  1106c
  SRM_DRV(srm)                    ***    185d
ac_read
  SRM_DRV(srm)                    ***    182d
ac_search
  SRM_DRV(srm)                    ***    184d
ac_write
  SRM_DRV(srm)                    ***    183d
access
  SRMAM(srmammodule)                      56d    62c    67c    70c    75c    84c   103d   109c   114c   117c   122c   131c
  SRM_DRV                               1997d  2011c  2083d  2097c
  SRM_DRV(srm)                          1097d  1110d
access_capabilities
  SRMDAM(srmdammodule)                    63d    64d   472d   514d
  SRM_DRV(srm)                           247d   284d   315d   411d
access_code
  SRM_DRV                               2011c  2097c
  SRM_DRV(srm)                           713d   803d
access_code_type
  SRMDAM(srmdammodule)            ***    578d
  SRM_DRV(srm)                           180d   247d
access_date
  SRM_DRV(srm)                    ***    736d
access_not_ready_during_data_operation
  AMIGO(amigodvr)                 ***    815c
  AMIGO(csamigo)                  ***     78d
  MINIT(hminit)                   ***    321c
accs
  EDRIVER(edriver)                        57d   167c   176c   178c
acstrarray
  SRMDAM(srmdammodule)                   206d   318c
acstrarrtype
  SRMDAM(srmdammodule)                   203d   206d
act_tfr
  IOLIB(general_4)                      1269c  1420c
  KERNEL(iodeclarations)          ***    573d
  SRM_DRV(srm)                          1196c  1343c
action
  INITLOAD(sysglobals)                   215d   277d
  SYSDEVS(sysdevs)                       374d   492d
actionspending
  INIT                            ***   2542c
  INIT(misc)                             235c   237c   238c
  INITLOAD(sysglobals)            ***    254d
  KEYS(keys)                      ***     61c
  M68KSYS(ci)                     ***    860c
  SYSDEVS(sysdevs)                       499c   501c   504c   505c
```

```
active_controller
  GLE_HPIB(gle_hpib_io)                10142d 10148c 10151c 10236c 10324c
  IOLIB(hpib_1)                          397d   466d   472c   475c   558c   646c   715c   765c
  IOLIB(hpib_2)                         1824c  1875c
  IOLIB(hpib_3)                         2066c  2122c
active_isc
  DC_DRV(intdc)                          434c   533c
  DISCHPIB(dischpib)              ***    467c
  IOLIB(general_4)                      1270c  1300c  1320c  1333c  1336c  1378c  1569c  1596c  1662c
  KERNEL(iodeclarations)          ***    572d
  SRM_DRV(srm)                          1197c  1275c  1290c  1318c
actual
  SRM_DRV(srm)                           483d   721d   814d  1452c
actual_num_files
  SRMDAM(srmdammodule)                   595c   597c
  SRM_DRV(srm)                    ***    385d
actual_num_passwords
  SRMDAM(srmdammodule)                   534c   536c
  SRM_DRV(srm)                    ***    407d
actual_tfr_type
  KERNEL(iodeclarations)                 559d   573d
actualsize
  LIFDAM(lifmodule)                      485d   507c   508c
add
  INITLOAD                              1727d  1764c  1765c  1771c
  LIBRARIAN                             1486d  1491c  1494c  1503c  1505c  1522c
add_char_data
  GLE_HPGL(gle_hpgl_out)                7065d  7105c  7110c  7111c  7126c  7178c  7238c  7309c  7313c  7315c  7317c  7319c  7340c  7344c
                                        7366c  7370c  7409c  7442c  7455c  7476c  7495c  7497c  7571c  7579c  7614c
  GLE_HPGLI(gle_hpgl_in)               18038d 18071c 18104c 18144c 18204c 18207c 18209c 18211c 18213c 18251c 18261c 18283c 18297c 18307c
                                       18319c 18344c 18352c
add_parm_data
  GLE_HPGL(gle_hpgl_out)                7080d  7314c  7316c  7318c  7320c  7343c  7345c  7369c  7371c  7410c  7448c  7452c  7454c  7456c
                                        7496c  7498c
  GLE_HPGLI(gle_hpgl_in)               18053d 18208c 18210c 18212c 18214c
addit
  INITLOAD(loader)                ***    865d
  LIBRARIAN                              188c  1523c  1614c  2384c  2388c  2413c
addr1
  GLE_RGL(gle_ras_out)            ***   8027d
addr2
  GLE_RGL(gle_ras_out)            ***   8028d
addr3
  GLE_RGL(gle_ras_out)            ***   8029d
addr_record
  AMIGO(amigodvr)                        690c   959c
  AMIGO(csamigo)                         151d   392d
addr_record_cmd
  AMIGO(csamigo)                          51d   401c
addr_record_cmd_buf
  AMIGO(csamigo)                         394d   400c   401c
addr_to_listen
  GLE_HPIB(gle_hpib_io)                10275d 10357c 10516c
  IOLIB(general_2)                       920c   997c  1029c  1042c
  IOLIB(general_4)                ***   1415c
  IOLIB(hpib_1)                          405d   597d   679c   752c
addr_to_talk
  GLE_HPIB(gle_hpib_io)                10188d 10269c 10537c
  IOLIB(general_2)                       853c   941c   972c  1016c  1055c  1076c
  IOLIB(general_4)                ***   1401c
  IOLIB(hpib_1)                          403d   510d   591c   703c
```

```
addrec
  CTABLE(ctr)                    ***    385d
  INIT(ldr)                           2373d  2375c  2385d
  INITLOAD(loader)                     884d   917d   999d  1013d  1020d  1023d  1028d  1030d  1039d  1045d  1050d  1054d  1056d  1083d
                                      1136d  1226d  1248d  1250d  1392c  1422d  1477d  1504d  1568d  1590d  1602c
  LIBRARIAN                            53d    56d    61d    67d    82d    83d    86d   313d   314d   319d  1166d  1272d  1318d  1346d
                                      1368d  1473d  1480d  1482d  1484d  1487d  1633d  1634d  1654d  1715d  1717d  1726d  1805d  1851d
                                      1862d  1868d  1998d  2008d  2149d  2196d  2310d  2337d  2358d  2737d  2830d  2833d  2964c  3076d
                                      3214d
  SEGMENTER(asm)                       112d   178d
addref
  LIBRARIAN                           1486d  1565c  1566c  1574c
address
  CS80(cs80)                           355d   556d   584c
  DGL_C_IN(dgl_confg_in)              12044d 12049c 12050c 12090d 12097c
  DGL_C_OUT(dgl_confg_out)            11159d 11279c 11284c 11293c 11313c 11314c 11322c 11338c 11407d 11419c
  ETU                                    52d    501c   502c   511c   517c   542c   543c   550c   675c   679c   961c  1190c
  GLE_HPIB(gle_hpib_io)               10014d 10493c 10494c 10516c 10537c
  INITLOAD(loader)                     1099d  1107d  1108d  1209d
  IOLIB(hpib_2)                        1782d  1796d  1798d  1854d  1856c  1974d  1976c  1982d  1984c
  LIBRARIAN                              34d    84d   313d  1858d  2010d  2128d  2359d  2360d
  TAPEBKUP(cs80tbr)                      26d   100d   126c
address1
  SRM_DRV(srm)                         260d  1502c
address_bounds
  CS80(cs80)                     ***   138d
  CS80(cs80dsr)                  ***   678c
  TAPEBKUP(cs80tbdvr)                  558c   790c
address_for_msge_in
  DISCHPIB(dischpib)                   329d   364c   395c   452c
address_for_msge_out
  DISCHPIB(dischpib)                   270d   293c   316c   457c
address_found
  DGL_C_OUT(dgl_confg_out)            11408d 11417c 11420c 11469c
addressed
  DISCHPIB(dischpib)                   278c   337c   402c
  GLE_HPIB(gle_hpib_io)               10125c 10127c 10208c 10296c
  IOLIB(hpib_1)                          449c   451c   530c   618c
  KERNEL(general_0)              ***    1210c
  KERNEL(iodeclarations)               502d   516d   530d
  PRINTER(prtdvr)                ***    220c
adjust_echo
  LIB(dgl_lib)                        20145d 20194c
adjust_return_echo
  DGL_HPGLI(dgl_hpgli)           ***  18103c
  DGL_KNOB(dgl_knob)             ***  18146c
  GEN(dgl_gen)                         3044d  3252d
adjusted_cell_height
  GLE_STEXT(gle_stext)                 4024d  4042c  4043c
adjusted_cell_width
  GLE_STEXT(gle_stext)                 4023d  4038c  4039c
adr
  DGL_RAS(dgl_raster)                 17255d 17262c 17263c 17264c 17265c 17268c 17269c 17270c
  INITLOAD(loader)               ***    864d
  LIBRARIAN                            187c   190c   194c   195c   196c  1503c  1505c  1522c  1572c  1573c  1574c  1614c  1616c  2384c
                                      2387c  2395c  2413c
advance
  SYSDEVS(sysdevs)                     516d   529c   536c   553c
advance_page
  DGL_HPGL(dgl_hpgl)                  17088d 17129c 17160c
afl
  CS80(cs80)                     ***   218d
ala
  KEYS(keys)                           146d   165c
```

```
ale
  KEYS(keys)                           147d   166c
aleng
  INIT(fs)                             809d   810d   829d   831d
ali
  KEYS(keys)                           148d   167c
all
  INITLOAD(loader)                    1082d  1206d  1336c
  KEYS(keys)                           406d   411c
allcapabilities
  SRMDAM(srmdammodule)                  63d   313c
allists
  HEAPT(hpm)                            60d    75d   126d   136d   194d
allocate
  LIFDAM(lifmodule)                    716d   747c   759c   783c
  SEGMENTER(asm)                        44d   238c   244c
allocate_bkgnd_info
  AMIGO(amigodvr)                      580c   596c
  CS80(cs80dvr)                       1002c  1100c
  DISCHPIB(bkgnd)                       62d   105d
  MINIT(hminit)                  ***    467c
  MINIT(qminit)                        955c  1120c
  TAPEBKUP(cs80tbdvr)            ***    349c
allocate_bkgnd_info_proc
  CTABLE(scanstuff)                   1070d  1118c  1155c  1165c
allow_modification
  MINIT(qminit)                       1034d  1073c  1076c  1077c  1091c
allow_release_timeout
  CS80(cs80)                     ***    230d
  CS80(cs80dsr)                  ***    882c
allrealstuff
  MOREFSYS                       ***      4d
allresolved
  INIT(ldr)                           2439c  2442c  2453c  2459c  2464c
  INITLOAD(loader)                    1060d  1191c  1311c  1334c  1624c  1627c
  SEGMENTER(asm)                       197c   199c
alo
  KEYS(keys)                           149d   168c
alpha
  C_HOOK                         ***     45d
  DGL_TOOLS(dgl_tools)           ***  20023d
alpha50
  CRT(crt)                       ***    720c
  C_HOOK                         ***     40d
  DGL_TOOLS(dgl_tools)                20018d 20061c
  INITLOAD(sysglobals)           ***    286d
alpha_key
  KEYS(keys)                           196c   219c   221c   270c   433c   437c
  NONUSKBD1(non_us_kbd1)         ***     42c
  SYSDEVS(sysdevs)               ***    258d
alphabet
  KEYS(keys)                           239d   285c
alphacrt
  CRT                            ***    734c
  CRT(crt)                              35d   712d   716c   724c
alphacrtinit
  CRT(crt)                             663d   705c   721c
alphalist
  INITLOAD(loader)                    1212d  1243c
alphastate
  CRT(crt)                             273c   274c   283c   285c   286c   679c   680c
  DGL_RAS(dgl_raster)                 17492c 17493c
  KEYS(keys)                             383c   391c
  SYSDEVS(sysdevs)               ***    117d
```

```
alphatype
  CRT(crt)                          ***     708c
  C_HOOK                            ***     142c
  DGL_C_OUT(dgl_confg_out)              11286c 11296c 11335c 11342c 11346c
  KEYS(keys)                            233d   239d
  SYSDEVS(sysdevs)                  ***     113d
alreadyopen
  SRMDAM(srmdammodule)                  660d   708c   820c   823c   843c   849c
altmode
  CRT(crt)                              102d   148d
  GCRT(crtb)                        ***    81d
  SYSDEVS(sysdevs)                  ***    96d
alu
  KEYS(keys)                            150d   169c
alvinisr
  DC_DRV(extdc)                     ***    199d
  DC_DRV(intdc)                     ***    307c
alvinit
  DC_DRV(extdc)                     ***    198d
  DC_DRV(init_dc)                   ***    676c
am
  ETU                                   436c   458c   479c
  INIT(misc)                            469c   470c   471c   472c   473c
  INITLOAD(bootdammodule)               713c   714c
  INITLOAD(loader)                      1120c  1611c
  INITLOAD(sysglobals)              ***    97d
  LIBRARIAN                             133c   139c   2742c
  LIFDAM(lifmodule)                     296c   317c   920c   922c   923c
  M68KSYS(ci)                           119c   334c   335c   547c   548c   567c
  SRMDAM(srmdammodule)                  948c   949c   950c   1503c
  UCSD_DAM(ucsdmodule)                  82c    155c   164c   165c   166c   500c   503c   504c   623c   676c
amigo_class
  CTABLE(scanstuff)                     1096d  1121c  1125c
amigo_class_type
  CTABLE(scanstuff)                     1086d  1096d
amigo_dev_type
  AMIGO(amigodvr)                       493d   506d
  AMIGO(csamigo)                        43d    137d   160d   205d
amigo_tm_name
  CTABLE(ctr)                           364d   707c   716c   733c   752c
amigodvr
  AMIGO(amigodvr)                   ***    422d
amigoinit
  AMIGO                             ***    32d
amigoio
  AMIGO(amigodvr)                       430d   554d
```

```
amrequesttype
  AMIGO(amigodvr)                   ***    430d
  ASCII(asciimodule)                       6d     13d
  BUBBLES(bubble)                          29d    53d
  CRT(crt)                          ***    402d
  CS80(cs80dvr)                            987d   1094d  1168d  1214d
  EPROMS(eproms)                           8d     12d
  F9885(f9885dvr)                   ***    40d
  GCRT(crtb)                        ***    272d
  GLE_KNOB(gle_knob_in)             ***    18047d
  INIT(initunits)                          2139d  2148d  2152d  2156d  2228d
  INIT(misc)                               483d   515d   534d
  INITLOAD(bootdammodule)                  544d   747d
  INITLOAD(mini)                           309d   384d
  INITLOAD(sysglobals)                     66d    68d
  KEYS(keys)                        ***    72d
  M68KSYS(ci)                              306d   731d
  PRINTER(prtdvr)                          40d    47d
  SRMDAM(srmdammodule)                     40d    178d
  SYSDEVS(sysdevs)                         372d   376d   407d   432d   484d
  UCSD_AM(ucsd_am)                  ***    40d
  UNITIO(uio)                              56d    77d
amtable
  ASCII                             ***    319c
  INIT(misc)                               471c   472c   473c   700c   706c   720c
  INITLOAD                          ***    1705c
  INITLOAD(bootdammodule)                  713c   714c
  INITLOAD(sysglobals)              ***    262d
  LIFDAM(lifmodule)                        296c   920c   922c   923c
  SRMDAM(srmdammodule)                     948c   949c   950c
  UCSD_AM(ucsd_am)                  ***    271c
  UCSD_DAM(ucsdmodule)                     82c    164c   165c   166c
amtableptrtype
  INITLOAD(sysglobals)                     173d   262d
amtabletype
  INITLOAD(sysglobals)                     168d   173d
amtype
  CTABLE(ctr)                       ***    453d
  INIT(initunits)                   ***    2183d
  INITLOAD(sysglobals)                     68d    97d    145d   168d   279d
  SYSDEVS(sysdevs)                         119d   154d
an_isrib
  DMA_DRV(init_dma)                        169d   170d   182d   199d
answer
  ETU                                      191d   197c   202c   203c   205c   206c   208c   213d   215c   221d   245c   259c   260c   289d
                                           291c   869d   1009c  1010c
  MINIT(midecs)                            43d    47c    48c    49c    50c
  TAPEBKUP                                 974d   979c   980c   981c   982c   983c
any_9895_unit_missing
  CTABLE                                   1339d  1350c  1403c
any_to_ucsd
  CONVERT(convert_text)                    35d    40d
anychange
  LIFDAM(lifmodule)                        199d   320c   353c   375c   415c   428c   457c   461c   475c   699c   709c   713c   1095c  1124c
                                           1145c  1170c
anychar
  GLE_TYPES(gle_types)                     1009d  1010d
  KEYS(keys)                               52d    351c   352c   500c   536c   583c
anychar_ptr
  GLE_HPGL(gle_hpgl_out)                   7067d  7141d
  GLE_HPGLI(gle_hpgl_in)                   18040d 18080d
  GLE_TYPES(gle_types)              ***    1010d
  GLE_UTLS(gle_utls)                       21008d 21009d 21010d 21011d 21012d 21039d 21069d 21106d 21118d 21119d
```

```
anycharsavehook
  KEYS(keys)                          55d   352c   501c
anytomem
  ETU                                418d  1043c
aoff
  LIBRARIAN                         1807d  1821c  1824c  1831c
area_draw_mode
  GLE_RGL(gle_ras_out)              8049d  8514c
areaisdbcrt
  CRT(crt)                           635c   647c   654c
  GCRT(crtb)                         474c   485c
  SYSDEVS(sysdevs)          ***       62d
areyoualivepack
  SRMAM(srmammodule)        ***      155c
  SRM_DRV(srm)                      1004d  1529d
arithop
  LIBRARIAN                          561d  1075c
arithoptype
  LIBRARIAN                          549d   561d
arp
  INIT(ldr)                 ***     2376c
  INITLOAD(loader)                   917d  1164c  1393c  1603c
  LIBRARIAN                 ***     2965c
ascii_buffer
  DGL_C_IN(dgl_confg_in)    ***    12022d
  DGL_C_OUT(dgl_confg_out)  ***    11034d
  GLE_FILE(gle_file_io)              9029d  9031d
  GLE_HPGL(gle_hpgl_out)             7013d  7015d
  GLE_HPGLI(gle_hpgl_in)            18013d 18015d
  GLE_HPIB(gle_hpib_io)             10040d 10042d
ascii_buffer_ptr
  DGL_C_IN(dgl_confg_in)            12071d 12088d
  DGL_C_OUT(dgl_confg_out)          11372d 11404d
  GLE_FILE(gle_file_io)              9029d  9065c
  GLE_HPGL(gle_hpgl_out)             7013d  7044c  7060c  7072c  7086c  7122c  7144c  7171c  7228c  7336c  7362c  7387c  7405c  7434c
                                     7472c  7491c  7519c
  GLE_HPGLI(gle_hpgl_in)            18013d 18032c 18045c 18059c 18083c 18102c 18139c 18249c 18295c 18332c
  GLE_HPIB(gle_hpib_io)             10040d 10514c 10535c
ascii_buffer_space
  DGL_C_IN(dgl_confg_in)            12022d 12099c
  DGL_C_OUT(dgl_confg_out)          11034d 11424c
asciiam
  ASCII                     ***      319c
  ASCII(asciimodule)                  6d    13d
asciifile
  ASCII                              319c   320c   321c
  ETU                       ***      930c
  INITLOAD(sysglobals)      ***       50d
asciimodule
  ASCII                     ***      317d
  ASCII(asciimodule)        ***        3d
```

```
asm
  A804XDVR(a804xdvr)        ***       31d
  ASCII(asciimodule)        ***        4d
  CLOCK(clock)              ***       33d
  CONVERT(convert_text)     ***       27d
  CRT(crt)                  ***       33d
  CS80(cs80dvr)             ***      972d
  DGL_HPGL(dgl_hpgl)        ***    17014d
  DGL_POLY(dgl_poly)        ***    20039d
  DGL_RAS(dgl_raster)       ***    17013d
  EDRIVER(edriver)          ***       24d
  EPROMS(eproms)            ***        6d
  ETU                       ***       26d
  GCRT(crtb)                ***       33d
  GEN(dgl_gen)              ***     3053d
  HPHIL(hphil)              ***       30d
  INIT                      ***     2514d
  INIT(fs)                  ***      739d
  INIT(initunits)           ***     2135d
  INIT(isr)                 ***       34d
  INIT(ldr)                 ***     2270d
  INIT(misc)                ***      153d
  INITLOAD                  ***     1664d
  INITLOAD(bootdammodule)   ***      535d
  INITLOAD(loader)          ***      809d
  INITLOAD(mini)            ***      304d
  KEYS(keys)                ***       33d
  LIBRARIAN                 ***       27d
  M68KSYS                   ***     1211d
  M68KSYS(ci)               ***       33d
  MINIT(mminit)             ***      111d
  MOREFSYS(mfs)             ***       10d
  PRINTER(prtdvr)           ***       37d
  SEGMENTER(asm)                      42d    48d
  TAIL                      ***       32d
  UCSD_AM(ucsd_am)          ***       29d
  UCSD_DAM(ucsdmodule)      ***       29d
asm_bcd_real
  MOREFSYS(mfs)                       40d   503c
asm_bcdround
  MOREFSYS(mfs)                      552d   605c
asm_flpt_reset
  MOREFSYS                           682d   688c
asm_real_bcd
  MOREFSYS(mfs)                      556d   582c
asmr
  MINIT(asmr)               ***     1317d
  MINIT(xminit)             ***     1371d
aspecialtype
  KEYS(keys)                         234d   241d   243d
aspect_ratio
  DGL_INQ(dgl_inq)          ***     6228c
  DGL_VARS(dgl_vars)        ***     1047d
  GEN(dgl_gen)                      3480c  3484c  3493c  3504c  3517c  3528c
  LIB(dgl_lib)                     20362c 20363c 20366c 20370c 21413c
assembler
  M68KSYS(ci)                         39d   595c   610c   619c   699c   716c  1112c
assign_and_clear_unit
  CTABLE(ctr)                        303d   832d   855c
assign_both_flpy_units
  CTABLE                    ***     1324c
  CTABLE(ctr)               ***      273d
assign_both_units
  CTABLE                            1287d  1323c  1397c  1404c
```

```
assign_even_unit
  CTABLE                                1272c  1328c
  CTABLE(ctr)                            267d   272d   274d
assign_flpy_unit_pair
  CTABLE                                1268d  1502c  1514c  1607c  1629c  1631c
assign_neither_flpy_unit
  CTABLE                                1193d  1195d  1302c
  CTABLE(brstuff)              ***       933d
  CTABLE(ctr)                  ***       271d
assign_odd_unit
  CTABLE                                1277c  1327c
  CTABLE(ctr)                            267d   272d   274d
assign_only_this_unit
  CTABLE                                1287d  1384c
assign_temp_unitable
  CTABLE                       ***      1796c
  CTABLE(ctr)                            304d   848d
assoc
  LIBRARIAN                             3303d  3310c  3311c  3314c  3330c  3431c  3441c  3454c  3460c  3468c  3469c  3497c
async
  AMIGO(amigodvr)              ***       602c
  DISCHPIB(bkgnd)                         41d   120c   135c   149c
  UNITIO(uio)                             29d    31d    54d    62c    66c    75d    83c    87c
asynchronous
  AMIGO(amigodvr)                        559d   564c   602c   631c   647c
atn_line
  IOLIB(hpib_2)                         1821c  1829c  1876c
  KERNEL(iodeclarations)       ***       395d
atp
  CS80(cs80)                   ***       107d
attempt_copy
  TAPEBKUP                              1227d  1351c
attempt_to_write_on_protected_track
  AMIGO(amigodvr)              ***       881c
  AMIGO(csamigo)               ***        80d
aua
  KEYS(keys)                             151d   172c
aue
  KEYS(keys)                             152d   173c
aui
  KEYS(keys)                             153d   174c
auo
  KEYS(keys)                             154d   175c
auto_sparing_invoked
  CS80(cs80)                   ***       189d
  CS80(cs80dsr)                ***       726c
  TAPEBKUP(cs80tbdvr)                    608c   840c
auto_unload
  TAPEBKUP                              1428d  1443c  1444c  1451c  1462d  1480c  1485c
auu
  KEYS(keys)                             155d   176c
await_blanking
  GLE_GEN(gle_gen)             ***      2258c
  GLE_HPGL(gle_hpgl_out)       ***      7558c
  GLE_RGL(gle_ras_out)                  8410c  8478c
  GLE_TYPES(gle_types)         ***      1029d
await_locator
  LIB(dgl_lib)                         20076d 20951d
ayastatusptr
  SRM_DRV(srm)                          1407d  1469c
ayastatustype
  SRM_DRV(srm)                          1402d  1407d
ayatimeout
  SRM_DRV(srm)                          1122d  1415c
```

```
b
  AMIGO(csamigo)                         231d   236c
  CS80(cs80)                             373d   377c
  CS80(cs80dsr)                          785d   935c
  CTABLE(brstuff)                        971d   981c
  DGL_POLY(dgl_poly)                   20252d 20259c
  DGL_RAS(dgl_raster)                  17067d 17105c
  GEN(dgl_gen)                          3048d  3049d  3166d  3181c  3182c  3183c  3184c  3185c  3186c  3190d  3195d  3198c  3200c  3205d
                                        3208c  3210c  3219c  3220c  3229c  3239c
  GLE_RGL(gle_ras_out)                  8157d  8187c  8191c  8195c  8199c  8203c
  HPHIL(hphil)                            47d    57d   140c   146c   167c   172c   305c   311c   314c
  INIT(fs)                               813d   814d   836d   838d
  INITLOAD(mini)                         334d   342d   343d   344d   345d   346d   347d   348d   349d   350d   351d   352d   353d   354d
                                         355d   356d   357d   358d   359d   360d   361d   362d   363d   364d   365d   366d   367d   368d
                                         369d   370d   380c
  KERNEL(iocomasm)             ***       704d
  LIBRARIAN                              754d   757c   760c   764c   771c  1805d  1811c  1817c  1818c  1827c  1828c  1835c  1839c  1841c
  LIFDAM(lifmodule)                      230d   232c   235c   236c
  MINIT(iramigo)                         211d   217c
  MIUI                                    15d   103d   182c   198c   201c   210c   353c   477c
  MOUSE(mouse)                            52d   114c
  SRMDAM(srmdammodule)                   472d   486c   514d   542c
  SYSDEVS(sysdevs)                        41d   410d
  UCSD_DAM(ucsdmodule)         ***        55d
b0
  CRT(crt)                     ***       176d
  DGL_HPGL(dgl_hpgl)           ***     17252d
  DGL_RAS(dgl_raster)          ***     17556d
  EDRIVER(edriver)             ***        62d
  LIBRARIAN                              1808d  1811c  1813c  1817c  1825c
b1
  BAT(bat)                                67d    77c
  CRT(crt)                     ***       176d
  DGL_HPGL(dgl_hpgl)           ***     17252d
  DGL_RAS(dgl_raster)          ***     17556d
  SYSDEVS(sysdevs)                       186d   380d   437d   438c   445d
b10
  CRT(crt)                     ***       176d
  DGL_HPGL(dgl_hpgl)           ***     17250d
  DGL_RAS(dgl_raster)          ***     17554d
b11
  DGL_HPGL(dgl_hpgl)           ***     17250d
  DGL_RAS(dgl_raster)          ***     17554d
b12
  DGL_HPGL(dgl_hpgl)           ***     17249d
  DGL_RAS(dgl_raster)          ***     17553d
b13
  CRT(crt)                                174d   688c
  DGL_HPGL(dgl_hpgl)           ***     17249d
  DGL_RAS(dgl_raster)          ***     17553d
b14
  CRT(crt)                                106d   152d   174d
  DGL_HPGL(dgl_hpgl)           ***     17249d
  DGL_RAS(dgl_raster)          ***     17553d
  GCRT(crtb)                   ***        85d
  SYSDEVS(sysdevs)                        75d    99d
b15
  CRT(crt)                     ***       174d
  DGL_HPGL(dgl_hpgl)           ***     17249d
  DGL_RAS(dgl_raster)          ***     17553d
  MIUI                                   391d   415d
```

```
b2
  BAT(bat)                          67d    78c
  CRT(crt)                  ***     176d
  DGL_HPGL(dgl_hpgl)        ***   17252d
  DGL_RAS(dgl_raster)       ***   17556d
  EDRIVER(edriver)          ***      64d
  SYSDEVS(syscevs)                 186d   380d   437d   438c   445d
b26
  KEYS(keys)                       235d   236d   248d   250d
b3
  BAT(bat)                          67d    79c
  CRT(crt)                  ***     176d
  CS80(cs80)                ***      80d
  DGL_HPGL(dgl_hpgl)        ***   17252d
  DGL_RAS(dgl_raster)       ***   17556d
  EDRIVER(edriver)          ***      65d
  SYSDEVS(sysdevs)                 186d   380d   437d   438c   445d
b4
  BAT(bat)                          67d    80c
  CRT(crt)                  ***     176d
  CS80(cs80)                ***      80d
  DGL_HPGL(dgl_hpgl)        ***   17251d
  DGL_RAS(dgl_raster)       ***   17555d
  EDRIVER(edriver)          ***      54d
  SYSDEVS(sysdevs)                 186d   380d   437d   438c   445d
b5
  BAT(bat)                          67d    81c
  CRT(crt)                  ***     176d
  CS80(cs80)                ***      80d
  DGL_HPGL(dgl_hpgl)        ***   17251d
  DGL_RAS(dgl_raster)       ***   17555d
  SYSDEVS(sysdevs)                 186d   380d   437d   438c   445d
b6
  CRT(crt)                  ***     176d
  CS80(cs80)                ***      80d
  DGL_HPGL(dgl_hpgl)        ***   17251d
  DGL_RAS(dgl_raster)       ***   17555d
b7
  CRT(crt)                  ***     176d
  CS80(cs80)                ***      80d
b8
  CRT(crt)                  ***     176d
  DGL_HPGL(dgl_hpgl)        ***   17250d
  DGL_RAS(dgl_raster)       ***   17554d
b9
  CRT(crt)                          131d   176d
  DGL_HPGL(dgl_hpgl)        ***   17250d
  DGL_RAS(dgl_raster)       ***   17554d
  GCRT(crtb)                ***      64d
  SYSDEVS(sysdevs)                  74d    84d
b9826info
  CRT(crt)                          86d   720c
b_idle
  BUBBLES(bubble)                   71c    82c    83c    88c   101c   102c   115c
b_info
  DC_DRV(intdc)                    409d   416c   419c
  IOLIB(general_4)                1197d  1202d  1208d  1214d  1221d  1226d  1228d  1230d  1231d  1233d  1235d  1237d  1240d  1244d
                                  1247d  1257d  1262c  1285d  1288c  1296d  1298c  1314d  1317c  1319c  1320c  1328d  1330c  1354d
                                  1367c  1384c  1396c  1402c  1410c  1416c  1439d  1445c  1452c  1463d  1469c  1472c  1477c  1489d
                                  1493c  1494c  1502c  1523c  1532d  1538c  1541c  1543c  1546c  1551c  1557d  1561c  1567c  1584d
                                  1588c  1594c  1612d  1623c  1631c  1640d  1644c  1652c  1658d  1661c
  SRM_DRV(srm)                    1265d  1267c  1272d  1274c  1285d  1287c  1289c  1291c
b_max
  DGL_POLY(dgl_poly)             20523d 20529c 20533c
```

```
b_min
  DGL_POLY(dgl_poly)             20522d 20528c 20541c
b_w_mode
  DC_DRV(intdc)             ***     423c
  DISCHPIB(dischpib)        ***     472c
  IOLIB(general_4)                 1422c  1472c
  KERNEL(iodeclarations)    ***     575d
  SRM_DRV(srm)              ***    1345c
ba
  CS80(cs80dsr)             ***     621c
  CTABLE                           1191d  1243d  1246c  1274c  1278c  1468c  1505c  1508c  1531c  1540c  1554c  1574c  1596c  1600c
                                   1623c
  CTABLE(brstuff)                   934d   959d  1008c
  CTABLE(ctr)                       325d   328d   331d   332d   333d   334d   335d   336d   337d   414c   474c   497c   668d   671c
                                    673c   677d   680c   682c   698d   703c   707c   711d   714c   716c   720d   725c   733c   737d
                                    742c   752c   756d   765c   768c   772d   784c   787c   791d   795c   796c   798c   881c
  CTABLE(options)                   114d   118d   120d   123d   125d   127d
  CTABLE(scanstuff)         ***    1115c
  DISCHPIB(dischpib)                264c   270d   279c   293c   316c   329d   338c   364c   395c   429c   452c   457c
  INIT(initunits)           ***    2207c
  INITLOAD                  ***    1680c
  INITLOAD(sysglobals)      ***     147d
  MIUI                              35c   200c   351c
  PRINTER(prtdvr)           ***     275c
  SRMAM(srmammodule)        ***     168c
  SRMDAM(srmdammodule)      ***     417c
  SRM_DRV(srm)              ***    1379c
  TAPEBKUP                         1009c  1017c  1092c
background
  DGL_INQ(dgl_inq)          ***    6318c
  GLE_HPGL(gle_hpgl_out)    ***    7696c
  GLE_TYPES(gle_types)      ***    1091d
backspace
  CRT(crt)                          104d   127d   150d
  GCRT(crtb)                         60d    83d
  MOREFSYS(mfs)                      52d   124c   151c   183c   215c   255c   293c   326c   365c
  SYSDEVS(sysdevs)                   80d    97d
backup
  SYSDEVS(sysdevs)                  519d   563c   575c
backwardlist
  LIBRARIAN                        2369d  2420c  2422c  2463c
backwardpatches
  LIBRARIAN                          80d  1668c  2422c  2607c  2608c
bad_blocks
  SRM_DRV(srm)              ***     346d
bad_blocks_encountered
  TAPEBKUP(cs80tbdvr)               513d   638c   675c   683c
bad_track
  MINIT(hminit)                     350d   373c   376c   381c   383c
badch
  CRT(crt)                           98d   144d
  GCRT(crtb)                ***      77d
  SYSDEVS(sysdevs)          ***      94d
badcheckword
  F9885(f9885dvr)                    49d   230c   236c
badcommand
  ETU                               130d   206c   814c  1257c
  F9885(f9885dvr)                    48d   222c
baddrivetype
  MINIT(xminit)                    1407d  1591c  1596c  1598c  1600c  1601c  1603c
```

```
osgfile
  INIT(misc)                          714c   715c
  INITLOAD(sysglobals)          ***    47d
  LIFDAM(lifmodule)             ***   421c
  UCSD_DAM(ucsdmodule)          ***   595c
badio
  EDRIVER(edriver)                    105d   114c   116c   131c   135c   139c   141c   190c   220c   235c   249c   276c   291c
  ETU                                 119d   126c   133c   163c   168c   203c   329c   552c  1004c
  LIFDAM(lifmodule)                    86d   103c   171c   177c   189c   222c   309c   394c   488c   508c   509c   882c   896c  1065c
                                     1068c  1071c  1073c  1086c  1162c
badmessage
  ETU                                 123d   649c   662c   780c   783c   803c   921c   991c
badverify
  F9885(f9885dvr)               ***    49d
baseaddr
  ETU                                  48d   672c   675c   689c  1185c  1197c  1211c  1219c
basicbintype
  SRMDAM(srmdammodule)                 67d   939c
basicprogtype
  SRMDAM(srmdammodule)                 68d   940c
bat
  BAT                           ***    99d
  BAT(bat)                      ***    32d
bat8041cmdreg
  BAT(bat)                             48d    76c
bat8041datareg
  BAT(bat)                             49d    70c    87c
bat8041statusreg
  BAT(bat)                             47d    55c    63c
batbytereceived
  CLOCK(clock)                        148c   149c   153c   154c   155c
  SYSDEVS(sysdevs)                    381d   440d   442c
batcmdhook
  BAT(bat)                      ***    93c
  SYSDEVS(sysdevs)                    190d   438c   607c
batcmdtype
  SYSDEVS(sysdevs)                    185d   190d
batcommand
  BAT(bat)                      ***    92c
  CLOCK(clock)                         96c    97c   147c   148c   149c   153c   154c   155c
  SYSDEVS(sysdevs)                    380d   437d
batdataout
  BAT(bat)                             69d    77c    78c    79c    80c    81c
batinit
  BAT                           ***   103c
  BAT(bat)                             35d    90d
batreadbyte
  BAT(bat)                             85d    94c
batreadhook
  BAT(bat)                      ***    94c
  SYSDEVS(sysdevs)                    191d   442c   606c
batreadtype
  SYSDEVS(sysdevs)                    187d   191d
batstatus
  BAT(bat)                             52d    55c    56c    60d    63c    64c
batterypresent
  BAT                           ***   102c
  BAT(bat)                      ***    73c
  CLOCK(clock)                  ***   141c
  SYSDEVS(sysdevs)              ***   189d
bba
  CS80(cs80)                    ***   222d
  TAPEBKUP(cs80tbdvr)                 637c   639c
```

```
bbadcount
  BUBBLES(bubble)               ***    45c
bbaddata
  BUBBLES(bubble)               ***    47c
bbadinterupt
  BUBBLES(bubble)               ***    43c
bbadsector
  BUBBLES(bubble)               ***    44c
bbuffer
  BUBBLES(bubble)                      76c    93c
bcb
  DC_DRV(intdc)                       254d   408d   416c
  DI_DRV(extdi)                 ***   165d
  G_DRV(extg)                   ***   182d
  H_DRV(exth)                   ***   180d
  RS_DRV(rs)                    ***   182d
bcd
  CS80(cs80dsr)                       781d   919c
  CS80(cs80dvr)                      1034d  1064c
  INITLOAD(bootdammodule)             555d   556d
  LIFDAM(lifmodule)                    37d    41d
  MINIT(qminit)                       941d  1008c
  MIUI                                389d   390d
  MORFSYS(mfs)                  ***   552d
  TAPEBKUP(cs80tbdvr)                 460d   486c
bcd12
  MIUI                                390d   404d   413d   426d
bcd_prod_num
  CS80(cs80dsr)                       778d   916c   919c
  CS80(cs80dvr)                      1031d  1061c  1064c
bcd_product_number
  MINIT(qminit)                       938d  1005c  1008c
  TAPEBKUP(cs80tbdvr)                 457d   483c   486c
bcd_str
  MORFSYS(mfs)                        399d   406c   503c   562d   582c   583c   605c
bcd_strtype
  MORFSYS(mfs)                         32d    40d   399d   552d   556d   562d
bcddigit
  MORFSYS(mfs)                         31d    34d
bcount
  BUBBLES(bubble)                      78c    80c    96c    97c    99c
bdattype
  SRMDAM(srmdammodule)                 66d   938c  1047c
bduration
  A804XDVR(a804xdvr)            ***   388c
  SYSDEVS(sysdevs)                    173d   414c
bdx_chain_semaphore
  DISCHPIB(bkgnd)               ***    52d
  DISCHPIB(dischpib)                  486c   498c   501c
bdx_nt
  DISCHPIB(bkgnd)               ***    54d
  DISCHPIB(dischpib)                  461c   513c   514c   516c   517c   529c   534c
bdx_pre_eoi
  AMIGOTamigodvr)               ***   942c
  CS80(cs80dvr)                      1122c  1128c
  DISCHPIB(bkgnd)               ***    53d
  DISCHPIB(dischpib)                  530c   531c   533c   534c
bdx_proc
  DISCHPIB(bkgnd)               ***    55d
  DISCHPIB(dischpib)                  462c   542c
```

```
beep
  CRT(crt)                    ***    470c
  ETU                               810c   815c  1258c
  GCRT(crtb)                  ***    341c
  KEYS(keys)                        364c   403c   414c   563c
  LIBRARIAN                   ***     90c
  SYSDEVS(sysdevs)                  366d   413d   501c
beeper
  SYSDEVS(sysdevs)                  367d   416d
beeperhook
  A804XDVR(a804xdvr)          ***    389c
  SYSDEVS(sysdevs)                  172d   414c   417c   601c
beepop
  A804XDVR(a804xdvr)                157d   389c
belgian_kbd
  A804XDVR(a804xdvr)          ***    190d
  NONUSKBD2                         155c   219c
  SYSDEVS(sysdevs)            ***    146d
bell
  INIT(misc)                  ***    162d
bellchar
  CRT(crt)                    ***    470c
  GCRT(crtb)                  ***    341c
  INIT(misc)                        162d   371c
  LIBRARIAN                   ***     93c
  M68KSYS(ci)                       683c   725c
  PRINTER(prtdvr)             ***     45c
bep
  PRINTER(prtdvr)                    44d   191c   197c
berrortype
  BUBBLES(bubble)             ***     37d
between
  LIB(dgl_lib)                     20850d 20853c 20866c 20867c
bfast
  EDRIVER(edriver)                   28d   157c   210c   294c
bfrequency
  A804XDVR(a804xdvr)          ***    388c
  SYSDEVS(sysdevs)                  173d   414c
bi_type
  DISCHPIB(bkgnd)                    37d    58d    75d
bia_ptr
  DISCHPIB(bkgnd)                    78d    86c    87c    89c   100c   111c
bia_type
  DISCHPIB(bkgnd)                    75d    78d
big_graph
  C_HOOK                             40d   143c
  DGL_TOOLS(dgl_tools)             20018d 20032c
biggraphics
  CRT(crt)                          238c   253c
  INITLOAD(sysglobals)        ***    285d
bigptr
  ETU                                42d   419d   603d   856d
bigsize
  HEAPT(hpm)                         73d   105c   114c
bigtemp
  KERNEL(general_0)                 969d  1107c  1108c  1134c  1135c
bigtemp3
  KERNEL(general_0)                 970d  1046c  1048c  1099c  1100c  1113c  1114c  1142c  1143c  1149c  1150c
binand
  IOLIB(serial_0)                  2317c  2338c
  IOLIB(serial_3)                  2646c  2664c  2716c  2782c
  KERNEL(iocomasm)            ***    705d
```

```
bincmp
  IOLIB(serial_0)                  2317c  2338c
  KERNEL(iocomasm)            ***    711d
bineor
  KERNEL(iocomasm)            ***    709d
binior
  IOLIB(serial_0)                  2254c  2275c
  KERNEL(iocomasm)            ***    707d
bint
  ETU                                840d   954c  1170d  1185c  1190c  1191c  1197c  1211c  1215c  1216c  1219c
biofail
  BUBBLES(bubble)             ***     48c
bip
  DISCHPIB(bkgnd)                    94d   100c   107d   111c   114c   115c   116c
bip_type
  AMIGO(amigodvr)                   598c   675c   702c   720c   766c   777c   799c   938c
  CS80(cs80dvr)                    1090c  1101c
  DISCHPIB(bkgnd)                    58d    94d   107d   129c   130c   149c   157c   158c   165c
  DISCHPIB(dischpib)                447c   497c   508c   526c
  MINIT(cs80ir)                     759c   806c
  MINIT(hminit)               ***   308c
  MINIT(qminit)                     984c  1133c
  TAPEBKUP(cs80tbdvr)         ***   735c
  TAPEBKUP(cs80tbr)           ***   284c
bip_valid
  DISCHPIB(bkgnd)                    94d    98c   101c   129c   157c   165c
bit
  CS80(cs80dsr)                     786d   936c
  MINIT(qminit)                     947d  1017c
  TAPEBKUP(cs80tbdvr)               466d   493c
bit0
  C_HOOK                      ***    46d
  DGL_TOOLS(dgl_tools)        ***  20024d
  GLE_HPGLI(gle_hpgl_in)      ***  18243d
  GLE_RGL(gle_ras_out)        ***   8284c
  GLE_TYPES(gle_types)        ***   1246d
bit1
  C_HOOK                      ***    46d
  DGL_TOOLS(dgl_tools)        ***  20024d
  GLE_HPGLI(gle_hpgl_in)      ***  18242d
  GLE_TYPES(gle_types)        ***   1246d
bit10
  GLE_TYPES(gle_types)        ***   1245d
bit11
  GLE_HPGLI(gle_hpgl_in)      ***  18232d
  GLE_TYPES(gle_types)        ***   1245d
bit12
  GLE_HPGLI(gle_hpgl_in)      ***  18232d
  GLE_TYPES(gle_types)        ***   1245d
bit13
  C_HOOK                      ***     44d
  DGL_TOOLS(dgl_tools)        ***  20022d
  GLE_HPGLI(gle_hpgl_in)      ***  18232d
  GLE_TYPES(gle_types)        ***   1245d
bit14
  C_HOOK                      ***     44d
  DGL_TOOLS(dgl_tools)        ***  20022d
  GLE_HPGLI(gle_hpgl_in)      ***  18232d
  GLE_TYPES(gle_types)        ***   1245d
bit15
  GLE_HPGLI(gle_hpgl_in)      ***  18232d
  GLE_TYPES(gle_types)        ***   1245d
```

```
bit2
  C_HOOK                          ***     46d
  DGL_TOOLS(dgl_tools)            ***  20024d
bit3
  C_HOOK                          ***     46d
  DGL_TOOLS(dgl_tools)            ***  20024d
bit4
  C_HOOK                          ***     46d
  DGL_TOOLS(dgl_tools)            ***  20024d
  GLE_TYPES(gle_types)            ***   1246d
bit5
  C_HOOK                          ***     46d
  DGL_TOOLS(dgl_tools)            ***  20024d
bit6
  C_HOOK                               39d   46d
  DGL_TOOLS(dgl_tools)              20017d 20024d
  GLE_TYPES(gle_types)            ***   1246d
bit7
  C_HOOK                               39d   46d
  DGL_TOOLS(dgl_tools)              20017d 20024d
bit8
  GLE_TYPES(gle_types)            ***   1245d
  LIBRARIAN                             438d   658c   849c  1006c  1011c  1048c  1081c  1096c  1101c  1104c  1113c
bit9
  GLE_TYPES(gle_types)            ***   1245d
bit_mask
  C_HOOK                                81d   101c   104c   105c
  DGL_RAS(dgl_raster)               17382d 17409c 17412c 17413c
  GCRT(crtb)                            154d   177c   180c   181c
bit_set
  GLE_HPIB(gle_hpib_io)             10148c 10243c 10252c 10259c 10261c 10331c 10340c 10347c 10349c
  IOLIB(hpib_1)                         472c   487c   565c   574c   581c   583c   653c   662c   669c   671c
  IOLIB(hpib_3)                        2138c  2148c  2157c  2167c
  IOLIB(serial_0)                      2412c  2460c
  KERNEL(iocomasm)                ***   703d
bitcount
  LIBRARIAN                             752d   762c   773c   785c   791c   800c
bitmap
  INITLOAD(loader)                      911d   995d
bitmapaddr
  GCRT(crtb)                            536c   543c
  SYSDEVS(sysdevs)                ***   131d
bitmaptype
  DGL_C_OUT(dgl_config_out)         11183c 11288c 11302c 11332c
  GCRT(crtb)                      ***   520c
  SYSDEVS(sysdevs)                ***   113d
bitop
  LIBRARIAN                             653d   673c   678c
bitoptype
  LIBRARIAN                             651d   653d
```

```
bkgnd
  AMIGO(amigodvr)                 ***   425d
  AMIGO(csamigo)                  ***    37d
  CS80(cs80)                      ***    66d
  CS80(cs80dsr)                   ***   598d
  CS80(cs80dvr)                   ***   972d
  CS80(tapebuf)                   ***    35d
  DISCHPIB                        ***   553d
  DISCHPIB(bkgnd)                 ***    28d
  DISCHPIB(dischpib)              ***   189d
  MINIT(cs80ir)                   ***   516d
  MINIT(hminit)                   ***   248d
  MINIT(iramigo)                  ***   187d
  MINIT(qminit)                   ***   878d
  MIUI                            ***    11d
  TAPEBKUP                        ***   953d
  TAPEBKUP(cs80tbdvr)             ***   305d
  TAPEBKUP(cs80tbr)               ***    14d
bkgnd_and_dischpib_present
  CTABLE(scanstuff)                    1080d  1112c  1165c
blank
  MOREFSYS(mfs)                          51d    96c   370c   391c
blankchk
  ETU                                   868d   954c   955c   956c
blanks
  ETU                                  1163d  1178c  1184c  1186c  1193c  1198c  1212c  1218c
blink1_reg
  GLE_TYPES(gle_types)            ***  1275d
blink2_reg
  GLE_TYPES(gle_types)            ***  1276d
block
  CS80(cs80dvr)                        1168d  1172c  1177c  1220d  1244c  1250c  1267c  1271c  1278c  1282c  1293c  1298c  1307c  1310c
                                       1316c
  INIT(misc)                            537d   556c   559c   592c   627c   628c   652c   663c   669c
  INITLOAD(loader)                     1118d  1120c
  LIBRARIAN                             131d   133c   137d   139c
  UCSD_AM(ucsd_am)                       71d    75c    76c    92c    94c   135d   142c   144c
  UNITIO(uio)                            29d    31d    54d    65c    75d    86c
block_address
  CS80(cs80dvr)                        1095d  1110c
block_boundaries
  CTABLE                          ***  1570c
  CTABLE(ctr)                           315d   621d   623c   624c
block_os
  CTABLE(ctr)                           330d   331d   686d   692c   698d   705c
blockpower
  CS80(cs80dvr)                        1218d  1236c  1237c  1240c  1244c  1245c  1247c  1271c  1298c  1302c  1310c
blocks
  LIBRARIAN                             131d   133c
  UCSD_DAM(ucsdmodule)                  186d   192c   194c   195c   217c   218c   235c   570d   582c   583c   584c   603c
blocksize
  CS80(cs80dvr)                        1219d  1240c  1241c  1257c  1275c
  INITLOAD(loader)                      813d  1142c  1145c  1147c  1148c  1149c  1152c  1153c  1154c  1322c  1325c  1326c  1328c  1330c
  LIBRARIAN                            1366d  1980c  1984c  2116c  2174c  2206c  2211c  2215c  2230c  2231c  2253c  2259c  2269c  2270c
                                       2288c  2289c  2290c  2293c  2301c  2514c  2522c  2524c  2849c  2850c  2867c  2882c  3152c  3189c
                                       3190c
  M68KSYS(ci)                     ***   806c
blocktime
  CS80(cs80)                      ***   104d
blockwrite
  LIBRARIAN                             131d  1985c  2118c  2207c  2212c  2232c  2271c  2868c  2883c  3112c
blu_intensity
  GLE_RGL(gle_ras_out)                 8045d  8199c
```

```
blue
  DGL_INQ(dgl_inq)                    6053c  6056c
  DGL_POLY(dgl_poly)                 20079c 20088c
  DGL_RAS(dgl_raster)                17026d 17027d 17028d 17029d 17030d 17031d 17032d 17033d 17034d 17035d 17036d 17037d 17038d 17039d
                                     17040d 17041d 17111c 17181c 17187c 17191c 17225c 17235c 17240c 17275c 17629c
  DGL_VARS(dgl_vars)          ***     1026d
bminit
  MINIT(bminit)               ***       56d
  MIUI                        ***       12d
bminitialize
  MINIT(bminit)                        59d    62d
  MIUI                        ***      353c
bmp
  INITLOAD(loader)            ***      911d
  LIBRARIAN                            194c   399c   400c  1573c  1689c  1690c  1790c
bnode
  INITLOAD(bootdammodule)              782d   789c   790c   791c
bnoerror
  BUBBLES(bubble)                       40c   164c
bnotbubble
  BUBBLES(bubble)             ***       46c
boff
  LIBRARIAN                           1806d  1827c  1830c  1831c
boffset
  SRM_DRV                             1944d  1957c  2025d  2037c
  SRM_DRV(srm)                        1088d  1104d
boolproc
  SYSDEVS(sysdevs)                      41d   156d
bools
  CS80(cs80dvr)                       1040d  1069c
boot
  LIBRARIAN                           3118d  3138c  3166c
boot_dam_name
  CTABLE(ctr)                          350d   664c
boot_findfile
  INITLOAD(bootdammodule)              585d   665c
boot_flags
  INITLOAD(bootdammodule)              579d   741c
boot_id
  INITLOAD(bootdammodule)              578d   740c
boot_lifhead
  INITLOAD(bootdammodule)              584d   638c
boot_mfclose
  INITLOAD(bootdammodule)              590d   686c
boot_mfopen
  INITLOAD(bootdammodule)              588d   655c
boot_minit
  INITLOAD(bootdammodule)              587d   632c
boot_mread
  INITLOAD(bootdammodule)              591d   764c
boot_msus
  INITLOAD(bootdammodule)              577d   702c   718c   799c
boot_space
  INITLOAD(bootdammodule)              738d   743c
boot_start_address
  SRMDAM(srmdammodule)        ***      947c
  SRM_DRV(srm)                         502d   667d  1770c
boot_tm_name
  CTABLE(ctr)                          357d   664c
bootdam
  INIT(initunits)                     2159d  2248c
  INITLOAD                    ***     1677c
  INITLOAD(bootdammodule)              543d   607d
```

```
bootdammodule
  CTABLE                      ***     1186d
  INITLOAD                    ***     1664d
  INITLOAD(bootdammodule)     ***      534d
  SRMDAM(srmdammodule)        ***       35d
  TAIL                        ***       32d
bootdev_lun
  CTABLE                              1487c  1830c  1831c  1840c
  CTABLE(ctr)                          299d   500c
bootdev_msus
  CTABLE                              1484c  1485c  1486c
  CTABLE(ctr)                          298d   495c
booting
  LIBRARIAN                             47d   268c   271c  2836c  3166c  3169c  3349c  3513c  3533c  3541c  3546c  3624c  3628c  3636c
                                      3638c  3644c  3691c  3734c
bootmod
  LIBRARIAN                           2734d  2836c
bootnode
  INITLOAD                    ***     1761c
  INITLOAD(bootdammodule)              547d   795d   798c   800c
bootrev1
  INITLOAD(mini)                       387d   488c
bootrevflag
  INITLOAD(mini)                       390d   488c
boottm
  INIT(initunits)                     2155d  2248c
  INITLOAD                    ***     1678c
  INITLOAD(bootdammodule)              544d   747d
boottype
  CLOCK(clock)                          47d   175c   176c
bopfailed
  BUBBLES(bubble)             ***       42c
botbyte
  CRT(crt)                              52d   276c   288c   682c
  GCRT(crtb)                  ***       99d
bound
  INITLOAD(loader)                    1220d  1237c  1238c
bounds
  DGL_VARS(dgl_vars)                  1010d  1051d  1053d  1055d  1057d  1059d  1061d  1063d  1065d  1067d  1172d  1177d  1187d  1192d
bp
  DISCHPIB(discnpib)                   196d   198d   202d   287d   298c   299c   303c   357d   369c   372c   375c   440d   477c   479c
                                       480c
bphydata
  MINIT(bminit)                         60d    95d   101c
  MIUI                        ***      477c
bpt
  CS80(cs80dvr)                        979d  1075c  1076c
  CTABLE                              1383c  1746s
  CTABLE(ctr)                          288d   527d   528d   529d   530d   531d   532d   533d   534d   535d   536d   537d   538d   540d
                                       585c   586c   600c   611c   617c   623c   624c
bptr
  EDRIVER(edriver)                      89d   166c   177c
  ETU                                  841d   955c   956c  1169d  1195c  1203c
  MINIT(asmr)                         1329d  1337d  1348d
  MINIT(xminit)               ***     1386d
br_msus_type
  CTABLE(brstuff)                      952d   974d
brate
  EDRIVER(edriver)                      48d   334d   339c
break
  CRT(crt)                             100d   146d
  GCRT(crtb)                  ***       79d
  SYSDEVS(sysdevs)            ***       95d
```

```
brec
  BUBBLES(bubble)                    76c    93c
  EDRIVER(edriver)                   86d    97d
  ETU                              1167d  1185c  1190c  1191c  1195c  1197c  1203c  1211c  1215c  1216c  1219c
bretry
  BUBBLES(bubble)                    79c    98c
brightness_sequence
  GLE_RGL(gle_ras_out)             8065d  8353c  8362c  8363c  8364c
brstuff
  CTABLE                    ***    1186d
  CTABLE(brstuff)           ***     918d
brt
  GLE_RGL(gle_ras_out)             8322d  8352c  8357c  8359c  8360c  8361c
bs
  INIT(misc)                ***     163d
  KEYS(keys)                       362c   485c
bsize
  LIFDAM(lifmodule)                387d   399c   400c   402c   403c   442c
bstart
  BUBBLES(bubble)                    77c    94c
bt_already_logged
  MINIT(hminit)                    349d   374c   376c   377c
bt_count
  MINIT(hminit)                    345d   356c   375c   379c   380c   381c   389c
bt_index
  MINIT(hminit)                    351d   375c   376c   389c   391c
bt_table
  MINIT(hminit)                    352d   376c   381c   391c
btbs
  CS80(cs80)                ***    224d
btimeout
  BUBBLES(bubble)           ***     41c
bto
  CTABLE(ctr)                      328d   677d   681c   682c
bub_dvr
  BUBBLES(bubble)           ***     26d
bub_init
  BUBBLES                   ***    182c
  BUBBLES(bubble)                   34d   134d
bub_isr
  BUBBLES(bubble)                   32d   124d   166c
bub_tm
  BUBBLES(bubble)                   29d    53d
bubble
  BUBBLES                   ***    179d
  BUBBLES(bubble)           ***     24d
  CTABLE                           1258c  1421c
  CTABLE(brstuff)           ***     993d
  CTABLE(ctr)                      259d   554c   809c
bubble_card
  KERNEL(general_0)         ***    1106c
  KERNEL(iodeclarations)    ***     331d
bubble_dav
  CTABLE                           1210d  1422c  1449c  1781s
bubble_default_dav
  CTABLE                    ***    1449c
  CTABLE(options)           ***     122d
bubble_mp
  CTABLE(ctr)                      538d   554c
bubble_tm_name
  CTABLE(ctr)                      366d   809c
bubbles
  BUBBLES                   ***     23d
```

```
bubdoinr
  BUBBLES(bubble)           ***    131c
bubdorread
  BUBBLES(bubble)           ***     80c
bubdorreset
  BUBBLES(bubble)           ***    108c
bubdowrite
  BUBBLES(bubble)           ***     99c
bubgetinfo
  BUBBLES(bubble)           ***    163c
buf
  CRT(crt)                         406d   409c   419c   429c   439c   440c   442c   443c   444c   457c
  DGL_C_IN(dgl_config_in)         12071d 12079c 12088d 12099c 12100c
  DGL_C_OUT(dgl_config_out)       11372d 11393c 11404d 11424c 11425c
  ETU                              856d   940c  1043c  1053c  1088c  1098c  1110c
  GCRT(crtb)                       276d   279c   289c   299c   309c   310c   312c   313c   314c   328c
  GLE_KNOB(gle_knob_in)           18045d 18054c
  INIT(fs)                         783d   784d   786d   787d   789d
  INIT(misc)                       428d   458c
  INITLOAD(mini)                   392d   426c   428c   432c   441c   445c   509c   510c
  KEYS(keys)                        77d    80c   103c   118c   120c   122c   125c   128c
  M68KEYS(ci)                      311d   335c   341c   349c   351c   352c   356c   363c   374c
  PRINTER(prtdvr)                   80d   280c   307c   308c
  UCSD_DAM(ucsdmodule)             571d   584c   604c   606c
  UNITIO(uio)                       28d    30d    53d    65c    74d    86c
buf_empty
  DC_DRV(intdc)                    445c   447c
  DISCHPIB(dischpib)        ***    479c
  IOLIB(general_4)                1272c  1290c  1303c  1575c  1577c
  KERNEL(iodeclarations)    ***     582d
  SRM_DRV(srm)                    1199c  1268c  1280c  1390c  1430c  1458c
buf_fill
  DC_DRV(intdc)                    464c   466c   476c   494c   495c   498c   512c
  DISCHPIB(dischpib)        ***    480c
  IOLIB(general_4)                1273c  1290c  1302c  1321c  1602c  1604c
  KERNEL(iodeclarations)    ***     583d
  SRM_DRV(srm)                    1200c  1268c  1279c  1292c  1392c  1431c  1459c
buf_info
  DISCHPIB(bkgnd)           ***     56d
  DISCHPIB(dischpib)               464c   508c   518c   526c
buf_info_type
  DC_DRV(intdc)             ***    409d
  DISCHPIB(bkgnd)           ***     56d
  IOLIB(general_4)                1197d  1202d  1208d  1214d  1221d  1226d  1228d  1230d  1231d  1233d  1235d  1237d  1240d  1244d
                                  1247d  1257d  1285d  1296d  1314d  1328d  1354d  1439d  1463d  1489d  1532d  1557d  1584d  1612d
                                  1640d  1658d
  KERNEL(iodeclarations)    ***     570d
  SRM_DRV(srm)                    1125d  1265d  1272d  1285d
buf_ptr
  DISCHPIB(dischpib)        ***    477c
  IOLIB(general_4)                1267c  1272c  1273c  1302c  1303c  1321c
  KERNEL(iodeclarations)    ***     580d
  SRM_DRV(srm)                    1195c  1279c  1280c  1292c  1388c  1428c  1458c
buf_read_cmd
  AMIGO(amigodvr)           ***    725c
  AMIGO(csamigo)            ***     58d
buf_size
  DISCHPIB(dischpib)        ***    478c
  IOLIB(general_4)                1271c  1321c
  KERNEL(iodeclarations)    ***     581d
  SRM_DRV(srm)                    1198c  1292c  1389c  1429c  1457c
buf_type
  KERNEL(iodeclarations)           568d   580d
```

```
buf_write_cmd
  AMIGO(amigodvr)                  ***    726c
  AMIGO(csamigo)                   ***     59d
bufblocks
  LIBRARIAN                             2834d  2849c  2850c  2865c  2866c
bufend
  ASCII(asciimodule)                     16d    21c    22c    37c    50c    59c    70c    72c    74c    85c    93c   119c   158c   202c
                                        222c   247c   248c
buffer
  A804XDVR(a804xdvr)                    126d   128c   151d   154c
  AMIGO(amigodvr)                       431d   625c
  ASCII(asciimodule)                      7d    14d   185c   187c   194c   203c   223c   225c   251c   270c
  BUBBLES(bubble)                        30d    54d    76c    93c
  CRT(crt)                              402d   409c   415c   420c   430c
  CS80(cs80dvr)                         988d  1395c
  EDRIVER(edriver)                       38d    76d   199c   215c   217c   220c   230c   232c   235c   242c   243c   285c   296c   308d
                                        322c
  EPROMS(eproms)                          9d    13d    61c    68c
  ETU                                   419d   427c   564d   572c   578c   593c   603d   609c
  F9885(f9885dvr)                        40d   110d   281c
  GCRT(crtb)                            272d   279c   285c   290c   300c
  INIT(initunits)                      2139d  2148d  2152d  2156d  2228d
  INIT(misc)                            484d   503c   505c   516d   521c   525c   526c   535d   600c   605c   629c   631c   646c   649c
                                        659c   664c   675c   677c   684c
  INITLOAD(bootdammodule)               545d   592d   748d   764c   773c
  INITLOAD(mini)                        309d   384d   509c
  INITLOAD(sysglobals)             ***   69d
  KEYS(keys)                             72d    80c    94s   104c   118c   129c
  LIBRARIAN                            2736d  2770c  2771c  2772c  2798c  2800c  2807c  2809c  2818c  2821c
  M68KSYS(ci)                           306d   341c   383c   731d   738c   740c
  MINIT(asmr)                          1328d  1329d
  MIUI                                  453d   463c   489c   493c   495c   497c
  PRINTER(prtdvr)                        40d    47d    28Cc
  SRMAM(srmammodule)                     41d   179d   196c   203c
  SYSDEVS(sysdevs)                      241d   373d   377d   408d   433d   434c   485d   486c   529c   536c   552c   563c   575c   637c
  UCSD_AM(ucsd_am)                       41d   152c   176c   186c   214c
buffer_busy
  IOLIB(general_4)                     1247d  1658d  1662c  1663c
buffer_cleanup
  GLE_HPGL(gle_hpgl_out)               7057d  7127c  7348c  7374c  7411c  7458c  7477c
buffer_data
  IOLIB(general_4)                     1233d  1285d  1290c  1319c  1384c  1410c  1538c  1561c  1623c
buffer_fudge
  GLE_HPGL(gle_hpgl_out)               7010d  7061c
  GLE_HPGLI(gle_hpgl_in)           ***18010d
buffer_mode
  GLE_GEN(gle_gen)                 ***  2222c
  GLE_HPGL(gle_hpgl_out)           ***  7546c
  GLE_RGL(gle_ras_out)             ***  8421c
  GLE_TYPES(gle_types)             ***  1030d
buffer_reset
  IOLIB(general_4)                     1230d  1296d  1320c  1384c
buffer_space
  IOLIB(general_4)                     1231d  1314d  1321c  1396c  1493c  1541c  1588c  1644c
buffered_transfer
  AMIGO(amigodvr)                       681c   686c   687c   722c   752c
  DISCHPIB(bkgnd)                  ***   45d
buffersize
  CRT(crt)                              231d   243c   249c   256c
buffsize
  INIT(misc)                            484d   491c   503c   505c   516d   521c   525c   526c   535d   588c   590c   596c   599c   603c
                                        610c   628c   637c   638c   640c   683c
  UCSD_AM(ucsd_am)                       41d   176c   187c   190c   227c   257c
```

```
buflength
  ASCII(asciimodule)                     10d    21c    22c   265c
bufparams
  ASCII(asciimodule)                     18d    49c    85c   266c   295c   305c
bufptr
  CS80(cs80dvr)                        1095d  1116c  1215d  1250c  1268c  1269c  1278c  1283c  1295c  1296c  1307c  1308c  1318c
  ETU                                   423d   427c   428c   429c   431c   448c   449c   454c   458c   461c   462c   466c   474c   475c
                                        476c   478c   482c   606d   609c   613c   614c   615c   618c   619c   620c
  F9885(f9885dvr)                        82d   178c   213c   231c   282c
  LIBRARIAN                            2736d  2772c  2774c  2778c  2781c  2793c  2794c  2795c  2813c  2814c  2816c  2817c
  LIFDAM(lifmodule)                     388d   403c   444c   447c
  MINIT(xminit)                        1386d  1481c  1544c  1630c  1724c  1759c  1800c
bufrec
  ETU                                   422d   429c   430c   431c   436c   440c   443c   446c   448c   449c   452c   454c   476c   477c
                                        478c   479c
bufsize
  ASCII(asciimodule)                      7d    14d   182c   190c   195c   202c   219c   222c   226c   239c   245c   248c
  BUBBLES(bubble)                        30d    54d    74c    78c    91c    95c    96c    97c
  EDRIVER(edriver)                       39d    77d   115c   196c   199c   204c   225c   226c   267c   280c   288c   291c   309d   322c
  EPROMS(eproms)                          9d    13d    51c    54c    61c    67c    69c    72c
  ETU                                   860d   938c   939c   940c  1043c  1050c  1051c
  INIT(initunits)                      2140d  2149d  2153d  2157d  2229d
  INIT(misc)                            542d   546c   547c   548c   555d   560c   561c   563c   564c   567c
  INITLOAD(bootdammodule)               545d   748d   757c   764c   773c
  INITLOAD(sysglobals)             ***   69d
  M68KSYS(ci)                           307d   335c   346c   353c   354c   359c   365c   367c   371c   376c   383c
  SRMAM(srmammodule)                     42d   180d   196c   203c
  SYSDEVS(sysdevs)                      373d   377d   408d   433d   434c   485d   486c
buftype
  ETU                                    41d    42d
buildchar
  KEYS(keys)                             53d   503c   548c   553c
buildcount
  KEYS(keys)                             54d   503c   549c   550c
burn
  ETU                                    43d   583c   596c
burnit
  ETU                                   564d   620c  1082c  1098c  1104c  1110c
burst_tfr
  KERNEL(iodeclarations)           ***   562d
bus_address
  CTABLE                               1212d  1465c  1468c
  PRINTER(prtdvr)                        69d   115c   221c   275c
busaddr
  MINIT(asmr)                          1331d  1335d  1342d  1346d
busy
  A804XDVR(a804xdvr)                     54d   135c
  BAT(bat)                               42d    56c
  CRT(crt)                         ***   232d
  EDRIVER(edriver)                       59d   218c   233c   247c
  UNITIO(uio)                           118d   121c   122c
button
  DGL_HPGLI(dgl_hpgli)                18037d 18081c 18096c 18099c
  DGL_KNOB(dgl_knob)                  18037d 18111c 18128c 18131c
  DGL_VARS(dgl_vars)               ***   1141d
  GLE_HPGLI(gle_hpgl_in)           ***  18227d
  LIB(dgl_lib)                        20077d 20952d 20996c
bx_bufptr
  AMIGO(amigodvr)                        625c   768c   843c   846c   951c
  DISCHPIB(bkgnd)                  ***    49d
bx_length
  AMIGO(amigodvr)                        626c   748c   749c   952c   955c
  DISCHPIB(bkgnd)                  ***    50d
```

```
bx_strt_rcrd
  AMIGO(amigodvr)                    624c   690c   691c   739c   818c   827c   830c   866c   918c   950c   959c   960c
  DISCHPIB(bkgnd)          ***        48d
bx_tfr_length
  AMIGO(amigodvr)                    749c   750c   768c   846c   867c   868c   918c   950c   951c   952c
  DISCHPIB(bkgnd)          ***        51d
bx_tries
  AMIGO(amigodvr)                    623c   832c   834c   853c   860c   871c   904c   905c
  DISCHPIB(bkgnd)          ***        47d
byte
  A804xDVR(a804xdvr)                  34d    40d    41d    42d    69d    71d    74d    80d    91d   138d   142d   146d   157d   163d
                                     169d   181d   195d   258d   326d   379d
  AMIGO(amigodvr)          ***       508d
  AMIGO(csamigo)                      63d    64d    91d    92d   102d   121d   145d   229d   231d   245d   246d   278d   279d
  BAT(bat)                            44d    67d    69d    85d
  BUBBLES(bubble)                    139d   140d
  CLOCK(clock)                        41d    42d    43d    44d
  CRT(crt)                            52d   263d
  CS80(cs80dvr)                      984d   993d  1017d
  CS80(tapebuf)            ***       41d
  CTABLE                            1214d  1372d
  CTABLE(scanstuff)                 1042d  1054d  1095d  1134d
  DISCHPIB(dischpib)                 196d   198d   202d   270d   287d   329d   357d   440d
  EDRIVER(edriver)                    54d    62d    63d    64d    65d    89d
  EPROMS(eproms)                      26d    27d
  ETU                                836d  1157d
  GCRT(crtb)               ***       99d
  GLE_HPGLI(gle_hpgl_in)            18223d 18229d
  HPHIL(hphil)                        47d    57d    71d    96d   133d   160d   229d   282d   302d
  INIT(isr)                           40d    41d    42d    47d    48d    49d    52d    62d    63d    64d    80d    81d    82d   111d
  INIT(misc)               ***       207d
  INITLOAD(bootdammodule)             570d   571d   572d   573d
  INITLOAD(loader)                    843d   926d   944d  1171d
  INITLOAD(mini)           ***       412d
  INITLOAD(sysglobals)                29d   146d   147d   148d   149d   206d   207d
  KEYS(keys)                         235d   355d   373d
  LIBRARIAN                ***       442d
  MINIT(asmr)                        1325d  1328d
  MINIT(hminit)            ***       337d
  MINIT(iramigo)                     192d   211d   221d   228d   229d
  MINIT(qminit)                      946d  1016c
  MINIT(xminit)            ***      1462s
  MOUSE(mouse)                        41d    52d    57d    75d   129d
  PRINTER(prtdvr)          ***       69d
  SEGMENTER(asm)           ***       76d
  SEGMENTER(segmenter)     ***       25d
  SRM_DRV(srm)                       297d   299d   300d   301d   302d   303d   304d  1403d  1404d
  SYSDEVS(sysdevs)                    37d    39d    40d    66d   158d   173d   185d   186d   187d   262d   263d   313d   338d   348d
                                     349d   367d   369d   371d   380d   381d   400d   402d   405d   416d   421d   422d   428d   429d
                                     437d   440d   441d   445d   447d
  TAPEBACKUP(cs80tbdvr)              465d   492c
byte0
  CLOCK(clock)                        44d    96c   149c   155c
byte1
  CLOCK(clock)                        43c    96c   148c   154c
byte2
  CLOCK(clock)                        42d    96c   153c
byte3
  CLOCK(clock)             ***       41d
byte_array
  GLE_FGL(gle_ras_out)               8098d  8101d
byte_offset
  SRM_DRV                           1957c  2037c
  SRM_DRV(srm)                       681d   755d
```

```
byte_per_pixel_display
  GLE_FGL(gle_ras_out)     ***      8074d
bytealphabettype
  KEYS(keys)                         236d   247d
  NONUSKBD1(non_us_kbd1)             101d   105d
bytecount
  INITLOAD(bootdammodule)  ***       591d
  SRMAM(srmammodule)                  52d    59c    65c    73c    84c    88c    99d   106c   112c   120c   131c   135c
byteext
  LIBRARIAN                          454d   529c   825c   826c   831c   832c
byteoffset
  AMIGO(amigodvr)          ***      624c
  BUBBLES(bubble)                     77c    94c
  CS80(cs80dvr)            ***      1395c
  CTABLE(ctr)                        477c   900c
  EPROMS(eproms)                      46c    59c    81c   106c
  F9885(f9885dvr)          ***      286c
  INIT(initunits)          ***      2210c
  INIT(misc)               ***      267c
  INITLOAD                 ***      1683c
  INITLOAD(mini)           ***      514c
  INITLOAD(sysglobals)     ***      160d
  M68KSYS(ci)                        738c   740c   813c
  MIUI                               509c   514c
bytes
  ETU                                605d   611c   613c   614c   616c   618c   620c   623c   626c   628c   632c
  UCSD_AM(ucsd_am)                    71d    82c    83c    88c    90c    95c    97c    98c
bytes_wide
  DGL_RAS(dgl_raster)               17318d 17331c 17339c
bytesiz
  LIBRARIAN                          417d   418d   548d   675c   679c   680c   699c   715c   883c   907c  1007c  1043c  1122c
bytesize
  HEAPI(hpm)                          46d    47d   288c   309c
bytesleft
  LIBRARIAN                          425d   474c   475c  1249c  1259c  1265c
bytesperline
  GLE_FGL(gle_ras_out)     ***      8040d
```

```
c
  A804XDVR(a804xdvr)                195d   201c   202c   203c   206c   207c   208c   258d   265c   266c   294c   295c
  AMIGO(amigodvr)          ***      915c
  AMIGO(csamigo)           ***      113d
  ASCII(asciimodule)                89d    94c
  CONVERT(convert_text)             91d    96c    97c   100c   103c   104c   107c   111c   114c   115c   118c
  CRT(crt)                         183d   192c   193c   194c   350d   362c   364c   366c   384c   386c   387c   389c   390c   404d
                                   457c   459c   476c   478c   479c   481c   482c   485c   502d   509c   537c   615c   619c
  CS80(cs80)                       406d   411c   412c
  DC_DRV(extdc)            ***     201d
  DGL_HPGL(dgl_hpgl)             17029d 17052c
  DGL_RAS(dgl_raster)            17583d 17637c 17638c
  ETU                              130d   132c
  GCRT(crtb)                       117d   132d   251d   260c   261c   262c   267c   274d   328c   330c   347c   348c   352c   370d
                                   381c   406c   451c   456c
  GEN(dgl_gen)                    3195d  3198c  3200c  3202c  3205d  3208c  3210c  3212c
  INIT(misc)                       248d   252c   253c   539d   581c   582c
  INITLOAD                        1738d  1750c  1751c  1752c
  KEYS(keys)                       378d   457c   464c   469c   470c   471c
  LIBRARIAN                         98d   101c   103c  1373d  1376c  1377c  2012d  2095c  2096c
  M68KSYS(ci)                      312d   315d   319c   322c   323c   324c   375c   379c   407d   414c   417c   447c   450c   479c
                                   482c   483c   484c   486c   497c   510c   518c   658d   713c   715c   727c
  MINIT(hminit)            ***     325c
  MIUI                              15d   178c   202c   351c   474c
  SYSDEVS(sysdevs)                  52d    61d   378d   389d   391d   393d   487d   512d   529c   533c   536c   538c   563c   573c
                                   575c   580d   582c   587d   588c
  UCSD_AM(ucsd_am)                  46d   113d   117c   179c   180c   189c   191c   199c   201c   203c   230c   234c   235c   242c
                                   245c   248c   251c   253c   256c
c1
  SYSDEVS(sysdevs)                 581d   582c
c_def
  DGL_RAS(dgl_raster)            17022d 17026d 17027d 17028d 17029d 17030d 17031d 17032d 17033d 17034d 17035d 17036d 17037d 17038d
                                 17039d 17040d 17041d
  DGL_VARS(dgl_vars)              1023d  1029d
c_hook
  C_HOOK                   ***     33d
c_map
  C_HOOK                           54d   132c   134c
c_map_def
  C_HOOK                           49d    54d
c_table_ptr
  DGL_RAS(dgl_raster)            17068d 17109c
cache_off
  EDRIVER(edriver)         ***    163c
cache_on
  EDRIVER(edriver)         ***    189c
calc_intercept
  DGL_POLY(dgl_poly)             20433d 20647c 20649c
calc_rate
  IOLIB(serial_3)                 2532d  2552c  2563c
calc_soft_text_xform
  GLE_HPGL(gle_hpgl_out)   ***    7545c
  GLE_RGL(gle_ras_out)     ***    8414c
  GLE_TYPES(gle_types)     ***    1172d
calc_text_xform
  DGL_VARS(dgl_vars)       ***    1291d
  GLE_STEXT(gle_stext)             4054c  4064c  4076c  4088c  4100c
  GLE_TYPES(gle_types)     ***    1171d
  LIB(dgl_lib)                   20317c 20345c 20528c 20716c 20721c 21442c
calc_vertex_info
  DGL_POLY(dgl_poly)             20513d 20685c
calculate_viewing
  GEN(dgl_gen)                    3021d  3385d  3631c
  LIB(dgl_lib)                   20313c 20341c
```

```
call_segment
  SEGMENTER(asm)           ***    255d
  SEGMENTER(segmenter)     ***     32d
call_segment_proc
  SEGMENTER(asm)           ***    278d
  SEGMENTER(segmenter)     ***     33d
callit
  SEGMENTER(asm)                   257d   275c   281c   286c
can_nonadv
  KEYS(keys)                       160c   261c
  NONUSKBD1(non_us_kbd1)   ***    146c
  SYSDEVS(sysdevs)         ***    277d
candownscroll
  CRT(crt)                 ***    118d
  GCRT(crtb)               ***     51d
  SYSDEVS(sysdevs)         ***     71d
canupscroll
  CRT(crt)                 ***    117d
  GCRT(crtb)               ***     50d
  SYSDEVS(sysdevs)         ***     71d
cap
  SRMDAM(srmdammodule)             201d   276c   307c
capabilities
  SRMDAM(srmdammodule)             304c   313c   320c   486c   542c   615c  1068c  1072c  1073c  1105c  1106c  1401c
  SRM_DRV(srm)                     284d   315d   411d
card
  BUBBLES(bubble)                   56d    63c    80c    99c   108c   146d   156c   157c   163c   167c
  EDRIVER(edriver)                  96d   134c   135c   139c   153c   207c
card_addr
  BUBBLES(bubble)          ***    131c
  KERNEL(general_0)        ***   1206c
  KERNEL(iodeclarations)            496d   510d   524d
card_id
  BUBBLES(bubble)                  153c   154c   160c
  DC_DRV(init_dc)                  631c   632c   633c   635c   639c   671c   679c   680c   686c
  DC_DRV(intdc)                    275c   276c   277c   293c   391c   392c
  DISCHPIB(dischpib)               233c   240c   280c   339c   343c   483c   510c
  DI_DRV(init_discint)             242c   281c
  DMA_DRV(init_dma)        ***     236c
  EDRIVER(edriver)                 136c   137c   149c   151c
  F9885(f9885dvr)          ***     262c
  G_DRV(init_gpio)                 257c   293c
  H_DRV(init_hpib)                 258c   259c   304c
  IOLIB(serial_0)                 2240c  2260c  2303c  2323c  2369c  2417c
  IOLIB(serial_3)                 2560c  2571c  2572c  2607c  2636c  2637c  2703c  2711c  2712c  2745c  2765c  2766c  2810c  2817c
                                  2845c  2852c
  KERNEL(general_0)                986c   994c   998c  1019c  1039c  1045c  1053c  1060c  1078c  1084c  1090c  1105c  1130c  1140c
  KERNEL(iodeclarations)   ***    543d
  PRINTER(prtdvr)          ***    224c
  RS_DRV(init_rs)                  247c   278c
  SRMAM(srmammodule)       ***    150c
card_ptr
  BUBBLES(bubble)                   63c   156c
  DC_DRV(init_dc)          ***    656c
  DI_DRV(init_discint)     ***    261c
  DMA_DRV(init_dma)                267c   273c
  EDRIVER(edriver)         ***    134c
  F9885(f9885dvr)          ***    263c
  G_DRV(init_gpio)         ***    276c
  H_DRV(init_hpib)         ***    284c
  KERNEL(general_0)                857c   876c   896c   915c  1002c  1020c  1030c  1206c
  KERNEL(iodeclarations)   ***    544d
  MIUI                     ***    350c
  RS_DRV(init_rs)          ***    264c
```

```
card_type
  BUBBLES(bubble)                      153c   154c   159c
  DC_DRV(init_dc)                      636c   687c
  DISCHPIB(dischpib)            ***    248c
  EDRIVER(edriver)                     135c   136c   137c   149c   151c
  GLE_HPIB(gle_hpib_io)              10146c 10210c 10234c 10298c 10322c
  IOLIB(hpib_1)                        470c   485c   532c   556c   620c   644c   713c   763c
  IOLIB(hpib_3)                 ***   2120c
  IOLIB(serial_0)                     2238c  2301c  2367c
  IOLIB(serial_3)                     2558c  2605c  2689c  2743c  2808c  2843c
  KERNEL(general_0)                    989c  1014c  1018c  1038c  1044c  1052c  1059c  1066c  1079c  1085c  1091c  1101c  1106c  1115c
                                      1131c  1141c  1152c
  KERNEL(iodeclarations)        ***    541d
  PRINTER(prtdvr)                      121c   214c   235c   290c
cardrec
  BUBBLES(bubble)                      138d   146d
casecocestart
  LIBRARIAN                            309d  1222c  1227c  1234c
casetable
  LIBRARIAN                            298d  1223c  1225c
cat
  LIFDAM(lifmodule)                    482d   489c   561d   564c   580d   607c
  M68KSYS(ci)                          791d   818c   821c
  UCSD_DAM(ucsdmodule)                 530d   645c
cat_info
  SRMDAM(srmdammodule)          ***    600c
  SRM_DRV(srm)                  ***    386d
catalog
  INITLOAD(sysglobals)          ***    136d
  LIFDAM(lifmodule)             ***   1152c
  SRMDAM(srmdammodule)                1651c  1700c
  UCSD_DAM(ucsdmodule)          ***    645c
catalogue_organization
  SRM_DRV(srm)                         338d  1498c
catarray
  LIFDAM(lifmodule)                     76d   580d
  SRMDAM(srmdammodule)                 459d   460d   502d   503d   565d   566d
  UCSD_DAM(ucsdmodule)                  46d   477c   511c   542c
catarrayptr
  SRMDAM(srmdammodule)                 460d   482c   503d   539c   566d   600c
catentry
  ETU                                  353d   867d
  INIT(misc)                    ***    179d
  LIFDAM(lifmodule)                     76d   482d   561d
  M68KSYS(ci)                   ***    791d
  MIUI                          ***    375d
  SRMDAM(srmdammodule)                 381d   449d   565d   989d
  UCSD_DAM(ucsdmodule)          ***     46d
catentryindx
  SRMDAM(srmdammodule)                 469d   511d   517c   527c   537c   539c   546c   557c   577d   580c   586c   598c   600c   630c
                                       640c
catentryptr
  SRMDAM(srmdammodule)                 989d   996c
catindx
  SRMDAM(srmdammodule)                 468d   510d   526c   530c   553c   576d   585c   589c   637c
catpack
  SRMDAM(srmdammodule)          ***    588c
  SRM_DRV(srm)                        1005d  1547d
catpasspack
  SRMDAM(srmdammodule)          ***    529c
  SRM_DRV(srm)                        1013d  1585d
catpasswords
  INITLOAD(sysglobals)          ***    138d
  SRMDAM(srmdammodule)                1652c  1702c
```

```
catr
  CS80(cs80)                    ***    105d
cb_index
  AMIGO(amigodvr)                      795d   844c   845c   848c   850c
cb_ptr
  AMIGO(amigodvr)                      794d   843c   846c   848c   849c
cblocksize
  INIT(misc)                    ***    189d
  LIFDAM(lifmodule)                    568c   615c
  SRMDAM(srmdammodule)                 410c   610c
  UCSD_DAM(ucsdmodule)                 521c   550c
cbuildtable
  GCRT(crtb)                           123d   508c
cchar
  GCRT(crtb)                           117d   352c   381c   386c   392c   401c   403c   406c   439c   447c   456c
cclear
  GCRT(crtb)                           121d   234c   395c   464c   466c
ccreatedate
  INIT(misc)                    ***    190d
  LIFDAM(lifmodule)                    574c   616c
  SRMDAM(srmdammodule)                 411c   435c   608c
  UCSD_DAM(ucsdmodule)                 522c   551c
ccreatetime
  INIT(misc)                    ***    191d
  LIFDAM(lifmodule)                    574c   616c
  SRMDAM(srmdammodule)                 412c   435c   608c
  UCSD_DAM(ucsdmodule)                 523c   525c   552c   555c
ccrunch
  INIT(fs)                      ***    744d
cd
  TAPEBKUP(cs80tbr)                    219d   234c   235c   236c   237c   238c   239c   240c   241c   242c   243c   244c   245c   246c
                                       247c   248c   249c
cdate
  LIFDAM(lifmodule)                     54d   515c   574c
cdbhigh
  GCRT(crtb)                           132d   451c
cdbscroll
  GCRT(crtb)                           130d   436c
cdbscrollr
  GCRT(crtb)                           131d   444c
cdn_eng_kbd
  A804XDVR(a804xdvr)            ***    187d
  NONUSKBD2                            173c   219c
  SYSDEVS(sysdevs)              ***    146d
cdn_fr_kbd
  A804XDVR(a804xdvr)            ***    190d
  NONUSKBD2                            173c   219c
  SYSDEVS(sysdevs)              ***    146d
ceft
  INIT(misc)                    ***    181d
  LIFDAM(lifmodule)             ***    610c
  SRMDAM(srmdammodule)          ***    603c
  UCSD_DAM(ucsdmodule)          ***    546c
centered
  DGL_INQ(dgl_inq)              ***   6311c
  DGL_RAS(dgl_raster)           ***  17594c
  DGL_VARS(dgl_vars)            ***   1079d
  GEN(dgl_gen)                        3488c  3512c
  LIB(dgl_lib)                  ***  21449c
```

```
centisecond
  A804XDVR(a804xdvr)                280c   309c
  CLOCK(clock)                       81c   120c
  INITLOAD(sysglobals)        ***   226d
  LIBRARIAN                   ***   235c
  LIFDAM(lifmodule)                 125c   126c   557c   588c
  M68KSYS(ci)                       125c   127c   278c   279c
  MIUI                        ***    84c
  SRMDAM(srmdammodule)              375c   388d
  UCSD_DAM(ucsdmodule)              523c   553c
ceoln
  M68KSYS(ci)                       317d   321c   323c
certify
  TAPEBKUP                         1457d  1508c
  TAPEBKUP(cs80tbdvr)               413d   425c
cexchange
  GCRT(crtb)                        126d   480c
cextra1
  INIT(misc)                  ***   192s
  LIFDAM(lifmodule)                 504c   571c   618c
  M68KSYS(ci)                 ***   823c
  MIUI                        ***   523c
  SRMDAM(srmdammodule)              405c   611c
  UCSD_DAM(ucsdmodule)              481c   483c   518c   556c
cextra2
  INIT(misc)                  ***   195s
  LIFDAM(lifmodule)                 572c   618c
  SRMDAM(srmdammodule)              408c   612c
  UCSD_DAM(ucsdmodule)              520c   557c
cfib
  ETU                                56d   248c   250c   257c   260c   270c   276c   280c   281c   335c   341c   359c   361c   370c
                                    389c   399c   405c   890c   893c   904c   923c   926c  1023c  1043c  1052c
cfile
  ETU                         ***    64d
cget
  A804XDVR(a804xdvr)          ***   241c
  CLOCK(clock)                       53c   103c   117c
  SYSDEVS(sysdevs)                  199d   460c
cgetdate
  CLOCK(clock)                ***   166c
  SYSDEVS(sysdevs)                  198d   464c
cgettime
  CLOCK(clock)                ***   167c
  SYSDEVS(sysdevs)                  198d   467c
ch
  ETU                               187d   188c   295d   298c   299c
  GLE_KNOB(gle_knob_in)           18109d 18118c 18136c 18152c 18157c 18164c
  INIT(fs)                          801d   802d   817d   819d
  LIBRARIAN                         265d   277c   278c
  M68KSYS(ci)                        71d   211d   214c   215c   217c   233d   236c   237c   239c   240c   241c   301c   302c   303c
                                    572d   575c   576c
  MINIT(qminit)               ***  1058d
  MOREFSYS(mfs)                     403d   418c   420c   421c   423c   424c   427c   430c   433c   438c   440c   443c   447c   449c
                                    453c   456c   461c   463c   467c   469c   471c   475c   477c   478c   482c   485c   486c
  SRMDAM(srmdammodule)             1527d  1551c  1556c  1561c  1568c  1578c  1586c
  SWVOL                               6d    24c    25c    26c
  TAPEBKUP                          966d   968c  1494d  1504c  1505c  1507c  1518c  1524c
chain
  M68KSYS(ci)                        72d   100d
chainfile
  M68KSYS(ci)                        57d    95c   102c   103c  1038c  1069c
chainflag
  INITLOAD(sysglobals)        ***   208d
```

```
chaining
  M68KSYS(ci)                        58d    96c   104c   141c  1035c  1037c  1066c  1068c  1071c  1098c  1171c
chan
  F9885(f9885dvr)                   112d   154c   155c   156c   178c
chan_indep_clr
  CS80(cs80)                        339d   381d   397c
  CS80(cs80dsr)               ***   897c
change
  GLE_HPGL(gle_hpgl_out)           7097d  7102c  7104c  7106c
change_state
  GLE_HPGL(gle_hpgl_out)           7093d  7125c  7177c  7342c  7368c  7408c  7437c  7475c  7494c
changecursor
  GCRT(crtb)                        118d   351c   353c   437c   440c   445c   448c   454c   458c   479c   484c
changed
  LIFDAM(lifmodule)                 389d   417c   424c   432c   439c   455c
  M68KSYS(ci)                       175d   254c   263c   274c   282c
changefname
  LIFDAM(lifmodule)                1060d  1133c
changeit
  UCSD_DAM(ucsdmodule)              437d   640c
changename
  INITLOAD(sysglobals)        ***   134d
  LIFDAM(lifmodule)           ***  1133c
  SRMDAM(srmdammodule)              1659c  1764c
  UCSD_DAM(ucsdmodule)        ***   640c
changeprotectpack
  SRMDAM(srmdammodule)              492c  1404c
  SRM_DRV(srm)                     1021d  1623d
changevolpack
  SRM_DRV(srm)                     1029d  1658d
channel_is_setup
  PRINTER(prtdvr)                    71d   212c   229c   248c   281c
channel_parity_error
  CS80(cs80)                  ***   133d
  CS80(cs80dsr)                     654c   667c
  TAPEBKUP(cs80tbdvr)               534c   547c   762c   779c
char_height
  GLE_SMARK(gle_smark)        ***  5126c
  GLE_STEXT(gle_stext)              4041c  4050c  4062c  4066c
  GLE_TYPES(gle_types)        ***  1122d
char_just_x
  GLE_STEXT(gle_stext)              4098c  4101c
  GLE_TYPES(gle_types)        ***  1127d
char_just_y
  GLE_STEXT(gle_stext)              4098c  4102c
  GLE_TYPES(gle_types)        ***  1128d
char_rot_h
  DGL_VARS(dgl_vars)          ***  1075d
  LIB(dgl_lib)                    20720c 20805c 20809c 21155c 21157c 21438c
char_rot_w
  DGL_VARS(dgl_vars)          ***  1077d
  LIB(dgl_lib)                    20720c 20804c 20808c 21154c 21157c 21437c
char_size
  GLE_GEN(gle_gen)            ***  2078c
  GLE_HPGL(gle_hpgl_out)      ***  7531c
  GLE_RGL(gle_ras_out)        ***  8401c
  GLE_SMARK(gle_smark)              5131c  5147c
  GLE_TYPES(gle_types)        ***  1031d
char_sizes
  GLE_HPGL(gle_hpgl_out)      ***  7695c
  GLE_TYPES(gle_types)        ***  1106s
char_space
  GLE_STEXT(gle_stext)              4046c  4074c  4077c
  GLE_TYPES(gle_types)        ***  1123d
```

```
char_width
  GLE_SMARK(gle_smark)          ***   5125c
  GLE_STEXT(gle_stext)                4037c  4046c  4062c  4065c
  GLE_TYPES(gle_types)          ***   1121d
character
  CRT(crt)                            192c   615c
  IOLIB(general_2)                    840d   842c
  PRINTER(prtdvr)                     145d   231c   232c
  SYSDEVS(sysdevs)              ***    48d
chardel
  CRT(crt)                             99d   145d
  GCRT(crtb)                    ***    78d
  SYSDEVS(sysdevs)              ***    94d
charptr
  AMIGO(amigodvr)               ***   794d
  CONVERT(convert_text)                45d    67c
  CRT(crt)                      ***   406d
  CS80(cs80dvr)                 ***  1215d
  DC_DRV(intdc)                 ***   498c
  DISCHPIB(bkgnd)               ***    49d
  DISCHPIB(dischpib)                  196d   198d   202d   263c   287d   357d   428c   440d
  F9885(f9885dvr)                      82d   110d
  GCRT(crtb)                    ***   276d
  GLE_KNOB(gle_knob_in)               18045d 18110d
  INIT(isr)                            39d    46d    61d    79d
  INITLOAD(bootdammodule)              781d   786c   787c   788c
  INITLOAD(mini)                ***   392d
  INITLOAD(sysglobals)                 32d   205d
  KEYS(keys)                    ***    77d
  M68KSYS(ci)                   ***   311d
  MINIT(xminit)                 ***  1520c
  PRINTER(prtdvr)                      68d    80d
  SRMRAM(srmammodule)                  53d   100d
  SRM_DRV                       ***  2104c
  UCSD_AM(ucsd_am)              ***    44d
  UNITIO(uio)                          28d    30d    53d    74d
charvar
  DGL_C_OUT(dgl_config_out)           11051d 11069c
check
  CTABLE(ctr)                         374d   657c   670c   671c   672c   679c   680c   681c   690c   691c   693c   702c   703c   704c
                                      706c   713c   714c   715c   724c   725c   726c   730c   732c   741c   742c   743c   744c   749c
                                      751c   764c   765c   766c   767c   783c   784c   785c   786c   808c   815c
  ETU                                  43d   573c
  MOUSE(mouse)                        132d   161c   168c
check2
  LIFDAM(lifmodule)                   739d   753c   760c   780c
check_chr
  MOREFSYS(mfs)                        59d   126c   153c   185c   217c   257c   295c   328c   367c
check_osj
  AMIGO(amigodvr)                     548d   778c   784d
check_ioresult
  MIUI                                379d   494c   496c   498c   534c
check_protectcode_set_array
  SRMDAM(srmdammodule)                171d   490c  1061c
check_status
  MINIT(hminit)                       314d   372c   445c
checkassoc
  LIBRARIAN                          3308d  3480c  3501c
checkcard
  ETU                                 637d   642c   648c   701c   781c
checkctrl
  M68KSYS(ci)                         315d   349c   368c
checkentry
  LIFDAM(lifmodule)                   805d   877c
```

```
checkeprom
  ETU                                 653d   729c   875c  1176c
checkftitle
  LIFDAM(lifmodule)                   220d   670c   969c  1066c  1087c  1164c
checkinref
  LIBRARIAN                          2279d  2641c
checkloop
  HPHIL(hphil)                        204d   236c   296c
checkloopop
  HPHIL(hphil)                  ***   296c
  SYSDEVS(sysdevs)              ***   306d
checkname
  UCSD_DAM(ucsdmodule)                170d   172c   178c   210c   285c   303c   430c   446c
checkrev
  INIT(ldr)                          2438c  2456c
  INITLOAD(loader)                   1085d  1578d  1623c
  SEGMENTER(asm)                ***   196c
checkscode
  ETU                                 646d   659c   759c
checkspace
  ETU                                 487d   494c   513c  1001c
  LIFDAM(lifmodule)                   736d   810c   856c   893c
  UCSD_DAM(ucsdmodule)                190d   228c   233c
checktextbuf
  LIBRARIAN                          2241d  2662c  2669c  2676c  2682c  2689c  2710c
checkvolid
  LIFDAM(lifmodule)                   307d   393c   501c   563c   592c   670c   968c  1066c  1085c
chs
  SRMDAM(srmdammodule)                911d   942c
ci
  ETU                           ***    26d
  LIBRARIAN                     ***    27d
  M68KSYS                       ***  1211d
  M68KSYS(ci)                   ***    30d
ci_clr
  CS80(cs80)                          390d   394c
ci_cmd
  M68KSYS(ci)                          63d  1063c  1094c  1100c  1101c  1105c  1107c  1109c  1110c  1143c  1151c  1152c  1154c
ci_idle
  M68KSYS(ci)                          62d   140c   142c  1098c  1173c
ci_switch
  M68KSYS                       ***  1214c
  TAIL                          ***    38c
cic
  CS80(cs80)                          387d   393c   394c   395c
cindex
  MOUSE(mouse)                         65d    83c    95c    96c   103c   104c
cinfo
  INIT(misc)                    ***   197d
  LIFDAM(lifmodule)                   575c   619c
  SRMDAM(srmdammodule)                415c   417c   613c   616c   618c   620c   621c   625c   626c
  UCSD_DAM(ucsdmodule)                526c   558c
cinfo0
  CRT(crt)                            175d   692c
cinfo1
  CRT(crt)                      ***   175d
cinfo2
  CRT(crt)                      ***   175d
cinfo3
  CRT(crt)                      ***   175d
ciseoln
  M68KSYS(ci)                         409d   413c   482c   489c   502c   512c   523c
```

```
ck_display_init
   DGL_INQ(dgl_inq)                    6043c  6069c
   DGL_POLY(dgl_poly)                 20112c 20132c 20148c 20174c 20933c
   GEN(dgl_gen)                        3040d  3081d
   LIB(dgl_lib)                       20253c 20382c 20415c 20437c 20453c 20475c 20501c 20545c 20683c 20699c 20739c 20761c 20779c 20797c
                                      20824c 20859c 21030c
ck_for_graphics_board
   DGL_C_OUT(dgl_confg_out)           11192d 11270c
ck_locator_init
   GEN(dgl_gen)                        3042d  3089d
   LIB(dgl_lib)                       20271c 20860c 20894c 20969c 21073c
ck_system_init
   DGL_INQ(dgl_inq)                    6042c  6068c  6169c
   DGL_POLY(dgl_poly)                 20111c 20131c 20147c 20173c 20932c
   GEN(dgl_gen)                        3038d  3073d
   LIB(dgl_lib)                       20252c 20270c 20291c 20326c 20353c 20381c 20389c 20402c 20414c 20436c 20452c 20474c 20500c 20544c
                                      20682c 20698c 20738c 20760c 20778c 20796c 20823c 20858c 20893c 20968c 21029c 21072c 21095c 21204c
                                      21256c 21305c
ckind
   INIT(misc)                  ***      182d
   LIFDAM(lifmodule)           ***      610c
   SRMDAM(srmdammodule)        ***      604c
   UCSD_DAM(ucsdmodule)        ***      545c
cl
   MINIT(cs80ir)                        710d   718c   719c   720c   721c   722c
clastdate
   INIT(misc)                  ***      190d
   LIFDAM(lifmodule)                    573c   617c
   SRMDAM(srmdammodule)                 413c   436c   609c
   UCSD_DAM(ucsdmodule)                 524c   554c
clasttime
   INIT(misc)                  ***      191d
   LIFDAM(lifmodule)                    573c   617c
   SRMDAM(srmdammodule)                 414c   436c   609c
   UCSD_DAM(ucsdmodule)                 525c   555c
cleandir
   LIFDAM(lifmodule)                    359d   395c   972c   983c  1091c  1099c
cleanup
   LIFDAM(lifmodule)                    212d  1158c  1171c  1174c
   TAPEBKUP                            1229d  1274c
   UCSD_DAM(ucsdmodule)                 137d   214c   296c   305c
clear
   CRT(crt)                             315d   471c   472c   473c
   CTABLE                              1241d  1253c  1255c  1257c  1259c  1261c
   GCRT(crtb)                           224d   342c   343c   344c
   GLE_GEN(gle_gen)            ***     2096c
   GLE_HPGL(gle_hpgl_out)      ***     7529c
   GLE_RGL(gle_ras_out)                8398c  8480c
   GLE_TYPES(gle_types)        ***     1032d
   IOLIB(hpib_2)                       1780d  1838d
   LIBRARIAN                           3289d  3295c  3335c  3338c  3341c  3342c  3345c  3520c  3528c  3574c  3582c  3587c  3591c  3593c
                                       3599c  3602c  3608c  3610c
clear_and_escape
   F9885(f9885dvr)                       94d   132c   188c   202c   209c   225c   245c
clear_display
   LIB(dgl_lib)                        20055d 20539d
clear_hpib
   IOLIB(hpib_0)                       1711d  1733d
   IOLIB(hpib_2)                       1820c  1821c  1829c  1874c  1876c
clear_logs
   MINIT(cs80ir)                        563d   701d   724c
   MINIT(qminit)               ***     1173c
clear_logs_option_byte
   MINIT(qminit)                        897d  1066c  1077c  1173c
```

```
clear_serial
   IOLIB(serial_0)                     2217d  2294d
clear_unit
   AMIGO(amigodvr)                      490d   582c
   CS80(cs80dvr)                        999d  1230c  1366c
   F9885(f9885dvr)                       85d   269c
   INITLOAD(mini)                       395d   497c
   PRINTER(prtdvr)                      108d   298c
clearanychar
   KEYS(keys)                           349d   368c   516c   540c   545c   553c
clearchars
   GCRT(crtb)                           226d   231c   233c   234c   235c
cleariohook
   CTABLE                      ***     1436c
   HPHIL(hphil)                         389c   370c
   INIT                        ***     2544c
   INITLOAD(sysglobals)        ***      273d
   KERNEL(general_0)           ***     1244c
   M68KSYS(ci)                 ***      295c
clearline
   CRT(crt)                    ***      130d
   GCRT(crtb)                  ***       63d
   M68KSYS(ci)                           88d   560c   563c
   SYSDEVS(sysdevs)            ***       83d
clearlogs
   MINIT(cs80ir)                        714d   720c
clearscr
   CRT(crt)                    ***      473c
   ETU                         ***     1279c
   GCRT(crtb)                  ***      344c
   INIT(misc)                  ***      167d
   M68KSYS(ci)                 ***      558c
   TAPEBKUP                            1293c  1432c  1466c  1513c  1531c
clearscreen
   CRT(crt)                    ***      129d
   GCRT(crtb)                  ***       62d
   M68KSYS(ci)                           67d   183c   558c   878c  1102c  1197c
   SYSDEVS(sysdevs)            ***       82d
clearunit
   AMIGO(amigodvr)             ***      575c
   BUBBLES(bubble)             ***      107c
   CRT(crt)                    ***      416c
   CS80(cs80dvr)               ***     1364c
   CTABLE(ctr)                 ***      842c
   EPROMS(eproms)                        47c    78c
   F9885(f9885dvr)             ***      268c
   GCRT(crtb)                  ***      286c
   INITLOAD(mini)              ***      496c
   INITLOAD(sysglobals)        ***       66d
   KEYS(keys)                  ***       84c
   M68KSYS(ci)                           567c   963c
   MINIT(bminit)               ***       79c
   MIUI                        ***      497c
   PRINTER(prtdvr)             ***      297c
   SRMAM(srmammodule)                    188c   206c
   UNITIO(uio)                 ***       97c
clip_limits
   GLE_GEN(gle_gen)            ***     2120c
   GLE_HPGL(gle_hpgl_out)      ***     7532c
   GLE_RGL(gle_ras_out)        ***     8400c
   GLE_TYPES(gle_types)        ***     1033d
clip_limits_xmax
   GLE_SCLIP(gle_sclip)        ***     6018c
   GLE_TYPES(gle_types)        ***     1131d
```

```
clip_limits_xmin
  GLE_SCLIP(gle_sclip)          ***   6017c
  GLE_TYPES(gle_types)          ***   1130d
clip_limits_ymax
  GLE_SCLIP(gle_sclip)          ***   6020c
  GLE_TYPES(gle_types)          ***   1133d
clip_limits_ymin
  GLE_SCLIP(gle_sclip)          ***   6019c
  GLE_TYPES(gle_types)          ***   1132d
clipping_support
  DGL_HPGL(dgl_hpgl)            ***  17264c
  DGL_INQ(dgl_inq)              ***   6305c
  DGL_RAS(dgl_raster)           ***  17595c
  DGL_VARS(dgl_vars)            ***   1107d
cllclear
  CRT(crt)                      ***    523c
  GCRT(crtb)                    ***    395c
  KEYS(keys)                    ***    464c
  SYSDEVS(sysdevs)                      51d   542c   548c
clldisplay
  CRT(crt)                      ***    526c
  GCRT(crtb)                    ***    397c
  KEYS(keys)                            470c   471c
  PRINTER(prtdvr)               ***    188c
  SYSDEVS(sysdevs)              ***     51d
cllput
  CRT(crt)                      ***    509c
  GCRT(crtb)                    ***    381c
  SYSDEVS(sysdevs)                      51d   533c   552c   556c   560c   564c   573c
cllshftl
  CRT(crt)                      ***    511c
  GCRT(crtb)                    ***    383c
  SYSDEVS(sysdevs)                      51d   528c
cllshftr
  CRT(crt)                      ***    517c
  GCRT(crtb)                    ***    389c
  SYSDEVS(sysdevs)                      51d   572c
clock
  CLOCK                         ***    182d
  CLOCK(clock)                  ***     32d
clockdata
  CLOCK(clock)                  ***    163d
  SYSDEVS(sysdevs)                      200d   205d   453d
clockdate
  M68KSYS(ci)                           178d   255c   260c   267c   285c
clockdatetime
  M68KSYS(ci)                           179d   283c   287c
clockfunc
  CLOCK(clock)                  ***    163d
  SYSDEVS(sysdevs)                      198d   205d   453d
clockinit
  CLOCK                         ***     30d
clockiohook
  A804XDVR(a804xdvr)            ***    393c
  CLOCK(clock)                           53c    93c   103c   117c
  SYSDEVS(sysdevs)                      209d   460c   610c
clockiotype
  SYSDEVS(sysdevs)                      206d   209d
clockop
  A804XDVR(a804xdvr)            ***    239d
  SYSDEVS(sysdevs)                      199d   206d   455d
clockops
  A804XDVR(a804xdvr)                    239d   393c
```

```
clockreqhook
  CLOCK(clock)                  ***    177c
  SYSDEVS(sysdevs)                      208d   464c   467c   471c   475c   609c
clockreqtype
  SYSDEVS(sysdevs)                      205d   208d
clocktime
  M68KSYS(ci)                           177d   256c   271c   280c   286c
closeall
  ETU                                   414d  1131c  1138c  1266c
closed_share_code
  SRM_DRV(srm)                  ***    164d
closedir
  ETU                                   333d   415c
closedirectory
  ETU                           ***    341c
  INITLOAD(sysglobals)          ***    136d
  LIFDAM(lifmodule              ***   1154c
  SRMDAM(srmdammodule)                 1650c  1695c
  UCSD_DAM(ucsdmodule)          ***    646c
closedown
  M68KSYS(ci)                           328d   342c   358c   377c
closef
  LIFDAM(lifmodule                      990d  1130c
closefile
  ETU                           ***    405c
  INIT(misc)                    ***    434c
  INITLOAD(bootdammodule)       ***    721c
  INITLOAD(loader)              ***   1391c
  INITLOAD(sysglobals)          ***    135d
  LIFDAM(lifmodule              ***   1130c
  SRMDAM(srmdammodule)                 1658c  1759c
  UCSD_DAM(ucsdmodule)          ***    641c
closefiles
  INIT(ldr)                     ***   2490c
  INITLOAD(loader)                     1090d  1385d  1561c  1647c
  LIBRARIAN                            3007c  3687c  3703c
  SEGMENTER(asm)                ***    214c
closein
  LIBRARIAN                            2957d  3006c  3031c  3079c  3105c  3684c  3687c  3702c
closeinfile
  ETU                                   397d   415c
closeit
  UCSD_DAM(ucsdmodule)                  404d   641c
closeout
  LIBRARIAN                            3103d  3637c
closepack
  SRMDAM(srmdammodule)                  340c   927c
  SRM_DRV(srm)                         1033d  1681d
closest_error
  DGL_RAS(dgl_raster)                 17084d 17107c 17158c 17160c
closest_index
  DGL_RAS(dgl_raster)                 17086d 17106c 17161c 17164c
closetype
  INIT(fs)                              744d   767d
clr_inhibit
  DGL_HPGL(dgl_hpgl)                  17251d 17290c
  DGL_RAS(dgl_raster)                 17555d 17609c
clr_with_1m
  GLE_RGL(gle_ras_out)                 8081d  8228c
clsize
  INIT(misc)                    ***    185s
  LIFDAM(lifmodule                      570c   612c   613c
  SRMDAM(srmdammodule)                  407c   606c
  UCSD_DAM(ucsdmodule)                  518c   548c
```

```
cmap_address
  GLE_RGL(gle_ras_out)          8067d  8343c
cmap_def
  GLE_RGL(gle_ras_out)          8015d  8021d
cmd
  A804XDVR(a804xdvr)             40d    42d  138d  139c  146d  147c  169d  171c  181d  197c  201c  206c  239d  241c
  BAT(bat)                       67d    76c
  CLOCK(clock)                  163d   165c
  CS80(cs80)                    355d   401d  408d  411c  416d  424d  428c  556d  576d  588c
  M68KSYS                ***     28d
  SYSDEVS(sysdevs)               40d   185d  205d  206d  371d  380d  405d  428d  430c  437d  438c  445d  453d  455d
cmd1
  CS80(cs80)             ***    240d
cmd100
  CS80(cs80)             ***    265d
cmd101
  CS80(cs80)             ***    265d
cmd102
  CS80(cs80)             ***    265d
cmd103
  CS80(cs80)             ***    265d
cmd104
  CS80(cs80)             ***    266d
cmd105
  CS80(cs80)             ***    266d
cmd106
  CS80(cs80)             ***    266d
cmd107
  CS80(cs80)             ***    266d
cmd108
  CS80(cs80)             ***    267d
cmd109
  CS80(cs80)             ***    267d
cmd11
  CS80(cs80)             ***    242d
cmd110
  CS80(cs80)             ***    267d
cmd111
  CS80(cs80)             ***    267d
cmd112
  CS80(cs80)             ***    268d
cmd113
  CS80(cs80)             ***    268d
cmd114
  CS80(cs80)             ***    268d
cmd115
  CS80(cs80)             ***    268d
cmd116
  CS80(cs80)             ***    269d
cmd117
  CS80(cs80)             ***    269d
cmd118
  CS80(cs80)             ***    269d
cmd119
  CS80(cs80)             ***    269d
cmd12
  CS80(cs80)             ***    243d
cmd120
  CS80(cs80)             ***    270d
cmd121
  CS80(cs80)             ***    270d
cmd122
  CS80(cs80)             ***    270d
cmd123
  CS80(cs80)             ***    270d
cmd124
  CS80(cs80)             ***    271d
cmd125
  CS80(cs80)             ***    271d
cmd126
  CS80(cs80)             ***    271d
cmd127
  CS80(cs80)             ***    271d
cmd128
  CS80(cs80)             ***    272d
cmd19
  CS80(cs80)             ***    244d
cmd20
  CS80(cs80)             ***    245d
cmd21
  CS80(cs80)             ***    245d
cmd22
  CS80(cs80)             ***    245d
cmd23
  CS80(cs80)             ***    245d
cmd25
  CS80(cs80)             ***    246d
cmd26
  CS80(cs80)             ***    246d
cmd27
  CS80(cs80)             ***    246d
cmd28
  CS80(cs80)             ***    247d
cmd29
  CS80(cs80)             ***    247d
cmd3
  CS80(cs80)             ***    240d
cmd30
  CS80(cs80)             ***    247d
cmd31
  CS80(cs80)             ***    247d
cmd5
  CS80(cs80)             ***    241d
cmd54
  CS80(cs80)             ***    253d
cmd63
  CS80(cs80)             ***    255d
cmd7
  CS80(cs80)             ***    241d
cmd75
  CS80(cs80)             ***    258d
cmd76
  CS80(cs80)             ***    259d
cmd77
  CS80(cs80)             ***    259d
cmd78
  CS80(cs80)             ***    259d
cmd79
  CS80(cs80)             ***    259d
cmd80
  CS80(cs80)             ***    260d
cmd81
  CS80(cs80)             ***    260d
cmd82
  CS80(cs80)             ***    260d
cmd83
  CS80(cs80)             ***    260d
```

cmd84															
CS80(cs80)	***	261d													
cmd85															
CS80(cs80)	***	261d													
cmd86															
CS80(cs80)	***	261d													
cmd87															
CS80(cs80)	***	261d													
cmd88															
CS80(cs80)	***	262d													
cmd89															
CS80(cs80)	***	262d													
cmd9															
CS80(cs80)	***	242d													
cmd90															
CS80(cs80)	***	262d													
cmd91															
CS80(cs80)	***	262d													
cmd92															
CS80(cs80)	***	263d													
cmd93															
CS80(cs80)	***	263d													
cmd94															
CS80(cs80)	***	263d													
cmd95															
CS80(cs80)	***	263d													
cmd96															
CS80(cs80)	***	264d													
cmd97															
CS80(cs80)	***	264d													
cmd98															
CS80(cs80)	***	264d													
cmd99															
CS80(cs80)	***	264d													
cmd_buffer															
AMIGO(csamigo)		144d	240d	265c	269c										
cmd_read_1															
A804XDVR(a804xdvr)		42d	146d	175c	209s	213c	221c								
HPHIL(hphil)		140c	149c	167c	216c	233c									
cmd_type															
CS80(cs80)		239d	355d	389d	393c	401d	408d	416d	423d	424d	427c	439c	451c	504d	505d
		526d	542d	543d	556d	568d	569d	570d	571d	573d	574d	576d	579c	580c	
CS80(cs80dvr)	***	1097d													
MINIT(cs80ir)		578d	579d	585c	586c	603d	633d	634d	639c	661d	688d	713d	745d	781d	830d
		831d	859d												
TAPEBKUP(cs80tb*)		43d	44d	50c	51c	65c	66c	85d	113d	115d	116d	145d	168d	192d	193d
		195d	221d	222d	224d	225d	227d	230d	267d						
cmdcharhook															
M68KSYS(ci)		61d	1064c	1100c	1174c	1175c	1176c								
cmdcold_load_read															
CS80(cs80)	***	242d													
cmdcopy_data															
CS80(cs80)	***	242d													
TAPEBKUP(cs80tb*)	***	238c													
cmddescribe															
CS80(cs80)		253d	492c												
cmdinit_diagnostic															
CS80(cs80)	***	252d													
MINIT(cs80ir)	***	640c													
cmdinit_media															
CS80(cs80)	***	253d													
MINIT(cs80ir)	***	667c													
TAPEBKUP(cs80tb*)	***	91c													

cmdinit_util_nem					
CS80(cs80)	***	252d			
MINIT(cs80ir)		693c	719c	793c	
cmdinit_util_rem					
CS80(cs80)	***	252d			
MINIT(cs80ir)	***	613c			
cmdinit_util_sem					
CS80(cs80)	***	252d			
MINIT(cs80ir)		751c	792c		
TAPEBKUP(cs80tb*)	***	275c			
cmdlocate_and_read					
CS80(cs80)	***	240d			
CS80(cs80dvr)	***	1108c			
cmdlocate_and_ver					
CS80(cs80)	***	241d			
TAPEBKUP(cs80tb*)	***	202c			
cmdlocate_and_wrt					
CS80(cs80)	***	240d			
CS80(cs80dvr)	***	1109c			
cmdno_op					
CS80(cs80)		253d	547c	581c	585c
MINIT(cs80ir)	***	836c			
TAPEBKUP(cs80tb*)		123c	199c	237c	
cmdprocedure					
M68KSYS(ci)		51d	52d		
cmdprocptr					
M68KSYS(ci)		52d	61d		
cmdreg					
A804XDVR(a804xdvr)		63d	139c		
cmdrelease					
CS80(cs80)		243d	479c		
cmdrelease_denied					
CS80(cs80)	***	243d			
cmdrequest_status					
CS80(cs80)		243d	464c		
cmdset_address_1v					
CS80(cs80)		244d	582c		
TAPEBKUP(cs80tb*)		124c	241c	246c	
cmdset_address_3v					
CS80(cs80)	***	244d			
MINIT(cs80ir)	***	837c			
cmdset_block_disp					
CS80(cs80)	***	244d			
cmdset_burst_abt					
CS80(cs80)	***	255d			
cmdset_burst_lbo					
CS80(cs80)	***	255d			
cmdset_length					
CS80(cs80)		246d	586c		
TAPEBKUP(cs80tb*)		200c	235c		
cmdset_options					
CS80(cs80)		254d	530c		
cmdset_release					
CS80(cs80)		254d	512c		
cmdset_retadd_mode					
CS80(cs80)	***	258d			
TAPEBKUP(cs80tb*)	***	121c			
cmdset_retry_time					
CS80(cs80)	***	254d			
cmdset_rps					
CS80(cs80)	***	254d			
cmdset_status_mask					
CS80(cs80)		255d	548c		

```
cmdset_unit_0
  CS80(cs80)                        248d   393c   427c   439c   579c
  MINIT(cs80ir)                     585c   639c
  TAPEBKUP(cs80tbr)                  50c    65c
cmdset_unit_1
  CS80(cs80)                 ***    248d
cmdset_unit_10
  CS80(cs80)                 ***    250d
cmdset_unit_11
  CS80(cs80)                 ***    250d
cmdset_unit_12
  CS80(cs80)                 ***    251d
cmdset_unit_13
  CS80(cs80)                 ***    251d
cmdset_unit_14
  CS80(cs80)                 ***    251d
cmdset_unit_15
  CS80(cs80)                        251d   511c
  TAPEBKUP(cs80tbr)          ***    234c
cmdset_unit_2
  CS80(cs80)                 ***    248d
cmdset_unit_3
  CS80(cs80)                 ***    248d
cmdset_unit_4
  CS80(cs80)                 ***    249d
cmdset_unit_5
  CS80(cs80)                 ***    249d
cmdset_unit_6
  CS80(cs80)                 ***    249d
cmdset_unit_7
  CS80(cs80)                 ***    249d
cmdset_unit_8
  CS80(cs80)                 ***    250d
cmdset_unit_9
  CS80(cs80)                 ***    250d
cmdset_vol_0
  CS80(cs80)                        256d   451c   580c
  MINIT(cs80ir)              ***    586c
  TAPEBKUP(cs80tbr)                  51c    66c
cmdset_vol_1
  CS80(cs80)                 ***    256d
cmdset_vol_2
  CS80(cs80)                 ***    256d
cmdset_vol_3
  CS80(cs80)                 ***    256d
cmdset_vol_4
  CS80(cs80)                 ***    257d
cmdset_vol_5
  CS80(cs80)                 ***    257d
cmdset_vol_6
  CS80(cs80)                 ***    257d
cmdset_vol_7
  CS80(cs80)                 ***    257d
cmdspare_block
  CS80(cs80)                 ***    241d
  MINIT(cs80ir)              ***    864c
cmdunload
  CS80(cs80)                 ***    258d
  TAPEBKUP(cs80tbr)          ***    172c
cmdwrite_file_mark
  CS80(cs80)                 ***    258d
  TAPEBKUP(cs80tbr)          ***    149c
cmmd40
  CLOCK(clock)                       88d    97c
```

```
cmmd41
  CLOCK(clock)                      129d   147c
cmmdb7
  CLOCK(clock)                       87d    96c
cmmdf2
  CLOCK(clock)               ***    135s
cmmdf3
  CLOCK(clock)                      134d   155c
cmmdf4
  CLOCK(clock)                      133d   154c
cmmdf5
  CLOCK(clock)                      132d   153c
cmmdf6
  CLOCK(clock)                      131d   149c
cmmdf7
  CLOCK(clock)                      130d   148c
cname
  ETU                        ***    376c
  INIT(misc)                 ***    180d
  LIFDAM(lifmodule)                 511c   566c   609c
  M68KSYS(ci)                ***    822c
  MIUI                              518c   519c
  SRMDAM(srmdammodule)              395c   404c   432c   452c   602c   997c  1004c
  UCSD_DAM(ucsdmodule)              485c   487c   515c   544c
cnormal
  INIT(fs)                   ***    744d
cnt
  DC_DRV(extdc)              ***    203d
  DGL_C_IN(dgl_confg_in)           12043d 12049c 12089d 12097c
  DGL_C_OUT(dgl_confg_out)         11158d 11279c 11406d 11419c
  DGL_HPGL(dgl_hpgl)               17028d 17052c 17072d 17077c 17098c 17100c 17106c 17108c 17114c 17116c
  GLE_HPGL(gle_hpgl_out)            7165d  7182c  7183c  7184c  7185c  7186c  7187c  7188c  7222d  7242c  7243c  7244c  7245c  7246c
                                    7247c  7248c
  GLE_HPGLI(gle_hpgl_in)           18096d 18108c 18109c 18110c 18111c 18112c 18113c 18114c 18134d 18148c 18149c 18150c 18151c 18152c
                                   18153c 18154c 18226d 18254c 18255c 18265c 18291d 18301c 18302c 18303c 18304c 18305c
  GLE_HPIB(gle_hpib_io)            10486d 10493c
  GLE_RGL(gle_ras_out)       ***    8377d
  LIB(dgl_lib)                     21201d 21217c 21218c 21253d 21264c
cntrl
  INIT(misc)                 ***    174d
  KEYS(keys)                 ***    527c
cntrlpausekey
  KEYS(keys)                         64d   521c
code
  LIBRARIAN                         298d  1198c  1210c  1216c
  SEGMENTER(asm)                     96d    98c
  SEGMENTER(segmenter)       ***     38d
code_space
  SEGMENTER(asm)                    292d   297c   304c   306c   311c
codecount
  LIBRARIAN                         310d   480c   485c  1297c
coded_addr
  AMIGO(csamigo)                    339d   347c   351c   352c   356c   357c   387c   400c
codefid
  M68KSYS(ci)                        47d   631c   642c  1025c
codefile
  INIT(misc)                        722c   723c
  INITLOAD                   ***   1708c
  INITLOAD(loader)           ***   1141c
  INITLOAD(sysglobals)       ***     48d
  LIBRARIAN                        2893c  3082c  3148c
  M68KSYS(ci)                       103c   665c   996c
codeindex
  LIBRARIAN                         311d   481c   484c  1297c
```

```
codegtf
  SEGMENTER(asm)                          141d   148c
codetype
  SRMRAM(srmdammodule)          ***       71d
coi
  MOUSE(mouse)                            54d   115c   118c
color_data_def
  GLE_RGL(gle_ras_out)                  8314d  8315d
color_data_ptr
  GLE_RGL(gle_ras_out)                  8323d  8328c  8333c  8334c  8335c
color_data_ptr_def
  GLE_RGL(gle_ras_out)                  8315d  8323d
color_list
  DGL_RAS(dgl_raster)                  17203d 17215c 17216c 17217c 17218c 17250d 17268c 17269c 17270c 17281c
  GLE_RGL(gle_ras_out)                  8247d  8279c
color_map
  GLE_RGL(gle_ras_out)                  8318d  8343c  8346c
color_map_def
  GLE_RGL(gle_ras_out)                  8312d  8313d  8319d
color_map_ptr
  GLE_RGL(gle_ras_out)                  8313d  8318d
color_map_support
  DGL_POLY(dgl_poly)                   20068c 20070c
  DGL_RAS(dgl_raster)                  17211c 17228c 17597c
  GLE_HPGL(gle_hpgl_out)        ***     7700c
  GLE_RGL(gle_ras_out)          ***     8341c
  GLE_TYPES(gle_types)          ***     1095d
color_table_def
  DGL_VARS(dgl_vars)                    1029d  1030d  1342d
color_table_def_space
  DGL_RAS(dgl_raster)           ***    17623c
  DGL_VARS(dgl_vars)            ***     1342d
color_table_ptr
  DGL_INQ(dgl_inq)              ***     6048c
  DGL_POLY(dgl_poly)            ***    20074c
  DGL_RAS(dgl_raster)                  17177c 17191c 17221c 17271c 17623c
  DGL_VARS(dgl_vars)            ***     1118d
color_table_ptr_def
  DGL_RAS(dgl_raster)           ***    17109c
  DGL_VARS(dgl_vars)                    1030d  1118d
color_table_size
  DGL_HPGL(dgl_hpgl)            ***    17277c
  DGL_INQ(dgl_inq)                      6046c  6047c  6420c  6430c
  DGL_POLY(dgl_poly)            ***    20118c
  DGL_RAS(dgl_raster)           ***    17622c
  DGL_VARS(dgl_vars)            ***     1087d
  LIB(dgl_lib)                         20418c 20459c 21454c
colorinfo
  CRT(crt)                                175d   692c
column
  CRT(crt)                                182d   190c   192c   194c   196c   197c   198c
comma
  LIBRARIAN                               464d   502c   607c   665c   669c   675c   679c   689c   692c   698c   703c   722c   852c   856c
                                          915c   935c   936c   940c   945c   954c   977c   982c   994c  1016c  1037c  1068c  1090c  1098c
                                         1108c  1113c  1115c  1126c  1128c  1130c  1134c  1136c
command
  AMIGO(csamigo)                          144d   240d   265c
  CS80(cs80dvr)                          1097d  1108c  1109c  1110c
  F9885(f9885dvr)               ***       108d
  GLE_HPIB(gle_hpib_io)                 10109d 10114c
  HPHIL(hphil)                             51d   322c
  INITLOAD(bootdammodule)                 780d   786c   787c
  IOLIB(hpib_1)                           394d   433d   438c
  M68KSYS(ci)                             969d  1187c
```

```
command_sec
  CS80(cs80)                              276d   412c   429c   516c   532c   550c   589c
  MINIT(cs80ir)                           616c   643c   670c   695c   722c   754c   800c   841c   866c
  TAPEBKUP(cs80tbr)                        67c    94c   127c   150c   173c   203c   249c   278c
command_table
  AMIGO(csamigo)                          250d   265c
command_table_type
  AMIGO(csamigo)                          248d   250d
command_type
  AMIGO(amigodvr)               ***       714d
  AMIGO(csamigo)                           46d   144d   240d   248d
commandchar
  LIBRARIAN                                52d   278c  1452c  1453c  1454c  3356c  3357c  3407c  3408c  3506c  3621c  3622c  3695c
commandinprogress
  GLE_KNOB(gle_knob_in)         ***    18114d
  KEYS(keys)                               75d   108c   116c
commandlevel
  ETU                                      32d  1282c
  TAPEBKUP                               1491d  1538c
compileandedit
  M68KSYS(ci)                            1005d  1022c  1113c
compiler
  M68KSYS(ci)                              39d   718c  1010c
complement_mode
  DGL_VARS(dgl_vars)            ***     1204d
  LIB(dgl_lib)                  ***    20992c
complement_support
  GLE_HPGL(gle_hpgl_out)        ***     7697c
  GLE_TYPES(gle_types)          ***     1092d
  LIB(dgl_lib)                         20461c 21283c
compliment
  GLE_RGL(gle_ras_out)          ***     8080d
compressgvr
  LIBRARIAN                              1633d  1910c  1950c  2312c
computerid
  SRM_DRV(srm)                  ***     1405d
cond
  LIBRARIAN                               431d  1034c  1035c  1042c  1054c  1055c  1056c
condcode
  LIBRARIAN                               559d  1035c  1042c  1056c
condcodetype
  LIBRARIAN                               551d   559d
configure
  CS80(cs80dsr)                           606d   761c   771d
  CS80(cs80dvr)                 ***      1006c
  HPHIL(hphil)                            222d   279c   297c   317c   372c   379c
  TAPEBKUP(cs80tbdvr)                     389c   651c   884c
configure_gle
  DGL_C_OUT(dgl_confg_out)              11007d 11502d
  DGL_KNOB(dgl_knob)            ***     18077c
  LIB(jgl_lib)                          21221c 21270c
configure_input_gle
  DGL_C_IN(dgl_confg_in)                12007d 12131d
  LIB(jgl_lib)                  ***     21322c
configureop
  HPHIL(hphil)                  ***       297c
  MOUSE(mouse)                  ***       195c
  SYSDEVS(sysdevs)              ***       306d
confirm_preeoi
  MINIT(hminit)                           306d   473c
confirm_timeout_and_reset_card
  DISCHPIB(dischpib)                      219d   266c   307c   325c   379c   405c   431c   491c   545c
constpassarray
  SRMDAM(srmdammodule)                     53d  1620c
```

```
consts
  LIBRARIAN                          298d  1145c  1194c  1242c  1461c  1463c
cont
  LIBRARIAN                          751d   765c   771c
cont_linestyles
  GLE_HPGL(gle_hpgl_out)            7623c  7630c  7639c  7655c  7674c
  GLE_TYPES(gle_types)       ***    1103d
contiguous_first_extent
  SRM_DRV(srm)                       499d  1767c
control
  DGL_C_OUT(dgl_confg_out)         11160d 11275c 11318c 11330c 11354c 11409d 11416c 11447c 11487c
  DGL_HPGL(dgl_hpgl)               17006d 17220d 17289c
  DGL_RAS(dgl_raster)              17006d 17544d 17608c
  EDRIVER(edriver)                    67d   176c   210c   211c   218c   233c   247c   255c   259c
  ETU                                 55d    78d   219d   352d
  KEYS(keys)                         214c   219c   430c   444c   445c   478c   479c   485c   507c   521c   527c
  LIB(dgl_lib)                     20023d 20026d 21195d 21219c 21246d 21268c
  NONUSKBD1(non_us_kbd1)     ***      54c
  SYSDEVS(sysdevs)           ***     266d
control_bfd
  DC_DRV(extdc)              ***     209d
  DC_DRV(intdc)                      267c   382c
control_cutter
  DGL_HPGL(dgl_hpgl)               17080d 17126c
control_def
  DGL_HPGL(dgl_hpgl)               17245d 17257d
  DGL_RAS(dgl_raster)              17549d 17580d
control_space
  DGL_C_OUT(dgl_confg_out)         11091c 11101c 11109c 11123c 11133c 11256c
  DGL_RAS(dgl_raster)        ***   17389c
  GLE_TYPES(gle_types)       ***    1278d
controller_fault
  CS80(cs80)                 ***     151d
  CS80(cs80dsr)              ***     668c
  TAPEBKUP(cs80tbdvr)                548c   780c
controllercopy
  TAPEBKUP(cs80tbdvr)                894d   907c
convert_dtow
  DGL_HPGLI(dgl_hpgli)             18030c 18102c
  DGL_KNOB(dgl_knob)         ***   18145c
  GEN(dgl_gen)                      3035d  3328d
convert_hsl_to_rgb
  DGL_RAS(dgl_raster)              17210c 17267c
  GEN(dgl_gen)                      3048d  3166d
convert_intwtod
  DGL_POLY(dgl_poly)         ***   20976c
  GEN(dgl_gen)                      3026d  3274d
convert_itod
  DGL_HPGLI(dgl_hpgli)             18029c 18080c 18101c
  DGL_KNOB(dgl_knob)         ***   18142c
  GEN(dgl_gen)                      3032d  3313d
convert_rgb_to_hsl
  DGL_INQ(dgl_inq)           ***    6056c
  DGL_RAS(dgl_raster)              17105c 17111c
  GEN(dgl_gen)                      3049d  3190d
convert_text
  CONVERT(convert_text)      ***      25d
convert_wtod
  DGL_POLY(dgl_poly)               20973c 20978c
  GEN(dgl_gen)                      3029d  3302d  3377c  3420c  3421c
  LIB(dgl_lib)               ***   20874c
convert_wtodmm
  LIB(dgl_lib)                     20091d 20246d
```

```
convert_wtolmm
  LIB(dgl_lib)                     20092d 20263d
cookkbd
  HPHIL(hphil)               ***      60d
copy_attempt_completed
  TAPEBKUP                          1224d  1352c  1358c  1364c  1366c
copy_start_address
  TAPEBKUP(cs80tbdvr)        ***     937c
  TAPEBKUP(cs80tbr)                   28d   255d   295c
copybuffer
  LIBRARIAN                         2833d  2850c  2867c  2868c  2873c
copydata
  TAPEBKUP(cs80tbr)                  225d   238c
copyfile
  LIBRARIAN                         3026d  3623c
copymodule
  LIBRARIAN                         2829d  3030c  3059c  3069c  3375c  3391c  3429c
copyon
  LIBRARIAN                         2951d  3628c
copypack
  SRMDAM(srmdammodule)       ***    1426c
  SRM_DRV(srm)                      1035d  1702d
copyright
  LIBRARIAN                           42d  1964c  2977c  3210c  3580c
copyright1
  MIUI                                47d    92c
copyright2
  MIUI                                48d    93c
copytext
  LIBRARIAN                         2125d  2994c
copytimeout
  SRM_DRV(srm)                       996d  1224c  1249c  1420c
corrupt_share_code
  SRMDAM(srmdammodule)       ***     625c
  SRM_DRV(srm)               ***     165d
cos_angle
  DGL_POLY(dgl_poly)               20881d 20901c 20903c
cosx_table
  GLE_STEXT(gle_stext)       ***    4038c
  GLE_TYPES(gle_types)       ***    1166d
cosy_table
  GLE_STEXT(gle_stext)       ***    4043c
  GLE_TYPES(gle_types)       ***    1167d
count
  A804XDVR(a804xdvr)                 126d   128c   267c   269c   274c   276c   296c   298c   302c
  ASCII(asciimodule)                 177d   202c   203c   204c   213d   222c   223c   224c   234d   248c   249c   251c   252c
  DC_DRV(intdc)                      313d   316c   317c   331d   334c   335c
  GLE_HPGL(gle_hpgl_out)            7066d  7074c  7076c  7083d  7088c  7089c
  GLE_HPGLI(gle_hpgl_in)           18039d 18047c 18049c 18056d 18061c 18062c
  GLE_RGL(gle_ras_out)       ***    8066d
  GLE_UTLS(gle_utls)               21008d 21069d 21101c
  IOLIB(general_2)                   821d  1070d  1081c
  SRM_DRV(srm)                      1299d  1307c  1409d  1439c  1440c  1449c  1450c  1452c  1454c  1460c
  SYSDEVS(sysdevs)                   217d   478c
  UCSD_AM(ucsd_am)                    45d   152c   214c   257c
countcode
  INIT(ldr)                  ***    2466c
  INITLOAD(loader)                  1087d  1368d  1629c
  SEGMENTER(asm)             ***     203c
counter
  EPROMS(eproms)                      40d    57c    58c    60c    64c    68c    70c    82c    91c    92c
  GLE_HPIB(gle_hpib_io)            10019d 10249c 10337c
  IOLIB(hpib_1)                      423d   571c   659c
```

```
counts
  MOUSE(mouse)                          159c   160c
  SYSDEVS(sysdevs)          ***         320d
cp
  INITLOAD(loader)          ***         916d
  LIBRARIAN                            1397c  1400c
cpen
  GLE_RGL(gle_ras_out)      ***        8053d
cpsize
  INIT(misc)                ***         183s
  LIFDAM(lifmodule)                     508c   509c   513c   521c   569c   570c   611c   613c
  M68KSYS(ci)               ***         824c
  MIUI                      ***         521c
  SRMDAM(srmdammodule)                  406c   605c
  UCSD_DAM(ucsdmodule)                  488c   490c   517c   547c   548c
cptr
  EDRIVER(edriver)                       91d   199c   215c   217c   220c   230c   232c   235c
  EPROMS(eproms)                         36d    61c    68c
cpvol
  ETU                                    62d   230c   234c   243c   247c   254c   361c   362c   363c   893c   896c   897c
cpx
  DGL_VARS(dgl_vars)        ***        1251d
  GLE_SMARK(gle_smark)                  5096d  5121c  5136c  5158c  5172c
  LIB(dgl_lib)                         20220d 20228c 20240c 20570c 20576c 20600c 20606c 21176c 21426c
cpy
  DGL_VARS(dgl_vars)        ***        1250d
  GLE_SMARK(gle_smark)                  5097d  5122c  5137c  5159c  5173c
  LIB(dgl_lib)                         20221d 20229c 20241c 20571c 20577c 20601c 20607c 21177c 21427c
cpymsg
  INITLOAD                  ***        1792c
  INITLOAD(loader)          ***        1619c
cr
  INIT(misc)                ***         188d
  KEYS(keys)                            447c   524c
create_new
  UCSD_DAM(ucsdmodule)                  182d   635c
create_temp_unitable
  CTABLE                    ***        1492c
  CTABLE(ctr)                           302d   821d
createdefs
  INITLOAD(loader)                     1419d  1560c
createfile
  INIT(misc)                ***         450c
  INITLOAD(sysglobals)      ***         135d
  LIFDAM(lifmodule          ***        1128c
  SRMDAM(srmdammodule)                 1655c  1732c
  UCSD_DAM(ucsdmodule)      ***         635c
createlinkpack
  SRMDAM(srmdammodule)                 1202c  1235c  1391c
  SRM_DRV                   ***        1780d
  SRM_DRV(srm)              ***        1055d
createpack
  SRMDAM(srmdammodule)                 1007c  1086c  1123c  1129c  1134c  1146c  1420c
  SRM_DRV(srm)                         1041d  1728d
createsysname
  INITLOAD                             1734d  1785c
creation_date
  SRMDAM(srmdammodule)                  435c   608c  1465c
  SRM_DRV(srm)                          282d   738d
cross_unit
  CS80(cs80)                            149d   294d
  TAPEBKUP(cs80tbarr)                   775c   867c
```

```
crt
  CRT                       ***         731d
  CRT(crt)                  ***          32d
  GLE_RGL(gle_ras_out)      ***        8390c
  GLE_TYPES(gle_types)      ***        1272d
crt_config
  C_HOOK                                39d   143c
  DGL_TOOLS(dgl_tools)                 20017d 20034c
crt_reg
  C_HOOK                                53d   144c
  DGL_TOOLS(dgl_tools)                 20029d 20046c
crt_reg_def
  C_HOOK                                43d    53d
  DGL_TOOLS(dgl_tools)                 20021d 20029d
crt_tm_name
  CTABLE(ctr)                           358d   645c
crt_wtb
  KERNEL(general_0                      795d  1234c
crtb
  GCRT                      ***         580d
  GCRT(crtb)                ***          32d
crtcmdwrd
  CRT(crt)                              51d   270d   280d   664d
  GCRT(crtb)                            98d   197d
crtcolumn
  MIUI                                 224d   246c   248c   253c   255c   260c   261c   269c
  TAPEBKUP                            1111d  1133c  1135c  1140c  1142c  1147c  1148c  1156c
crtcommand
  CRT(crt)                              263d   275c   276c   287c   288c   681c   682c
crtcon
  CRT(crt)                               93d   139d
  GCRT(crtb)                ***          72d
  SYSDEVS(sysdevs)          ***          92d
crtconfigreg
  CRT(crt)                  ***         687c
  INITLOAD(sysglobals)      ***         284d
crtconsttype
  CRT(crt)                               94d   139d
  GCRT(crtb)                ***          72d
  SYSDEVS(sysdevs)                       66d    92d
crtcontroladdr
  CRT(crt)                               89d   135d   676c   677c
  GCRT(crtb)                ***          68d
  SYSDEVS(sysdevs)          ***          89d
crtcreg
  CRT(crt)                  ***         120d
  GCRT(crtb)                ***          53d
  SYSDEVS(sysdevs)                       76d   107d
crtctrl
  CRT(crt)                  ***         120d
  GCRT(crtb)                ***          53d
  SYSDEVS(sysdevs)          ***         107d
crtdebug
  CRT(crt)                              544d   704c
  GCRT(crtb)                            413d   514c
crtfreg
  CRT(crt)                  ***         110d
  GCRT(crtb)                ***          43d
  SYSDEVS(sysdevs)                       68d   105d
crtidreg
  CRT(crt)                              173d   688c   692c
```

```
crtinfo
  CRT(crt)                              132d   185c   666c   720c
  GCRT(crtb)                             65d   502c
  M68KSYS(ci)                           462c   493c   505c  1089c
  MINIT(mminit)              ***        158c
  SYSDEVS(sysdevs)           ***        108d
crtinithook
  CRT(crt)                   ***        705c
  GCRT(crtb)                 ***        517c
  SYSDEVS(sysdevs)                      125d   627c
crtio
  CRT(crt)                              428c   442c   443c
  GCRT(crtb)                            298c   312c   313c
  INIT(initunits)                      2147d  2244c  2250c
  KEYS(keys)                 ***         94s
  SYSDEVS(sysdevs)                      376d   484d
crtiohook
  CRT(crt)                   ***        702c
  GCRT(crtb)                 ***        516c
  SYSDEVS(sysdevs)                      119d   486c   628c
crtirec
  CRT(crt)                               86d   132d
  GCRT(crtb)                 ***         65d
  SYSDEVS(sysdevs)                       87d   108d
crtkinds
  SYSDEVS(sysdevs)                      113d   130d
crtline
  MIUI                                  224d   246c   248c   255c
  TAPEBKUP                             1111d  1133c  1135c  1142c
crtllhook
  CRT(crt)                   ***        703c
  DGL_C_OUT(dgl_confg_out)             11068c 11185c 11186c
  GCRT(crtb)                 ***        515c
  KEYS(keys)                            464c   470c   471c
  PRINTER(prtdvr)            ***        188c
  SYSDEVS(sysdevs)                      126d   528c   533c   542c   548c   552c   556c   560c   564c   572c   573c   582c   629c
crtllops
  CRT(crt)                   ***        502d
  GCRT(crtb)                 ***        370d
  SYSDEVS(sysdevs)                       51d    52d   378d   487d
crtlltype
  DGL_C_OUT(dgl_confg_out)   ***      11023d
  SYSDEVS(sysdevs)                       52d   126d
crtmemaddr
  CRT(crt)                               88d   134d   668c
  GCRT(crtb)                 ***         67d
  MINIT(mminit)              ***        159c
  SYSDEVS(sysdevs)           ***         89d
crtptr
  MINIT(mminit)                         149d   159c   162c
crtregtype
  CRT(crt)                               50d   263d
  GCRT(crtb)                 ***         97d
crtscreen
  CRT(crt)                               57d    58d
crttype
  CRT(crt)                   ***        119d
  GCRT(crtb)                 ***         52d
  SYSDEVS(sysdevs)           ***        106d
crtword
  CRT(crt)                   ***         57d
  GCRT(crtb)                 ***         93d
  SYSDEVS(sysdevs)           ***         47d
```

```
crunch
  INITLOAD(sysglobals)       ***        136d
  LIFDAM(lifmodule)          ***       1151c
  SRMDAM(srmdammodule)       ***       1794c
  UCSD_DAM(ucsdmodule)       ***        647c
crunchit
  UCSD_DAM(ucsdmodule)                  567d   647c
crunchv
  LIFDAM(lifmodule)                     384d  1151c
cs80
  CS80(cs80)                 ***         63d
  CS80(cs80dsr)              ***        598d
  CS80(cs80dvr)              ***        972d
  CTABLE                               1408c  1564c  1595c  1599c
  CTABLE(brstuff)                       993d  1009c
  CTABLE(ctr)                           255d   572c   768c   787c   796c
  CTABLE(scanstuff)          ***       1137c
  MINIT(cs80ir)              ***        516d
  MINIT(qminit)              ***        878d
  TAPEBKUP(cs80tbdvr)        ***        305d
  TAPEBKUP(cs80tbr)          ***         14d
cs80_devid
  CS80(cs80dvr)                         994d  1021c  1062c  1064c
cs80_devtype
  CS80(cs80dvr)                         993d  1020c  1059c
cs80_hardvols
  CS80(cs80dvr)                         995d  1022c  1067c  1069c
cs80_mp
  CS80(cs80dvr)                         996d  1023c  1071c
cs80_tm_name
  CTABLE(ctr)                           365d   768c   787c
cs80dav
  CTABLE(scanstuff)                    1041d  1133d  1137c
cs80disc_pp
  CTABLE(ctr)                ***        572c
  CTABLE(options)            ***        186d
cs80dsr
  CS80(cs80dsr)              ***        595d
  CS80(cs80dvr)              ***        972d
  MINIT(qminit)              ***        878d
  TAPEBKUP(cs80tbdvr)        ***        305d
cs80dt
  CTABLE                               1214d  1372d  1380c  1381c  1568c
  CTABLE(scanstuff)                    1042d  1054d  1134d  1138c  1141c
cs80dvr
  CS80(cs80dvr)              ***        969d
cs80hardvols
  CTABLE                               1216d  1374d  1380c  1383c  1568c  1579c  1762s  1763s
  CTABLE(scanstuff)                    1043d  1056d  1135d  1138c  1143c
cs80id
  CTABLE                               1215d  1373d  1380c  1568c  1576c  1746s  1756s
  CTABLE(scanstuff)                    1042d  1055d  1134d  1138c  1142c
cs80init
  CS80                       ***         30d
cs80io
  CS80(cs80dvr)                         987d  1351d
cs80ir
  MINIT(cs80ir)              ***        513d
  MINIT(qminit)              ***        878d
cs80mp
  CTABLE                               1375d  1380c  1383c
  CTABLE(scanstuff)                    1043d  1057d  1135c  1138c  1144c
cs80tape_msus
  CTABLE                               1205d  1382c  1444c  1594c  1598c
```

```
cs80tbdvr
  TAPEBKUP                    ***    953d
  TAPEBKUP(cs80tbdvr)         ***    302d
cs80tbr
  TAPEBKUP                    ***    953d
  TAPEBKUP(cs80tbdvr)         ***    305d
  TAPEBKUP(cs80tbr)           ***     11d
csa
  TAPEBKUP(cs80tbr)                  264d   274c   275c   276c   277c   278c
csa_bytes
  TAPEBKUP                    ***   1240c
  TAPEBKUP(cs80tbdvr)                338d   937c
  TAPEBKUP(cs80tbr)                   28d   255d   281c
csamigo
  AMIGO(amigodvr)             ***    425d
  AMIGO(csamigo)              ***     34d
  MINIT(hminit)               ***    248d
  MINIT(iramigo)              ***    187d
cscrolldown
  GCRT(crtb)                         120d   247c
cscrollup
  GCRT(crtb)                         119d   242c
cscrollwinddn
  GCRT(crtb)                         129d   433c
cscrollwindow
  GCRT(crtb)                         128d   431c
cset
  CLOCK(clock)                ***     93c
  SYSDEVS(sysdevs)            ***    199d
csetdate
  CLOCK(clock)                ***    168c
  SYSDEVS(sysdevs)                   198d   471c
csettime
  CLOCK(clock)                ***    169c
  SYSDEVS(sysdevs)                   198d   475c
cshiftleft
  GCRT(crtb)                         124d   385c
cshiftright
  GCRT(crtb)                         125d   391c
cstart
  INIT(misc)                  ***    187s
  LIFDAM(lifmodule)                  567c   570c   614c
  SRMDAM(srmdammodule)               409c   607c
  UCSD_DAM(ucsdmodule)               519c   549c
csvp
  MINIT(hminit)                      413d   429c   455c   456c   458c   460c
ct
  CS80(cs80)                  ***     97d
  MINIT(qminit)                      973c  1063c  1074c  1162c
ct_type
  CS80(cs80)                          78d    97d
ctable
  CTABLE                      ***     36d
ctarec
  UCSD_DAM(ucsdmodule)                45d    48d
```

```
cteol
  CRT(crt)                    ***    472c
  ETU                                105c   112c   125c   132c   200c   230c   244c   297c   300c   560c   720c   732c   733c   801c
                                     933c  1006c  1040c  1061c  1066c  1067c  1123c  1151c
  GCRT(crtb)                  ***    343c
  INIT(misc)                         164d   371c   416c
  LIBRARIAN                           93c   100c   102c  2094c  2098c  2099c  2928c  3080c  3132c  3180c  3196c  3202c  3208c  3209c
                                    3216c  3234c  3250c  3291c  3337c  3339c  3340c  3348c  3353c  3354c  3362c  3374c  3392c  3428c
                                    3446c  3484c  3537c  3545c  3561c  3562c  3569c  3570c  3571c  3579c  3590c  3605c  3675c
  M68KSYS(ci)                        560c   664c   667c   674c   683c   689c   711c   712c   714c
  TAPEBKUP                          1066c  1503c
cteos
  CRT(crt)                    ***    471c
  ETU                                758c   772c   788c   874c  1030c  1175c  1208c
  GCRT(crtb)                  ***    342c
  INIT(misc)                  ***    166d
  LIBRARIAN                         1443c  1451c  3331c
ctet
  AMIGO(csamigo)                     242d   248d   251d   252d   253d   254d   255d   256d   257d   258d   259d   260d   261d   262d
                                     263d
ctp
  UCSD_DAM(ucsdmodule)                48d   477c   511c   532c
ctr
  CTABLE                      ***   1186d
  CTABLE(brstuff)             ***    928d
  CTABLE(ctr)                 ***    229d
  CTABLE(scanstuff)           ***   1035d
ctrlreg
  HPHIL(hphil)                       132d   140c   142c   143c   144c   146c   159d   167c   168c   169c   170c   172c
cts_line
  IOLIB(serial_0)                   2398c  2446c
  KERNEL(iodeclarations)      ***    408d
ctsec
  CLOCK(clock)                       115d   120c   121c
cumdata
  MOUSE(mouse)                        63d    72c   134c   135c   148c
cupdatecursor
  GCRT(crtb)                         122d   199c   429c
cur_disp_lim
  DGL_KNOB(dgl_knot)          ***  18092c
  DGL_VARS(dgl_vars)          ***   1067d
  GEN(dgl_gen)                       3322c  3324c  3360c  3363c  3476c  3494c  3495c  3503c  3518c  3519c  3527c  3536c  3551c  3557c
                                     3566c  3573c  3586c
  LIB(dgl_lib)                      20277c 20278c 21361c
cur_vir_lim
  DGL_INQ(dgl_inq)            ***   6265c
  DGL_VARS(dgl_vars)          ***   1069d
  GEN(dgl_gen)                       3552c  3567c
  LIB(dgl_lib)                      20301c 20302c 20360c 21414c
current
  GLE_FILE(gle_file_io)              9033d  9067c  9068c
  GLE_HPGL(gle_hpgl_out)             7017d  7047c  7061c  7075c  7076c  7088c  7089c  7131c  7149c  7150c  7152c  7156c  7182c  7184c
                                     7186c  7188c  7189c  7213c  7242c  7244c  7246c  7248c  7249c  7326c  7352c  7378c  7395c  7416c
                                     7462c  7481c  7504c  7521c  7574c  7583c  7584c  7585c  7586c  7592c  7596c  7605c  7718c
  GLE_HPGLI(gle_hpgl_in)            18017d 18034c 18048c 18049c 18061c 18062c 18087c 18088c 18089c 18108c 18110c 18112c 18114c 18115c
                                    18148c 18150c 18152c 18154c 18155c 18254c 18255c 18256c 18265c 18266c 18301c 18303c 18305c 18306c
                                    18336c 18347c 18356c 18357c 18358c 18359c 18365c
  GLE_HPIB(gle_hpib_io)             10044d 10519c 10524c 10547c
current_buffer_mode
  GLE_HPGL(gle_hpgl_out)             7061c  7391c  7711c
  GLE_TYPES(gle_types)        ***    1139d
current_color_index
  GLE_HPGL(gle_hpgl_out)      ***    7413c
  GLE_TYPES(gle_types)        ***    1146d
```

```
current_cursor_state
  GLE_HPGL(gle_hpgl_out)              7499c  7709c
  GLE_TYPES(gle_types)        ***     1135d
  LIB(dgl_lib)                       20231c 20236c
current_cursor_x
  GLE_TYPES(gle_types)        ***     1136d
  LIB(dgl_lib)                       20232c 20237c
current_cursor_y
  GLE_TYPES(gle_types)        ***     1137d
  LIB(dgl_lib)                       20232c 20238c
current_drawing_mode
  GLE_TYPES(gle_types)        ***     1148d
  LIB(dgl_lib)                ***    20979c
current_echo_type
  DGL_HPGLI(dgl_hpgli)        ***    18061c
  DGL_KNOB(dgl_knob)          ***    18062c
  DGL_VARS(dgl_vars)          ***     1325d
  GEN(dgl_gen)                ***     3257c
  LIB(dgl_lib)                       20150c 20178c
current_fill_index
  GLE_TYPES(gle_types)        ***     1147d
current_linestyle
  GLE_HPGL(gle_hpgl_out)      ***     7438c
  GLE_SMARK(gle_smark)        ***     5116c
  GLE_TYPES(gle_types)        ***     1141d
  LIB(dgl_lib)                ***    20976c
current_linestyle_mode
  GLE_HPGL(gle_hpgl_out)      ***     7440c
  GLE_SMARK(gle_smark)        ***     5114c
  GLE_TYPES(gle_types)        ***     1144d
  LIB(dgl_lib)                ***    20978c
current_linestyle_pattern
  GLE_HPGL(gle_hpgl_out)      ***     7441c
  GLE_SMARK(gle_smark)        ***     5117c
  GLE_TYPES(gle_types)        ***     1142d
  LIB(dgl_lib)                ***    20974c
current_linewidth
  GLE_TYPES(gle_types)        ***     1149d
  LIB(dgl_lib)                ***    20975c
current_patterr_length
  GLE_HPGL(gle_hpgl_out)      ***     7439c
  GLE_SMARK(gle_smark)        ***     5115c
  GLE_TYPES(gle_types)        ***     1143d
  LIB(dgl_lib)                ***    20977c
current_polygon_blue
  GLE_RGL(gle_ras_out)        ***     8191c
  GLE_TYPES(gle_types)        ***     1154d
current_polygon_color
  GLE_TYPES(gle_types)        ***     1151d
current_polygon_green
  GLE_RGL(gle_ras_out)        ***     8190c
  GLE_TYPES(gle_types)        ***     1153d
current_polygon_red
  GLE_RGL(gle_ras_out)        ***     8189c
  GLE_TYPES(gle_types)        ***     1152d
current_pos_x
  GLE_HPGL(gle_hpgl_out)              7346c  7372c  7707c
  GLE_SMARK(gle_smark)        ***     5121c
  GLE_TYPES(gle_types)        ***     1110d
  LIB(dgl_lib)                ***    20228c
```

```
current_pos_y
  GLE_HPGL(gle_hpgl_out)              7347c  7373c  7708c
  GLE_SMARK(gle_smark)        ***     5122c
  GLE_TYPES(gle_types)        ***     1111d
  LIB(dgl_lib)                ***    20229c
current_record
  SRM_DRV(srm)                ***     577d
current_timeout
  PRINTER(prtdvr)                      76d   219c   233c   241c   251c   282c
current_vol
  CTABLE(ctr)                          313d   314d   604d   609c   615d   617c
current_vu
  CS80(cs80)                  ***     208d
  CS80(cs80dsr)               ***     763c
  TAPEBKUP(cs80tbdvr)                  653c   886c
currentcrt
  CRT(crt)                    ***     708c
  C_HOOK                      ***     142c
  DGL_C_OUT(dgl_confg_out)            11183c 11286c 11288c 11296c 11302c 11332c 11335c 11342c 11346c
  GCRT(crtb)                  ***     520c
  SYSDEVS(sysdevs)                     130d   631c
currentif
  CS80(cs80)                  ***     116d
  MINIT(qminit)               ***     914c
cursaddr
  CRT(crt)                             270d   272c   273c   274c   275c   276c   664d   678c   679c   680c   681c   682c
  GCRT(crtb)                  ***     197d
cursor
  GLE_GEN(gle_gen)            ***    2180c
  GLE_HPGL(gle_hpgl_out)      ***    7544c
  GLE_RGL(gle_ras_out)        ***    8417c
  GLE_TYPES(gle_types)        ***    1034d
  LIB(dgl_lib)                       20162d 20180c 20197c 20207c 21284c
cursor_color
  DGL_VARS(dgl_vars)          ***    1331d
  LIB(dgl_lib)                       20990c 21448c
cursor_size_x
  DGL_VARS(dgl_vars)          ***    1328d
cursor_size_y
  DGL_VARS(dgl_vars)          ***    1329d
cursor_x
  GLE_RGL(gle_ras_out)        ***    8047d
cursor_y
  GLE_RGL(gle_ras_out)        ***    8048d
cursoraddr
  GCRT(crtb)                           107d   481c   482c
cursormask
  CRT(crt)                             107d   153d
  GCRT(crtb)                  ***      86d
  SYSDEVS(sysdevs)            ***     100d
cursx
  CRT(crt)                             566c   614c   619c   627c   634c   649c
  GCRT(crtb)                           429c   451c   456c   466c   472c
  SYSDEVS(sysdevs)            ***      60d
cursy
  CRT(crt)                             566c   614c   619c   628c   634c   649c
  GCRT(crtb)                           429c   451c   456c   466c   472c
  SYSDEVS(sysdevs)            ***      60d
cvol
  ETU                                   63d   241c   242c   247c   274c   276c   281c   363c   376c   897c   899c   933c
cvtdatetime
  LIFDAM(lifmodule)                    548d   573c   574c   617c
```

cyclic															
A804(DVR(a804xdvr)		263c	265c	291c	294c										
SYSDEVS(sysdevs)	***	213d													
cyl															
AMIGO(csamigo)		90d	352c	357c	372c	373c									
cyl_per_med															
AMIGO(csamigo)		129d	193d	194d	196d	197d	198d	199d	200d	201d	202d	203d	225c	351c	352c
		372c													
MINIT(hminit)		292c	487c												
cylinder_compare_error															
AMIGO(amigodvr)		806c	821c												
AMIGO(csamigo)	***	72d													
cylinder_ert															
MINIT(cs80ir)	***	551d													
cyln															
MINIT(cs80ir)		522d	564d	818d	838c										
d															
AMIGO(csamigo)	***	100d													
BAT(bat)		69d	70c												
GLE_UTLS(gle_utls)		21010d	21106d	21113c	21115c										
LIBRARIAN		415d	630c	665c	668c	674c	853c	886c	894c	1015c	1019c	1036c	1069c	1098c	1113c
		1115c	1126c	1130c	1134c										
LIFDAM(lifmodule)		113d	120c	121c	122c	123c	124c	125c	126c						
MIUI		15d	179c	202c	321c	351c	474c								
MORE_FSYS(mfs)		19d	22d	553d	560d	580c	587c	593c	598c	606c	634c	635c	636c	637c	642c
		650c	651c	652c	667d	673c									
MOUSE(mouse)		41d	71c	83c	89c	90c	91c	101c	132d	134c	141c	146c			
SYSDEVS(sysdevs)		470d	471c												
UCSD_DAM(ucsdmodule)		67d	145c	146c	149c	159d	160c	309c	310c	316c	317c	360c	363c	373c	411c
		412c	421c	451c	452c	453c	477c	497c	538c	539c	590c	591c	613c	617c	
d0															
CTABLE(brstuff)	***	949d													
UCSD_DAM(ucsdmodule)		67d	87c	89c	142c	152c	153c	215c	220c	222c	233c	253c	254c	260c	273c
		336c	361c	384c	394c	467c	468c	511c	536c	586c	587c	620c	621c	654c	657c
d1															
CTABLE(brstuff)		949d	1002c												
d10															
CTABLE(brstuff)	***	949d													
d11															
CTABLE(brstuff)	***	949d													
d12															
CTABLE(brstuff)	***	949d													
d13															
CTABLE(brstuff)	***	949d													
d14															
CTABLE(brstuff)	***	949d													
d15															
CTABLE(brstuff)	***	949d													
d16															
CTABLE(brstuff)	***	950d													
d17															
CTABLE(brstuff)	***	950d													
d18															
CTABLE(brstuff)	***	950d													
d19															
CTABLE(brstuff)	***	950d													
d2															
CTABLE(brstuff)	***	949d													
KERNEL(general_0)		829d	834d	844d											
d20															
CTABLE(brstuff)	***	950d													
d21															
CTABLE(brstuff)	***	950d													

d22															
CTABLE(brstuff)	***	950d													
d23															
CTABLE(brstuff)	***	950d													
d24															
CTABLE(brstuff)	***	950d													
d25															
CTABLE(brstuff)	***	950d													
d26															
CTABLE(brstuff)	***	950d													
d27															
CTABLE(brstuff)	***	950d													
d28															
CTABLE(brstuff)	***	950d													
d29															
CTABLE(brstuff)	***	950d													
d3															
CTABLE(brstuff)	***	949d													
d30															
CTABLE(brstuff)	***	950d													
d31															
CTABLE(brstuff)	***	950d													
d4															
CTABLE(brstuff)	***	949d													
d5															
CTABLE(brstuff)	***	949d													
d6															
CTABLE(brstuff)	***	949d													
d7															
CTABLE(brstuff)	***	949d													
d8															
CTABLE(brstuff)	***	949d													
d9															
CTABLE(brstuff)	***	949d													
d_address															
TAPEBKUP(cs80tbr)		231d	247c	248c											
d_byte															
MINIT(iramigo)		192d	221d	237c											
d_loc_echo_x															
DGL_HPGLT(dgl_hpgli)	***	18072c													
DGL_KNOB(dgl_knob)	***	18098c													
DGL_VARS(dgl_vars)	***	1271d													
GEN(dgl_gen)	***	3378c													
LIB(dgl_lib)		20152c	20154c	20157c	20195c	20202c	20203c	20206c	20874c						
d_loc_echo_y															
DGL_HPGLT(dgl_hpgli)	***	18073c													
DGL_KNOB(dgl_knob)	***	18099c													
DGL_VARS(dgl_vars)	***	1272d													
GEN(dgl_gen)	***	3378c													
LIB(dgl_lib)		20151c	20154c	20155c	20195c	20202c	20205c	20206c	20874c						
d_setaid															
TAPEBKUP(cs80tbr)		230d	246c												
d_volunit															
TAPEBKUP(cs80tbr)		229d	244c	245c											
daccess															
INIT_LOAD(loader)	***	840d													
LIBRARIAN		2729c	2897c												
UCSD_DAM(ucsdmodule)		146c	415c	540c	554c	592c									
daddr															
HPHIL(hphil)	***	256c													
MOUSE(mouse)	***	188c													
SYSDEVS(sysdevs)	***	338d													

```
dam
  ASCII(asciimodule)          ***     81c
  CTABLE                             1268d  1274c  1278c
  CTABLE(ctr)                         447d   471c   506d   511c   512c   515c   516c   518c   520c   639c   658c   694c   707c   716c
                                      733c   752c   768c   787c   809c   816c
  ETU                                 341c   370c   405c   904c   926c
  GLE_KNOB(gle_knob_in)       ***   18041c
  INIT(initunits)             ***    2204c
  INIT(misc)                          498c   617c
  INITLOAD                    ***    1677c
  INITLOAD(loader)                   1391c  1610c
  INITLOAD(sysglobals)        ***     144d
  LOCKMOD(lockmodule)                  49c    72c    90c
  M68KSYS(ci)                         160c   287c   615c   825c
  MIUI                                249c   529c   533c
  TAPEBKUP                    ***    1136c
  UCSD_AM(ucsc_am)            ***      57c
  UNITIO(uio)                 ***      48c
dam_fib
  MIUI                                376d   525c   533c
dam_name
  CTABLE(ctr)                         422d   460c   498c
dam_proc
  CTABLE(ctr)                         445d   460c   461c   466c   471c
damrequesttype
  INIT(initunits)                    2141d  2160d  2165d  2232d
  INIT(misc)                          213d   426d
  INITLOAD(bootdammodule)             543d   607d
  INITLOAD(sysglobals)                133d   140d
  LIFDAM(lifmodule)                    28d   195d
  SRMDAM(srmdammodule)                 45d  1626d
  UCSD_DAM(ucsdmodule)                 32d    66d
damtype
  CTABLE(ctr)                 ***     447d
  INIT(initunits)             ***    2182d
  INITLOAD(sysglobals)                140d   144d
danish_kbd
  A804XDVR(a804xdvr)          ***     191d
  NONUSKBD2                           179c   206c   209c
  SYSDEVS(sysdevs)            ***     147d
darray
  HPHIL(hphil)                        101c   103c   107c   109c   114c   121c   196c
  SYSDEVS(sysdevs)            ***     326d
dat
  SRM_DRV                            2000d  2007c  2086d  2093c  2104c
  SRM_DRV(srm)                       1100d  1113d
data
  A804XDVR(a804xdvr)                   34d    41d    42d    80d    83c    91d    95c   100c   101c   102c   103c   105c   106c   108c
                                      142d   143c   146d   147c   169d   174c   175c   181d   203c   209s   218c   221c   223c   224c
  BAT(bat)                             85d    87c
  CLOCK(clock)                        163d   166c   167c   168c   169c
  CRT(crt)                            263d   266c
  GLE_FILE(gle_file_io)              9034d  9067c
  GLE_HPGL(gle_hpgl_out)             7018d  7075c  7088c  7151c  7182c  7184c  7186c  7188c  7242c  7244c  7246c  7248c  7583c
  GLE_HPGLI(gle_hpgl_in)            18018d 18048c 18061c 18088c 18108c 18110c 18112c 18114c 18148c 18150c 18152c 18154c 18254c 18255c
                                    18265c 18301c 18303c 18305c 18356c
  GLE_HPIB(gle_hpib_io)             10045d 10520c 10543c 10544c 10545c 10546c
  INITLOAD(bootdammodule)             780d   788c
  KEYS(keys)                          199c   201c   419c   427c   433c   443c   519c   540c
  NONUSKBD1(non_us_kbd1)               44c    46c    51c    52c    55c    56c
  SRM_DRV                     ***    2104c
  SRM_DRV(srm)                        723d   809d
  SYSDEVS(sysdevs)                    187d   205d   206d   263d   441d   442c   447d   453d   455d
```

```
data_communication
  CTABLE(scanstuff)           ***    1087d
data_error_log
  MINIT(cs80ir)                       565d   728d   766c
  MINIT(qminit)               ***    1197c
data_ptr
  DGL_C_OUT(dgl_confg_out)    ***   11167d
  GLE_FILE(gle_file_io)              9018d  9062d  9065c
  GLE_HPIB(gle_hpib_io)             10026d 10027d 10507d 10514c 10528d 10535c
data_records
  SRMDAM(srmdammodule)               1087c  1124c  1135c  1147c  1421c
  SRM_DRV(srm)                ***     158d
databuffer
  A804XDVR(a804xdvr)                   67d    83c   128c
databyte
  A804XDVR(a804xdvr)                   71d    74d    77c
  HPHIL(hphil)                         71d    96d   101c   103c   107c   109c   114c   121c   282d   287c   302d   305c   316c
  MOUSE(mouse)                         75d    83c    89c    91c   101c   111c   114c   118c   119c   129d   141c   146c   163c   170c
  SYSDEVS(sysdevs)                     37d   402d
dataended
  HPHIL(hphil)                        327c   340c
  MOUSE(mouse)                ***     157c
  SYSDEVS(sysdevs)            ***     302d
datafib
  LIFDAM(lifmodule)                   390d   404c   405c   440c   444c   447c   469c
datafile
  INIT(misc)                          472c   718c   720c
  INITLOAD(bootdammodule)             668c   673c   707c
  INITLOAD(loader)            ***     837d
  INITLOAD(sysglobals)        ***      51d
  LIFDAM(lifmodule)                   142c   169c   296c   544c   923c   960c  1004c
  SRMDAM(srmdammodule)                 90c   950c  1451c
  UCSD_DAM(ucsdmodule)                 82c   165c
datahook
  A804XDVR(a804xdvr)                   36d   101c   395c
datahooktype
  A804XDVR(a804xdvr)                   34d    36d
dataisr
  A804XDVR(a804xdvr)                   80d   395c
dataoverrun
  F9885(f9885dvr)                      49d   241c
dataproc
  HPHIL(hphil)                        195c   260c   344c
  MOUSE(mouse)                ***     190c
  SYSDEVS(sysdevs)                    333d   340d
datareg
  A804XDVR(a804xdvr)                   61d    95c   143c
datasize
  INITLOAD(loader)                    922d  1460c
  LIBRARIAN                          1925c  2440c  2475c  2486c  2488c  2561c  2659c  2700c
datasource
  MINIT(cs80ir)                       787d   799c
dataspace
  LIFDAM(lifmodule)                   680d   718c   741c   744c   746c   754c   756c   773c   781c   783c   795c   809c   810c   855c
                                      856c   887c   889c   891c   892c   893c  1017d  1038c  1045c  1048c  1052c
datastarting
  HPHIL(hphil)                ***     345c
  MOUSE(mouse)                ***     156c
  SYSDEVS(sysdevs)            ***     302d
datatype
  INITLOAD(loader)                    857d   922d   931d
  LIBRARIAN                   ***    2334d
  SRMDAM(srmdammodule)                 70d  1452c
```

```
date
  INITLOAD(loader)              ***    941d
  INITLOAD(sysglobals)          ***    230d
  LIBRARIAN                            106d   111c   331c  1960c
  LIFDAM(lifmodule)                    548d   550c   551c   552c   553c   554c
  M68KSYS(ci)                          110d   123c   124c   153c   155c   163c   285c
  MIUI                                  52d    61c    64c
  SRMDAM(srmdammodule)                 366c   367c   368c  1465c
  SRM_DRV(srm)                  ***    251d
  UCSD_DAM(ucsdmodule)                 654c   657c
date_type
  SRMDAM(srmdammodule)          ***    356d
  SRM_DRV(srm)                         249d   282d   283d   736d   738d
daterec
  CLOCK(clock)                          49d   100d
  INIT(misc)                    ***    190d
  INITLOAD(loader)                     836d   840d   941d   942d
  INITLOAD(sysglobals)                 217d   230d
  LIBRARIAN                             38d    39d   106d
  LIFDAM(lifmodule)                    115d   324d   339d   548d   585d
  M68KSYS(ci)                          110d   178d   778d
  MIUI                                  44d    52d
  SRMDAM(srmdammodule)                 357d   387d
  SYSDEVS(sysdevs)                     203d   384d   386d   463d   469d   470d
dateset
  M68KSYS(ci)                          172d  1138c  1183c
datetimerec
  INITLOAD(sysglobals)          ***    229d
  M68KSYS(ci)                          150d   179d
  SRMDAM(srmdammodule)          ***   1438d
  UCSD_DAM(ucsdmodule)          ***     49d
datetype
  CLOCK(clock)                         166c   168c
  SYSDEVS(sysdevs)              ***    203d
dav
  CTABLE                              1193d  1195d  1245c  1246c  1247c  1248c  1270c  1316c  1334c  1344c  1359c  1364c  1380c  1416c
                                      1419c  1422c  1463c  1471c  1525c  1568c  1594c  1598c
  CTABLE(brstuff)                      934d  1007c  1008c  1011c  1012c  1016c  1017c
  CTABLE(ctr)                          281d   413c   414c   415c   416c   495c
dav_line
  KERNEL(iodeclarations)        ***    394d
dav_type
  CTABLE                              1191d  1208d  1209d  1210d  1341d
  CTABLE(brstuff)               ***    934d
  CTABLE(ctr)                   ***    281d
  CTABLE(options)                      112d   118d   120d   123d   125d   127d
  CTABLE(scanstuff)                   1040d  1041d  1083d  1133d
day
  CLOCK(clock)                          69c   106c
  INITLOAD(sysglobals)          ***    219d
  LIBRARIAN                            113c   116c  3711c
  LIFDAM(lifmodule)                    122c   333c   348c   552c   553c   554c   587c
  M68KSYS(ci)                          125c   131c   133c   135c   156c   264c   779d
  MIUI                                  44d    65c    68c    69c    89c
  SRMDAM(srmdammodule)                 367c   387d
  SRM_DRV(srm)                  ***    253d
  UCSD_DAM(ucsdmodule)                 522c   551c
davy
  M68KSYS(ci)                          112d   115c   125c   131c   133c
db
  MINIT(iramigo)                       229d   237c
```

```
dbcinfo
  CRT(crt)                      ***    544d
  GCRT(crtb)                    ***    413d
  SYSDEVS(sysdevs)                      55d    64d   489d
dbclear
  CRT(crt)                      ***    621c
  GCRT(crtb)                    ***    462c
  SYSDEVS(sysdevs)              ***     53d
dbcline
  CRT(crt)                      ***    626c
  GCRT(crtb)                    ***    466c
  SYSDEVS(sysdevs)              ***     53d
dbcrt
  SYSDEVS(sysdevs)                      64d   489d
dbcrthook
  CRT(crt)                      ***    704c
  GCRT(crtb)                    ***    514c
  SYSDEVS(sysdevs)                     127d   630c
dbcrtop
  CRT(crt)                      ***    544d
  GCRT(crtb)                    ***    413d
  SYSDEVS(sysdevs)                      53d    64d   489d
dbcrttype
  SYSDEVS(sysdevs)                      64d   127d
dbexcg
  CRT(crt)                      ***    638c
  GCRT(crtb)                    ***    477c
  SYSDEVS(sysdevs)              ***     53d
dbgotoxy
  CRT(crt)                      ***    563c
  GCRT(crtb)                    ***    429c
  SYSDEVS(sysdevs)              ***     53d
dbhighl
  CRT(crt)                      ***    612c
  GCRT(crtb)                    ***    451c
  SYSDEVS(sysdevs)              ***     54d
dbinfo
  CRT(crt)                      ***    561c
  GCRT(crtb)                    ***    427c
  SYSDEVS(sysdevs)              ***     53d
dbinit
  CRT(crt)                      ***    630c
  GCRT(crtb)                    ***    468c
  SYSDEVS(sysdevs)              ***     53d
dbput
  CRT(crt)                      ***    619c
  GCRT(crtb)                    ***    453c
  SYSDEVS(sysdevs)              ***     53d
dbrec
  CRT(crt)                             544d   558c
  GCRT(crtb)                           413d   424c
dbscrolldn
  CRT(crt)                      ***    571c
  GCRT(crtb)                    ***    433c
  SYSDEVS(sysdevs)              ***     54d
dbscrolll
  CRT(crt)                             591c   595c   604c
  GCRT(crtb)                    ***    435c
  SYSDEVS(sysdevs)              ***     54d
dbscrollr
  CRT(crt)                      ***    591c
  GCRT(crtb)                    ***    443c
  SYSDEVS(sysdevs)              ***     54d
```

```
dbscrollup
  CRT(crt)                          571c   574c
  GCRT(crtb)                  ***   431c
  SYSDEVS(sysdevs)            ***    53d
dc_drivers
  DC_DRV(init_dc)                   585d   612c   613c   642c
dc_dummy
  KERNEL(general_0)                 966d  1124c  1125c  1127c  1128c
dc_err_ptr
  DC_DRV(intdc)                     239d   242d
dc_err_type
  DC_DRV(intdc)                     237d   239d
dc_error
  DC_DRV(init_dc)                   702c   704c   707c
  DC_DRV(intdc)                     242d   282c   284c
dc_init_fault
  DC_DRV(init_dc)                   668c   699c   701c
  DC_DRV(intdc)                     241d   280c
dc_initialize
  DC_DRV                      ***   172d
dcd_file_type
  SRM_DRV(srm)                ***   220d
dcd_line
  IOLIB(serial_0)                  2393c  2441c
  KERNEL(iodeclarations)      ***   409d
dcd_system_type
  SRM_DRV(srm)                ***   219d
dcl_message
  IOLIB(hpib_2)               ***  1847c
  KERNEL(iodeclarations)      ***   306d
dcursoraddr
  GCRT(crtb)                        473c   482c   483c
  SYSDEVS(sysdevs)            ***    58d
dd
  CLOCK(clock)                       50d    57c    58c    60c    61c    69c
de_log
  MINIT(cs80ir)                     565d   728d   752c
  MINIT(qminit)                    1104d  1191c  1197c
de_log_type
  MINIT(cs80ir)                     527d   565d   728d   737d
  MINIT(qminit)               ***  1104d
deallocate_bkgnd_info
  AMIGO(amigodvr)                   583c   618c   966c
  CS80(cs80dvr)                    1007c  1131c
  DISCHPIB(bkgnd)                    63d   125d   150c
  MINIT(hminit)               ***    495c
  MINIT(qminit)                      991c   994c  1271c
  TAPEBKUP(cs80tbdvr)         ***    353c
deallocate_bkgnd_info_proc
  CTABLE(scanstuff)                 1071d  1127c  1156c  1166c
debugflag
  MINIT(xminit)                     1394d  1504c  1508c  1546c  1559c  1569c  1580c  1615c  1623c  1642c  1660c  1678c  1696c  1709c
                                    1719c  1743c  1750c  1755c  1779c  1794c  1824c
debugger
  INIT                        ***  2543c
  INITLOAD(sysglobals)        ***   270d
  KEYS(keys)                         67c    70c   419c
  M68KSYS(ci)                       885c   890c  1170c  1192c
debugging
  M68KSYS(ci)                       869d   885c   904c   912d   930c   938c   987d   992c  1001c  1016d  1019c  1026c
debuginput
  MINIT(xminit)               ***  1399d
debugkey
  KEYS(keys)                        418d   505c   520c   525c
```

```
decode
  LIBRARIAN                         543d  1198c  1206c  1212c  1239c
decoded_addr
  AMIGO(amigodvr)             ***   839c
  AMIGO(csamigo)                    142d   363d   374c   415c
decodestate
  LIBRARIAN                         308d  1145c  1193c  1198c  1213c  1214c  1216c  1223c  1228c  1242c  1440c  1455c  1457c  1460c
                                   1461c  1462c  1463c  1469c
decodestatetype
  LIBRARIAN                         298d   308d
decodestuff
  LIBRARIAN                        1159d  1248c
decpnt
  MOREFSYS(mfs)                     402d   413c   446c   500c
def
  INITLOAD(loader)            ***  1039d
  LIBRARIAN                         2026c  2043c  2051c  2052c  2053c  2059c  2072c  2076c  2077c  2082c  2087c  2092c  2101c  2102c
  MINIT(hminit)                      264d   275d   279c
  MINIT(midecs)               ***     23d
  MINIT(mminit)               ***    126d
  MINIT(qminit)               ***    914c
  MIUI                               295c   302c   306c   316c
def_disp_lim
  DGL_VARS(dgl_vars)          ***  1055d
  LIB(dgl_lib)                     21130c 21131c 21132c 21133c 21143c
def_loc_lim
  DGL_VARS(dgl_vars)          ***  1061d
  LIB(dgl_lib)                     21043c 21340c 21363c
defaddr
  CTABLE(ctr)                        394c   395c
  INITLOAD(loader)                  1013d  1187c  1259c  1266c  1309c  1348c  1431c  1452c  1467c  1486c
  LIBRARIAN                         1165d  1168c  1169c  1170c  1172c  1174c  1181c  1187c  1188c  1326c  1559c  1560c  2026c  2043c
  SEGMENTER(asm)                     122c   123c
default
  ETU                                192d   197c
default_clear_logs_option_byte
  MINIT(qminit)                      903d  1066c
default_color_table_size
  DGL_VARS(dgl_vars)          ***  1004d
  LIB(dgl_lib)                ***  21454c
default_initialize_options_byte
  MINIT(qminit)                      901d  1064c
default_msus
  CTABLE(brstuff)                    973d   999c
default_poly_table
  DGL_HPGL(dgl_hpgl)               17226d 17285c
  DGL_RAS(dgl_raster)              17561d 17655c
default_poly_table_def
  DGL_HPGL(dgl_hpgl)               17223d 17226d
  DGL_RAS(dgl_raster)              17547d 17561d
default_poly_table_size
  DGL_VARS(dgl_vars)          ***  1005d
  LIB(dgl_lib)                ***  21453c
default_volume_ert_passes
  MINIT(qminit)                      902d  1065c
defaulthighlight
  CRT(crt)                           165d   416c   683c
  GCRT(crtb)                         113d   286c   377c   455c   509c
defaulttimeout
  SRM_DRV(srm)                       994d  1222c  1237c  1422c
defblock
  INITLOAD(loader)                   957d  1264c
  LIBRARIAN                          375c  2113c
```

```
defblocks
  LIBRARIAN                          2011d  2116c  2117c  2118c  2119c
defdelta
  INITLOAD(loader)                   1476d  1479c  1484c  1486c
defective_track
  AMIGO(amigodvr)              ***    814c
  AMIGO(csamigo)               ***     77d
deferredaction
  INIT(misc)                   ***    237c
  INITLOAD(sysglobals)         ***    277d
  SYSDEVS(sysdevs)                    500c   505c
define_color_map
  GLE_GEN(gle_gen)             ***   2198c
  GLE_HPGL(gle_hpgl_out)       ***   7550c
  GLE_RGL(gle_ras_out)               8280c  8420c  8482c
  GLE_TYPES(gle_types)         ***   1035d
define_drawing_mode
  GLE_GEN(gle_gen)             ***   2204c
  GLE_HPGL(gle_hpgl_out)       ***   7549c
  GLE_RGL(gle_ras_out)         ***   8418c
  GLE_TYPES(gle_types)         ***   1036d
definecaseword
  LIBRARIAN                          1149d  1221c  1230c
defineword
  LIBRARIAN                           494d  1144c  1207c
delimit
  LIBRARIAN                          1165d  1169c  1170c  1187c
defmodptr
  INITLOAD(loader)                   1179d  1182c  1183c  1185c  1187c  1188c  1189c
defptr
  INITLOAD(loader)                   1248d  1262c  1264c  1265c  1269c  1276c
  LIBRARIAN                          1484d  1572c  1574c  1577c  1578c  1579c
  MINIT(xminit)                      1411d  1447c  1565c  1644c  1645c  1651c  1747c  1748c  1754c  1781c  1785c
defsize
  CTABLE(ctr)                  ***    395c
  INITLOAD(loader)                    958d  1014d  1188c  1258c  1259c  1260c  1262c  1264c  1265c  1310c  1349c  1432c  1467c
  LIBRARIAN                           375c  1169c  1327c  1560c  2019c  2043c  2114c  2115c  2116c
  SEGMENTER(asm)               ***    123c
defsout
  LIBRARIAN                           49d  2114c  2974c  3577c  3630c
deftable
  INITLOAD(loader)             ***   1172d
  MINIT(xminit)                      1409d  1448c  1651c  1754c  1787c  1792c
deftablelength
  INITLOAD(loader)             ***   1175d
deg_to_rad
  DGL_POLY(dgl_poly)                20042d 20899c 20993c
del
  INIT(misc)                   ***    173d
  KEYS(keys)                          478c   509c
  MINIT(cs80dir)                      742d   750c   751c   752c   753c   754c
  MOREFSYS(mfs)                        53d    63c   124c   151c   183c   215c   255c   293c   326c   365c
del1
  SRMDAM(srmdammodule)                217d   225c   665d   674c   684c   689c   691c   697c
del2
  SRMDAM(srmdammodule)                218d   226c   666d   674c   689c
delay71
  A804XDVR(a804xdvr                   272c   299c   317c
  SYSDEVS(sysdevs)             ***    213d
delay_timer
  DISCHPIB(dischpib                   216d   281c   340c   344c
delayt
  A804XDVR(a804xdvr                   263c   291c
  SYSDEVS(sysdevs)             ***    213d
```

```
delentry
  UCSD_DAM(ucsdmodule)                379d   401c   423c   433c
delim
  SRMDAM(srmdammodule)                212d   222c   224c   229c   251c   266c   277c   279c
delta2
  LIBRARIAN                          2367d  2453c  2454c  2494c
delta_x_21
  DGL_POLY(dgl_poly)                20275d 20287c 20297c 20298c 20307c 20308c
delta_x_31
  DGL_POLY(dgl_poly)                20277d 20290c 20298c 20308c
delta_x_43
  DGL_POLY(dgl_poly)                20279d 20293c 20297c 20307c
delta_y_21
  DGL_POLY(dgl_poly)                20276d 20288c 20297c 20298c 20307c 20308c
delta_y_31
  DGL_POLY(dgl_poly)                20278d 20291c 20298c 20308c
delta_y_43
  DGL_POLY(dgl_poly)                20280d 20294c 20297c 20307c
dend
  LIFDAM(lifmodule)                   200d   284c   415c   869c
denom
  DGL_POLY(dgl_poly)                20274d 20297c 20300c 20301c
density
  DGL_HPGL(dgl_hpgl)                17227d 17228d 17229d 17230d 17231d 17232d 17233d 17234d 17235d 17236d 17237d 17238d 17239d 17240d
                                    17241d 17242d
  DGL_INQ(dgl_inq)             ***   6076c
  DGL_POLY(dgl_poly)                20156c 20157c 20184c
  DGL_RAS(dgl_raster)               17562d 17563d 17564d 17565d 17566d 17567d 17568d 17569d 17570d 17571d 17572d 17573d 17574d 17575d
                                    17576d 17577d
  DGL_VARS(dgl_vars)           ***   1033d
dentry
  INITLOAD(bootdammodule)             613d   665c   668c   671c   673c   674c   711c
  LIFDAM(lifmodule)                   201d   300c   418c   419c   421c   425c   428c   434c   437c   438c   453c   457c   465c   466c
                                      534c   535c   597c   598c   601c   602c   607c   639c   640c   646c   664c   807c   820c   855c
                                      860c   876c   915c   916c   935c   936c   937c   938c   939c   940c   941c   942c   943c   944c
                                      945c   954c   956c   999c  1002c  1009c  1021c  1023c  1026c  1032c  1038c  1050c  1052c  1053c
                                     1054c  1076c  1077c  1078c  1093c  1094c  1095c  1165c
deovblk
  INITLOAD(loader)             ***    833d
  LIBRARIAN                          3108c  3159c
  M68KSYS(ci)                  ***    776d
  UCSD_DAM(ucsdmodule)                98c   233c   361c   488c   490c   517c   518c
describe
  CS80(cs80)                          344d   485d   496c
  CS80(cs80dsr)                ***    909c
  CS80(cs80dvr)                ***   1053c
  HPHIL(hphil)                        179d   244c
  MINIT(qminit)                       968c  1267c
  TAPEBUP(cs80tbdvr)           ***    442c
describe_bytes
  CS80(cs80)                          344d   485d   494c
  CS80(cs80dsr)                       777d   909c   913c
  CS80(cs80dvr)                      1030d  1053c  1057c
  MINIT(qminit)                       891d   908c   921c   968c   973c  1002c  1016c  1060c  1162c  1192c  1267c
  TAPEBUP                            1249c  1299c  1316c  1440c  1470c
  TAPEBUP(cs80tbdvr)                  312d   442c   481c   492c
describe_type
  CS80(cs80)                           92d   344d   485d
  CS80(cs80dsr)                ***    777d
  CS80(cs80dvr)                ***   1030d
  MINIT(qminit)                ***    891d
  TAPEBUP(cs80tbdvr)           ***    312d
describedata
  HPHIL(hphil)                         96c   189c
```

```
descrip
  HPHIL(hphil)                          101c   103c   104c   107c   109c   110c   114c   115c   116c   118c   121c   196c   257c
  MOUSE(mouse)                           88c    92c    93c   159c   160c
  SYSDEVS(sysdevs)              ***     331d
descriprec
  SYSDEVS(sysdevs)                      311d   331d
dest
  LIBRARIAN                             546d   914c
destaddr
  SRM_DRV(srm)                  ***     300d
destfid
  SRM_DRV(srm)                         1038d  1705d  1718c
destination_blk_addr
  TAPEBKUP(cs80tbr)                      34d   210d   248c
destination_block_address
  TAPEBKUP                             1240c  1263c  1345c
  TAPEBKUP(cs80tbdvr)                   334d   901c
destination_block_size
  TAPEBKUP                             1221d  1285c  1319c  1416c
destination_file_id
  SRM_DRV(srm)                          476d  1718c
destination_is_a_7914
  TAPEBKUP                             1220d  1237c  1258c  1315c  1374c
destination_is_a_tape
  TAPEBKUP                             1219d  1231c  1243c  1255c  1268c  1280c  1318c  1323c  1339c  1371c  1379c  1413c
destination_lun
  TAPEBKUP                             1217d  1307c  1308c  1310c  1314c
destination_offset
  SRM_DRV(srm)                          477d  1719c
destination_size
  TAPEBKUP                             1222d  1250c  1268c  1282c  1284c  1320c  1323c  1381c  1382c  1415c
destination_uep
  TAPEBKUP                             1232c  1247c  1248c  1256c  1269c  1286c  1308c  1309c  1312c  1315c  1372c  1387c  1417c
  TAPEBKUP(cs80tbdvr)                   332d   742c   901c
  TAPEBKUP(cs80tbr)                      33d   210d   244c   245c
destoff
  SRM_DRV(srm)                         1039d  1706d  1719c
dev
  AMIGO(amigodvr)                       506d   511c   519c   520c   534c
  CTABLE(brstuff)                       956d  1002c  1006c
  MIUI                                  319d   321c
dev_adr
  LIB(dgl_lib)                        20025d 20038d 21245d 21264c 21278c 21295d 21317c 21328c
dev_buf_ptr
  GLE_TYPES(gle_types)                 1065d  1088d  1215d  1218d
dev_dep_stuff
  DGL_C_IN(dgl_confg_in)              12034c 12054c
  DGL_C_OUT(dgl_confg_out)            11056c 11059c 11281c 11394c 11427c
  DGL_INQ(dgl_inq)                     6198c  6214c
  DGL_RAS(dgl_raster)                 17186c 17441c 17592c
  GLE_GEN(gle_gen)              ***     2132c
  GLE_GENI(gle_geni)            ***     3030c
  GLE_HPGL(gle_hpgl_out)               7045c  7100c  7121c  7172c  7229c  7335c  7361c  7404c  7433c  7471c  7490c  7518c
  GLE_KNOB(gle_knob_in)               18083c 18119c 18186c 18197c 18216c
  GLE_RGL(gle_ras_out)                 8181c  8231c  8296c  8326c  8381c
  GLE_TYPES(gle_types)                 1072d  1222d
dev_name
  MIUI                                  106d   193c
dev_name_type
  MIUI                                  103d   107d
dev_type
  CTABLE(brstuff)                       948d   956d   987d
devaddr
  HPHIL(hphil)                           52d   343c   349c
```

```
device
  AMIGO(amigodvr)                       511c   659c   685c   689c   752c   958c
  AMIGO(csamigo)                        137d   180d   163c   164c   165c   166c   167c   168c   169c   170c   171c   179c   185c   207c
                                        315c
  GLE_HPIB(gle_hpib_io)               10188d 10196c 10198c 10211c 10213c 10227c 10276d 10284c 10286c 10299c 10303c 10315c
  IOLIB(general_2)                      797d   799d   802d   806d   808d   811d   814d   817d   820d   822d   832d   853c   913d   920c
                                        936d   941c   966d   972c   992d   997c  1012d  1016c  1025d  1029c  1038d  1042c  1051d  1055c
                                       1071d  1076c
  IOLIB(general_4)                     1199d  1205d  1211d  1218d  1223d  1351d  1360c  1361c  1401c  1415c  1436d  1445c  1460d  1469c
                                       1486d  1494c  1529d  1543c
  IOLIB(hpib_1)                         403d   406d   408d   411d   510d   518c   520c   533c   535c   549c   598d   606c   608c   621c
                                        625c   637c   696d   701c   703c   706c   744d   750c   752c   756c
  IOLIB(hpib_2)                        1780d  1783d  1787d  1789d  1792d  1793d  1799d  1838d  1841c  1842c  1861d  1864c  1866c  1870c
                                       1897d  1900c  1902c  1905c  1908c  1918d  1920c  1921c  1935d  1938c  1939c  1951d  1954c  1956c
                                       1960c  1963c  1989d  1991c
  IOLIB(hpib_3)                        2033d  2094d  2099c
  MINIT(hminit)                 ***     489c
  MIUI                                   20d   171c   172c   173c   174c   175c   176c   177c   178c   179c   180c   181c   182c   183c
                                        191c   193c   197c   198c   201c   202c   210c   329c   347c   351c   472c
device_addr
  DGL_C_IN(dgl_confg_in)        ***    12111c
  DGL_C_OUT(dgl_confg_out)      ***    11480c
  GLE_HPIB(gle_hpib_io)               10012d 10493c
device_address
  SRM_DRV(srm)                          340d  1500c
device_address_present
  SRM_DRV(srm)                          339d  1499c
device_address_type
  SRM_DRV(srm)                          259d   340d
device_buf
  DGL_C_IN(dgl_confg_in)              12079c 12100c
  DGL_C_OUT(dgl_confg_out)            11393c 11425c
  GLE_GEN(gle_gen)              ***     2131c
  GLE_GENI(gle_geni)            ***     3029c
  GLE_HPGL(gle_hpgl_out)                7044c  7047c  7060c  7072c  7086c  7122c  7144c  7147c  7171c  7180c  7228c  7240c  7336c  7362c
                                        7387c  7405c  7434c  7472c  7491c  7519c  7573c  7581c
  GLE_HPGLI(gle_hpgl_in)              18032c 18034c 18045c 18059c 18083c 18085c 18102c 18106c 18139c 18146c 18249c 18253c 18263c 18295c
                                      18299c 18332c 18346c 18354c
  GLE_TYPES(gle_types)                 1071d  1221d
device_ident
  AMIGO(amigodvr)                       495d   519c
device_ident_type
  AMIGO(amigodvr)                       492d   495d
device_info
  DGL_C_IN(dgl_confg_in)              12049c 12097c 12111c
  DGL_C_OUT(dgl_confg_out)            11279c 11419c 11439c 11480c
  DGL_KNOB(dgl_knob)            ***   18075c
  GLE_GEN(gle_gen)              ***     2133c
  GLE_GENI(gle_geni)            ***     3031c
  GLE_TYPES(gle_types)                 1073d  1223d
  LIB(dgl_lib)                        21212c 21266c 21319c
device_info_char_count
  DGL_C_IN(dgl_confg_in)              12049c 12097c 12112c
  DGL_C_OUT(dgl_confg_out)            11279c 11419c 11440c 11481c
  DGL_KNOB(dgl_knob)            ***   18076c
  GLE_TYPES(gle_types)                 1074d  1224d
  LIB(dgl_lib)                        21213c 21265c 21318c
device_letter
  CTABLE(scanstuff)                    1101d  1105c  1106c
device_maps
  AMIGO(csamigo)                        195d   215c
device_maps_type
  AMIGO(csamigo)                        191d   195d
```

```
device_msus
  CTABLE                          1239d  1245c  1246c  1247c  1248c  1251c
device_name
  LIB(dgl_lib)                   20022d 21194d 21209c 21216c 21218c
device_table
  AMIGO(amigodvr)                  454d   484c   503d   534c
device_table_type
  AMIGO(amigodvr)                  446d   454d   493d   503d
device_work_area
  DGL_C_OUT(dgl_confg_out)       11157d 11280c 11281c 11373d 11394c 11405d 11426c 11427c
deviceaddress
  GLE_RGL(gle_ras_out)             8032d  8463c
devicename
  MINIT(qminit)                    936d  1010c  1011c  1014c  1018c  1019c  1021c  1027c  1029c
  MIUI                             131d   192c   193c   195c
  TAPEBKUP(cs80tbdvr)              455d   487c   488c   490c   494c   495c   497c   502c   504c
devicerec
  SYSDEVS(sysdevs)                 329d   346d
devicetype
  DGL_RAS(dgl_raster)            17186c 17442c 17447c 17452c 17457c 17463c 17477c 17486c 17512c 17517c 17522c 17532c
  GLE_RGL(gle_ras_out)       ***   8031d
devid
  AMIGO(amigodvr)            ***    686c
  AMIGO(csamigo)                   209c   318c   319c   321c
  CS80(cs80dsr)                    921c   938c
  CS80(cs80dvr)                    984d  1017d  1021c
  CTABLE(ctr)                ***    478c
  INIT(initunits)            ***   2211c
  INITLOAD                   ***   1684c
  INITLOAD(sysglobals)       ***    151d
  MINIT(hminit)              ***    278c
  PRINTER(prtdvr)            ***    277c
  TAPEBKUP                        1298c  1315c
devstate
  HPHIL(hphil)                     100c   101c   104c   105c   107c   110c   111c   114c   115c   116c   118c   122c   193c
  MOUSE(mouse)                      81c    84c    89c    90c    91c    92c    96c   101c   102c   104c   156c
  SYSDEVS(sysdevs)           ***    330d
devtype
  CS80(cs80dvr)                    984d  1017d  1020c
dfirstblk
  INITLOAD(loader)                 827d  1318c
  LIBRARIAN                       2728c  2730c  2857c  2861c  2890c  3155c
  M68KSYS(ci)                ***    774d
  UCSD_DAM(ucsdmodule)              94c   162c   228c   245c   343c   362c   365c   369c   371c   418c   480c   489c   547c   549c
                                   596c   598c   599c   601c
dfkind
  INITLOAD(loader)           ***    830d
  LIBRARIAN                       2893c  3156c
  M68KSYS(ci)                ***    775d
  UCSD_DAM(ucsdmodule)              95c   247c   291c   313c   484c   545c   546c   595c
dgl_autl
  LIB(dgl_lib)               ***  20102d
dgl_background_index
  DGL_HPGL(dgl_hpgl)         ***  17294c
  DGL_RAS(dgl_raster)              17229c 17236c 17237c 17240c
  DGL_VARS(dgl_vars)         ***   1139d
  LIB(dgl_lib)               ***  20550c
dgl_char_height
  DGL_INQ(dgl_inq)           ***   6186c
  DGL_VARS(dgl_vars)         ***   1073d
  LIB(dgl_lib)                    20719c 20831c 21150c 21152c 21435c
```

```
dgl_char_width
  DGL_INQ(dgl_inq)           ***   6185c
  DGL_VARS(dgl_vars)         ***   1071d
  LIB(dgl_lib)                    20719c 20830c 21148c 21152c 21434c
dgl_confg_in
  DGL_C_IN(dgl_confg_in)     ***  12002d
  LIB(dgl_lib)               ***  20108d
dgl_confg_out
  DGL_C_OUT(dgl_confg_out)   ***  11002d
  DGL_KNOB(dgl_knob)         ***  18016d
  LIB(dgl_lib)               ***  20107d
dgl_current_color
  DGL_INQ(dgl_inq)           ***   6354c
  DGL_POLY(dgl_poly)              20212c 20598c 20599c
  DGL_VARS(dgl_vars)         ***   1089d
  LIB(dgl_lib)                    20426c 20461c 20464c 21014c 21170c 21444c
dgl_current_color_model
  DGL_INQ(dgl_inq)                 6049c  6426c
  DGL_RAS(dgl_raster)             17209c 17266c 17619c 17620c 17645c
  DGL_VARS(dgl_vars)         ***   1100d
  LIB(dgl_lib)                    20405c 21462c
dgl_current_linestyle
  DGL_INQ(dgl_inq)           ***   6359c
  DGL_POLY(dgl_poly)              20213c 20944c 20947c
  DGL_VARS(dgl_vars)         ***   1090d
  LIB(dgl_lib)                    20482c 21445c
dgl_current_linewidth
  DGL_INQ(dgl_inq)           ***   6364c
  DGL_POLY(dgl_poly)              20214c 20945c 20949c
  DGL_VARS(dgl_vars)         ***   1091d
  LIB(dgl_lib)                    20441c 21446c
dgl_current_polygon_angle
  DGL_POLY(dgl_poly)              20158c 20889c 20992c
  DGL_VARS(dgl_vars)         ***   1096d
  LIB(dgl_lib)                    21168c 21458c
dgl_current_polygon_color
  DGL_INQ(dgl_inq)           ***   6380c
  DGL_POLY(dgl_poly)              20069c 20074c 20095c 20119c 20215c 20318d 20599c 20840c 20902c
  DGL_VARS(dgl_vars)         ***   1093d
  LIB(dgl_lib)                    21165c 21455c
dgl_current_polygon_crosshatch
  DGL_POLY(dgl_poly)              20156c 20321d 20595c 20832c 20890c 20989c
  DGL_VARS(dgl_vars)         ***   1099d
  LIB(dgl_lib)                    21162c 21460c
dgl_current_polygon_density
  DGL_POLY(dgl_poly)              20157c 20210c 20243c 20887c 20890c 20891c 20895c 20997c 21000c 21004c
  DGL_VARS(dgl_vars)         ***   1097d
  LIB(dgl_lib)                    21167c 21457c
dgl_current_polygon_edge
  DGL_POLY(dgl_poly)              20159c 21059c 21104c 21135c 21166c
  DGL_VARS(dgl_vars)         ***   1098d
  LIB(dgl_lib)                    21161c 21459c
dgl_current_polygon_linestyle
  DGL_INQ(dgl_inq)           ***   6433c
  DGL_POLY(dgl_poly)              20137c 20216c 20319d 20902c 20947c 20948c 21018c
  DGL_VARS(dgl_vars)         ***   1095d
  LIB(dgl_lib)                    21163c 21456c
dgl_current_polygon_spacing
  DGL_POLY(dgl_poly)              20322d 20596c
dgl_current_polygon_style
  DGL_INQ(dgl_inq)           ***   6385c
  DGL_POLY(dgl_poly)              20160c 20187c
  DGL_VARS(dgl_vars)         ***   1094d
  LIB(dgl_lib)                    21164c 21461c
```

```
dgl_current_timming_mode
  DGL_INQ(dgl_inq)              ***   6369c
  DGL_VARS(dgl_vars)            ***   1092d
  LIB(dgl_lib)                      20391c 21169c 21447c
dgl_cursor
  LIB(dgl_lib)                      20216d 21284c
dgl_gen
  DGL_HPGLI(dgl_hpgli)          ***  18015d
  DGL_INQ(dgl_inq)              ***   6031d
  DGL_KNOB(dgl_knob)            ***  18015d
  DGL_POLY(dgl_poly)            ***  20035d
  DGL_RAS(dgl_raster)           ***  17019d
  GEN(dgl_gen)                  ***   3006d
  LIB(dgl_lib)                  ***  20103d
dgl_hpgl
  DGL_C_OUT(dgl_confg_out)      ***  11020d
  DGL_HPGL(dgl_hpgl)            ***  17002d
dgl_hpgl_init
  DGL_C_OUT(dgl_confg_out)          11447c 11487c
  DGL_HPGL(dgl_hpgl)                17006d 17220d
dgl_hpgli
  DGL_C_IN(dgl_confg_in)        ***  12019d
  DGL_HPGLI(dgl_hpgli)          ***  18002d
dgl_hpgli_init
  DGL_C_IN(dgl_confg_in)        ***  12117c
  DGL_HPGLI(dgl_hpgli)              18006d 18123d
dgl_ibody
  LIB(dgl_lib)                  ***  20110d
dgl_inq
  DGL_INQ(dgl_inq)              ***   6002d
dgl_knob
  DGL_C_IN(dgl_confg_in)        ***  12018d
  DGL_KNOB(dgl_knob)            ***  18002d
dgl_knob_init
  DGL_C_IN(dgl_confg_in)        ***  12058c
  DGL_KNOB(dgl_knob)                18006d 18189d
dgl_lib
  DGL_POLY(dgl_poly)            ***  20038d
  LIB(dgl_lib)                  ***  20012d
dgl_poly
  DGL_POLY(dgl_poly)            ***  20008d
dgl_polygon_color_current
  DGL_POLY(dgl_poly)                20098c 20120c 20950c
  DGL_VARS(dgl_vars)            ***   1146d
  LIB(dgl_lib)                      20420c 21166c
dgl_raster
  DGL_C_OUT(dgl_confg_out)      ***  11019d
  DGL_RAS(dgl_raster)           ***  17002d
dgl_raster_init
  DGL_C_OUT(dgl_confg_out)      ***  11354c
  DGL_RAS(dgl_raster)               17006d 17544d
dgl_scaled_draw
  LIB(dgl_lib)                  ***  20665c
dgl_scaled_move
  LIB(dgl_lib)                  ***  20635c
dgl_tools
  DGL_C_IN(dgl_confg_in)        ***  12015d
  DGL_C_OUT(dgl_confg_out)      ***  11016d
  DGL_TOOLS(dgl_tools)          ***  20003d
  LIB(dgl_lib)                  ***  20109d
```

```
dgl_types
  DGL_HPGL(dgl_hpgl)            ***  17010d
  DGL_HPGLI(dgl_hpgli)          ***  18010d
  DGL_INQ(dgl_inq)              ***   6004d
  DGL_KNOB(dgl_knob)            ***  18010d
  DGL_POLY(dgl_poly)            ***  20010d
  DGL_RAS(dgl_raster)           ***  17010d
  DGL_VARS(dgl_vars)            ***     52d
  GEN(dgl_gen)                  ***   3008d
  LIB(dgl_lib)                  ***  20014d
  TYPES(dgl_types)              ***   1002d
dgl_vars
  DGL_C_IN(dgl_confg_in)        ***  12016d
  DGL_C_OUT(dgl_confg_out)      ***  11021d
  DGL_HPGL(dgl_hpgl)            ***  17011d
  DGL_HPGLI(dgl_hpgli)          ***  18011d
  DGL_INQ(dgl_inq)              ***   6029d
  DGL_KNOB(dgl_knob)            ***  18011d
  DGL_POLY(dgl_poly)            ***  20034d
  DGL_RAS(dgl_raster)           ***  17011d
  DGL_VARS(dgl_vars)            ***     50d
  GEN(dgl_gen)                  ***   3054d
  LIB(dgl_lib)                  ***  20101d
di
  LIFDAM(lifmodule)                  582d   594c   624c   625c
diag_unit
  MINIT(qminit)                     1102d  1124c  1129c  1136c  1141c
diagnostic_release_required
  CS80(cs80)                    ***   159d
  CS80(cs80dsr)                 ***   710c
  TAPEBKUP(cs80tbdvr)                592c   824c
diagnostic_request
  CS80(cs80)                         183d   312d   328d
  CS80(cs80dsr)                 ***   717c
  TAPEBKUP(cs80tbdvr)                599c   831c
diagnostic_result
  CS80(cs80)                         156d   299d
  CS80(cs80dsr)                 ***   670c
  TAPEBKUP(cs80tbdvr)                550c   782c
diagnostic_to_be_run
  MINIT(qminit)                      894d  1063c  1071c  1072c  1122c
digit
  GEN(dgl_gen)                      3151d  3157c  3162c  3163c
  GLE_UTLS(gle_utls)               21047d 21054c 21055c 21076d 21096c 21097c
digitize_enabled
  GLE_KNOB(gle_knob_in)            18019d 18153c 18188c 18199c 18269c
dindex
  LIFDAM(lifmodule)                  200d   300c   414c   418c   419c   424c   432c   459c   460c   637c   642c   656c   657c   664c
                                     723c   730c   731c   767c   768c   789c   790c   812c   825c   858c   865c   866c   878c   879c
                                     890c   932c   945c   997c   999c  1009c  1020c  1050c  1053c  1070c  1095c
dir
  LIFDAM(lifmodule)                   79d   246c   251c   285c   300c   315c   317c   319c   352c   365c   367c   375c   407c   418c
                                     457c   466c   528c   530c   535c   594c   597c   637c   639c   664c   695c   698c   708c   709c
                                     713c   866c   876c   945c   999c  1009c  1032c  1050c  1053c  1076c  1078c  1095c  1122c  1144c
                                    1182c
  SRM_DRV(srm)                      1298d  1309c  1325c  1346c
  UCSD_DAM(ucsdmodule)                53d    60c    74c    87c   145c   149c   153c   155c   215c   221c   225c   226c   256c   257c
                                     260c   272c   276c   277c   288c   293c   309c   317c   333c   339c   340c   360c   362c   373c
                                     389c   390c   394c   411c   421c   451c   453c   468c   477c   497c   498c   500c   503c   504c
                                     516c   538c   590c   613c   617c   621c   623c   630c   657c   675c   676c
dir_is_dvrtemp
  SRMDAM(srmdammodule)              385d   426c
dir_len
  MIUI                              400d   435d
```

```
dir_of_ifr
  DISCHPIB(dischpib)          ***    443d
  IOLIB(general_4)                  1201d  1207d  1213d  1220d  1225d  1353d  1438d  1462d  1488d  1531d
  KERNEL(iodeclarations)             564d   577d
  SRM_DRV(srm)                ***   1298d
dir_se
  MIUI                               397d   432d
dircatentry
  ETU                                353d   369c   376c   867d   918c
  MIUI                               375d   516c   528c
  SRMDAM(srmdammodule)               381d   393c   449d   451c   452c
direct_control
  DC_DRV(extdc)               ***    207d
  DC_DRV(intdc)               ***    374c
direct_status
  DC_DRV(extdc)               ***    205d
  DC_DRV(init_dc)             ***    678c
  DC_DRV(intdc)                      273c   351c   389c
direction
  DC_DRV(intdc)                      426c   440c   530c   548c
  DISCHPIB(dischpib)                 474c   517c   528c   538c
  IOLIB(general_4)                  1424c  1570c  1597c
  KERNEL(iodeclarations)      ***    577d
  SRM_DRV(srm)                ***   1346c
directory
  INITLOAD(loader)                  1030d  1258c  1264c  1283c  1321c  1323c  1325c  1332c  1376c  1408c  1439c  1441c  1521c  1523c
                                    1580c
  LIBRARIAN                          323c   326c   395c  1275c  1277c  1383c  1681c  1694c  1936c  1938c  2339c  2610c  2612c  2749c
                                    2752c
directory_password
  SRM_DRV(srm)                       457d  1691c
directory_records
  SRMDAM(srmdammodule)              1008c  1130c
  SRM_DRV(srm)                ***    159d
directorysize
  INITLOAD(loader)                   946d  1326c  1328c  1330c  1332c
  LIBRARIAN                          352c  1965c  1980c
directsize
  INITLOAD(loader)                  1137d  1146c  1147c  1148c  1149c  1154c  1159c
direntry
  INITLOAD(bootdammodule)            557d   558d
  INITLOAD(loader)                   826d  1033d  1146c
  LIBRARIAN                   ***   2722d
  LIFDAM(lifmodule)                   59d    77d   201d   362d   685d
  M68KSYS(ci)                 ***    773d
  UCSD_DAM(ucsdmodule)                50d    67d   102c   159d   187d   381d   483c
direntryp
  INITLOAD(bootdammodule)            557d   585d   613d
dirfile
  LIFDAM(lifmodule)                   77d    79d
  UCSD_DAM(ucsdmodule)                50d    53d
dirfilecode
  SRMDAM(srmdammodule)               991d  1008c  1016d  1130c
dirid
  SRMDAM(srmdammodule)               382d   419c   421c   427c
dirname
  CTABLE                            1229d  1234c
  INIT(fs)                    ***    760d
diropen
  ETU                                 58d   337c   342c   371c   372c   379c   886c   919c  1228c
dirrange
  INITLOAD(loader)                   822d   834d
  UCSD_DAM(ucsdmodule)               138d   264d   265d   284d   302d   323d   325d   353d   379d   380d   398d   405d   429d   438d
                                     569d
```

```
dirstart
  UCSD_DAM(ucsdmodule)                39d    80c   100c   330c   482c
dirstatus
  ETU                                 54d    65d
dirty
  HEAPT(hpm)                          81d   124d   189c   201c   265c
disable
  A804XDVR(a804xdvr)                 163d   165c
disable_auto_release
  CS80(cs80)                  ***    235d
  CS80(cs80dsr)               ***    882c
disable_report_changes
  TAPEBKUP                          1031d  1393c
disableuserisrs
  M68KSYS(ci)                        292d   896c  1196c
disassemble
  LIBRARIAN                         1412d  1460c  1461c  1462c  1463c
dischpib
  AMIGO(amigodvr)             ***    425d
  AMIGO(csamigo)              ***     37d
  CS80(cs80)                  ***     66d
  CS80(cs80dvr)               ***    972d
  DISCHPIB(dischpib)          ***    186d
  MINIT(cs80ir)               ***    516d
  MINIT(hminit)               ***    248d
  MINIT(iramigo)              ***    187d
  MINIT(qminit)               ***    878d
  TAPEBKUP(cs80tbdvr)         ***    305d
  TAPEBKUP(cs80tbr)           ***     14d
discid
  LIFDAM(lifmodule)                   43d   253c   497c
discint_drivers
  DI_DRV(init_discint)               192d   217c   218c   245c
discint_initialize
  DI_DRV                      ***    128d
discunit
  MINIT(xminit)                     1366d  1585c  1628c  1722c  1733c  1757c  1798c
disp_dev_adr
  DGL_INQ(dgl_inq)            ***   6456c
  DGL_VARS(dgl_vars)          ***   1109d
  LIB(dgl_lib)                     21036c 21137c 21138c 21231c 21278c 21349c 21350c 21409c
disp_eq_loc
  DGL_HPGLI(dgl_hpgli)        ***  18087c
  DGL_VARS(dgl_vars)          ***   1114d
  GEN(dgl_gen)                ***   3581c
  LIB(dgl_lib)                     20899c 21043c 21046c 21078c 21137c 21348c 21360c 21408c
disp_file_name
  DGL_INQ(dgl_inq)                  6447c  6449c  6450c  6451c
  DGL_VARS(dgl_vars)          ***   1110s
  LIB(dgl_lib)                     21035c 21232c 21279c 21410c
disp_init
  DGL_HPGLI(dgl_hpgli)        ***  18051c
  DGL_INQ(dgl_inq)                  6196c  6211c  6318c  6324c  6330c  6336c  6342c  6348c  6374c  6390c  6396c  6402c  6408c  6414c
                                    6420c  6437c  6498c
  DGL_KNOB(dgl_knob)          ***  18052c
  DGL_VARS(dgl_vars)          ***   1241d
  GEN(dgl_gen)                      3086c  3415c  3535c
  LIB(dgl_lib)                     20392c 20562c 20592c 20621c 20652c 20714c 20942c 20970c 20998c 21034c 21078c 21098c 21136c 21207c
                                   21259c 21312c 21348c 21382c
```

```
disp_just
  DGL_HPGL(dgl_hpgl)          ***  17263c
  DGL_INQ(dgl_inq)            ***   6311c
  DGL_RAS(dgl_raster)         ***  17594c
  DGL_VARS(dgl_vars)          ***   1079d
  GEN(dgl_gen)                      3488c  3512c
  LIB(dgl_lib)                ***  21449c
displ
  LIBRARIAN                          432d  1058c  1062c  1068c
display
  CTABLE(scanstuff)                 1087d  1125c
display_draw
  LIB(dgl_lib)                      20134d 20166c 20169c 20186c 20188c 20196c 20203c 20204c 20205c 20206c
display_echo_mult
  DGL_HPGL(dgl_hpgl)          ***  17287c
  DGL_KNOB(dgl_knob)          ***  18100c
  DGL_RAS(dgl_raster)         ***  17657c
  DGL_VARS(dgl_vars)          ***   1326d
  LIB(dgl_lib)                ***  21450c
display_finit
  LIB(dgl_lib)                      20021d 21193d
display_handler_char_count
  GLE_GEN(gle_gen)            ***   2129c
  GLE_HPGL(gle_hpgl_out)      ***   7689c
  GLE_RGL(gle_ras_out)        ***   8501c
  GLE_TYPES(gle_types)        ***   1081d
display_handler_name
  GLE_HPGL(gle_hpgl_out)      ***   7688c
  GLE_RGL(gle_ras_out)        ***   8500c
  GLE_TYPES(gle_types)        ***   1080d
display_init
  LIB(dgl_lib)                      20025d 21245d
display_limits
  GEN(dgl_gen)                       3023d  3603d
  LIB(dgl_lib)                      20522c 21041c 21144c 21420c
display_max_x
  DGL_KNOB(dgl_knob)          ***  18082c
  DGL_POLY(dgl_poly)          ***  20620c
  DGL_RAS(dgl_raster)         ***  17331c
  GLE_HPGL(gle_hpgl_out)            7206c  7681c
  GLE_RGL(gle_ras_out)              8505c  8509c
  GLE_TYPES(gle_types)        ***   1087d
  LIB(dgl_lib)                ***  21124c
display_max_y
  DGL_KNOB(dgl_knob)          ***  18084c
  DGL_POLY(dgl_poly)                20613c 20628c
  DGL_RAS(dgl_raster)         ***  17335c
  GLE_HPGL(gle_hpgl_out)            7208c  7682c
  GLE_RGL(gle_ras_out)              8506c  8510c
  GLE_TYPES(gle_types)        ***   1088d
  LIB(dgl_lib)                ***  21126c
display_min_x
  DGL_RAS(dgl_raster)         ***  17331c
  GLE_HPGL(gle_hpgl_out)            7205c  7679c
  GLE_RGL(gle_ras_out)        ***   8503c
  GLE_TYPES(gle_types)        ***   1085d
  LIB(dgl_lib)                ***  21123c
display_min_y
  DGL_RAS(dgl_raster)         ***  17335c
  GLE_HPGL(gle_hpgl_out)            7207c  7680c
  GLE_RGL(gle_ras_out)        ***   8504c
  GLE_TYPES(gle_types)        ***   1086d
  LIB(dgl_lib)                ***  21125c
```

```
display_move
  LIB(dgl_lib)                      20123d 20165c 20168c 20185c 20187c 20189c 20195c 20202c
display_name
  DGL_HPGL(dgl_hpgl)                17035c 17036c 17037c 17038c 17039c 17040c 17123c 17137c 17138c 17146c 17147c 17148c 17149c 17150c
                                    17151c 17157c 17158c
  DGL_INQ(dgl_inq)            ***   6514c
  DGL_RAS(dgl_raster)               17586c 17590c
  GLE_HPGL(gle_hpgl_out)             7232c  7233c  7234c  7253c  7254c  7271c  7272c  7280c  7281c  7289c  7297c  7583c  7584c  7590c
                                     7616c  7619c  7620c  7627c  7634c  7635c  7636c  7643c  7644c  7645c  7646c  7647c  7648c  7649c
                                     7650c  7651c  7652c  7660c  7661c  7662c  7663c  7666c  7667c
  GLE_RGL(gle_ras_out)               8433c  8440c  8447c  8454c  8461c  8469c
  GLE_TYPES(gle_types)        ***   1078d
  LIB(dgl_lib)                      21215c 21218c
display_name_char_count
  DGL_INQ(dgl_inq)                   6511c  6513c  6515c
  GLE_GEN(gle_gen)            ***   2128c
  GLE_HPGL(gle_hpgl_out)             7585c  7591c  7619c  7620c  7627c  7634c  7635c  7636c  7643c  7644c  7645c  7646c  7647c  7648c
                                     7649c  7650c  7651c  7652c  7660c  7661c  7662c  7663c  7666c  7667c
  GLE_RGL(gle_ras_out)               8434c  8441c  8448c  8455c  8462c  8470c
  GLE_TYPES(gle_types)        ***   1079d
  LIB(dgl_lib)                      21216c 21217c
display_res_x
  DGL_INQ(dgl_inq)                   6200c  6216c
  DGL_RAS(dgl_raster)         ***  17469c
  GLE_HPGL(gle_hpgl_out)      ***   7691c
  GLE_RGL(gle_ras_out)               8435c  8442c  8449c  8456c  8464c  8471c
  GLE_TYPES(gle_types)        ***   1083d
  LIB(dgl_lib)                      20258c 20511c 20512c 21181c
display_res_y
  DGL_INQ(dgl_inq)                   6201c  6217c
  DGL_RAS(dgl_raster)         ***  17470c
  GLE_HPGL(gle_hpgl_out)      ***   7692c
  GLE_RGL(gle_ras_out)               8436c  8443c  8450c  8457c  8465c  8472c
  GLE_TYPES(gle_types)        ***   1084d
  LIB(dgl_lib)                      20259c 20514c 20515c
display_term
  LIB(dgl_lib)                      20028d 21024d 21098c 21207c 21259c
disptime
  M68KSYS(ci)                        108d   258c   269c  1097c
dither_pattern
  GLE_RGL(gle_ras_out)               8046d  8205c  8206c  8214c
dither_support
  DGL_POLY(dgl_poly)          ***  20067c
  GLE_HPGL(gle_hpgl_out)      ***   7705c
  GLE_TYPES(gle_types)        ***   1100d
dither_type
  GLE_RGL(gle_ras_out)               8014d  8046d
division_code
  SRM_DRV(srm)                ***    218d
dkvid
  CTABLE                      ***   1847c
  ETU                                877c   880c
  INIT(ldr)                   ***   2318c
  INITLOAD(sysglobals)        ***    266d
  M68KSYS(ci)                        195c   679c   688c   707c
dl
  INITLOAD(bootdammodule)     ***    586d
dlast
  LIFDAM(lifmodule)                  200d   283c   368c   379c   419c   460c   463c   625c   642c   657c   693c   879c   890c  1029c
```

```
doblankcheck
  ETU                                1155d  1253c
docat
  LIFDAM(lifmodule)                   580d  1152c
dochar
  LIBRARIAN                          1373d  1400c  1404c
doclockops
  CLOCK(clock)                        163d   177c
doconfigure
  ETU                                 709d   872c  1173c  1236c  1251c
docrtio
  CRT(crt)                            402d   702c
  GCRT(crtb)                          272d   516c
doedit
  LIBRARIAN                          3297d  3631c
doinit
  NONUSKBD2                           123d   227c
doit
  A804XDVR(a804xdvr)                   92d    96c   100c   102c   103c   105c   106c   108c
  SYSDEVS(sysdevs)                     38d   403d
dokbdio
  KEYS(keys)                           72d   582c
dokey
  A804XDVR(a804xdvr)                  326d   330c   331c   334c   340c
  KEYS(keys)                          355d   373d   422c
domakedirectory
  LIFDAM(lifmodule)                   482d  1155c
dominant
  GLE_RGL(gle_ras_out)               8077d  8239c  8282c
dominate_mode
  DGL_VARS(dgl_vars)          ***     1201d
done
  A804XDVR(a804xdvr)                   71d    74d
  DC_DRV(intdc)                       414d   457c   468c   480c   500c   515c   523c
  DGL_POLY(dgl_poly)                20557d 20567c 20581c 20587c
  EDRIVER(edriver)                     99d   118c   162c   169c   187c   188c   271c   273c   274c   287c   288c   289c
  EPROMS(eproms)                       41d    83c    92c    95c   100c   101c
  ETU                                 656d   665c   684c   689c   692c   711d   848d   950c   956c   969c  1025c  1044c  1056c  1060c
                                     1063c  1113c  1122c
  HPHIL(hphil)                         71d    96d   282d   302d   316c
  INIT(ldr)                          2324d  2334c  2343c  2347c
  INITLOAD                           1712d  1721c  1723c
  KEYS(keys)                          375d   382c   389c   395c   399c   403c   415c   425c   459c   474c   480c   499c   515c   519c
                                      525c   538c   550c   551c   561c
  LIBRARIAN                           152d   184c   198c   201c  1483d  1555c  1561c  1562c  1569c  1582c  1584c  1724d  1756c  1762c
                                     1770c  1771c  1773c  1776c  1785c  1792c  1794c  1795c  2005d  2041c  2047c  2058c  2059c  2061c
                                     2065c  2079c  2081c  2105c  2106c
  LIFDAM(lifmodule)                   583d   594c   595c   598c   625c   678d   748c   817c   836c   874c   879c   880c
  M68KSYS(ci)                         875d   882c   895c   908c  1033d  1165d  1201c  1202c  1203c
  MOUSE(mouse)                         75d   111c   119c   130d   134c   137c   162c   163c   169c   170c
  SRMDAM(srmdammodule)                465d   507d   518c   527c   535c   551c   573d   581c   586c   591c   596c   635c   668d   672c
                                      673c   675c
dontcare
  ETU                                  54d   254c   261c
doopendirectory
  LIFDAM(lifmodule)                   561d  1153c
doopenpack
  SRMDAM(srmdammodule)                900d   980c  1159c  1164c  1424c
dooropen
  F9885(f9885dvr)                      48d   219c
dooropened
  F9885(f9885dvr)                      62d   134c
dooverwritefile
  LIFDAM(lifmodule)                  1083d  1129c
```

```
doprefix
  CTABLE                      ***     1234c
  INIT(fs)                    ***      760d
  M68KSYS(ci)                          678c   679c   849c
doprompt
  ETU                                  709d   768c
dopurgename
  LIFDAM(lifmodule)                    668d  1148c
dor
  CS80(cs80)                  ***      220d
doread
  INIT(fs)                    ***      790d
doreconfigure
  HPHIL(hphil)                ***       58d
dorecover
  SRM_DRV(srm)                        1356d  1395c  1473c
doreset
  MOUSE(mouse)                         67d   175c
dosetsysdate
  CLOCK(clock)                        100d   168c
dosetsystime
  CLOCK(clock)                        114d   169c
dosysdate
  CLOCK(clock)                         49d   166c
dosystime
  CLOCK(clock)                         73d   167c
dotransfer
  ETU                                 834d  1249c
double
  KERNEL(general_0)                   968d   979c  1023c  1026c  1092c  1098c
doupdatecursor
  CRT(crt)                            269d   289c   311c   494c   567c   650c   653c   701c
  GCRT(crtb)                          196d   220c   362c   513c
down
  A804XDVR(a804xdvr)          ***      49d
  CRT(crt)                             97d   143d
  GCRT(crtb)                  ***      76d
  SYSDEVS(sysdevs)            ***      93d
downchar
  CRT(crt)                    ***     468c
  GCRT(crtb)                  ***     339c
  INIT(misc)                  ***     165d
doy
  LIFDAM(lifmodule)                    115d   117c   118c
dp
  GLE_RGL(gle_ras_out)                8162d  8206c
dpt1
  GLE_RGL(gle_ras_out)                8149d  8150d  8162d  8163d  8164d  8165d  8166d  8167d  8168d  8169d  8170d  8171d  8172d  8173d
                                      8174d  8175d  8176d  8177d  8178d
dpt2
  GLE_RGL(gle_ras_out)                8150d  8162d
dr
  LIFDAM(lifmodule)                    585d   587c
draw
  DGL_POLY(dgl_poly)          ***    20825c
  GLE_GEN(gle_gen)            ***     2066c
  GLE_HPGL(gle_hpgl_out)      ***     7528c
  GLE_RGL(gle_ras_out)        ***     8397c
  GLE_SMARK(gle_smark)        ***     5165c
  GLE_TYPES(gle_types)        ***     1037d
  LIB(dgl_lib)                       20141c 20608c
draw_pg
  DGL_POLY(dgl_poly)                 20317d 20902c
```

```
draw_polygon
  DGL_POLY(dgl_poly)                20844d 21133c 21164c
drawing
  GLE_HPGL(gle_hpgl_out)             7021d  7104c  7111c  7365c  7368c
driv
  F9885(f9885dvr)                      60d   116d   126c   148c
drive_attention
  AMIGO(amigodvr)            ***      912c
  AMIGO(csamigo)             ***       84d
  MINIT(hminit)              ***      324c
driver
  MOUSE(mouse)                         61d   181c   183c   184c   192c
driver_name
  SRM_DRV(srm)                        337d  1497c
driver_state
  GLE_HPGL(gle_hpgl_out)             7026d  7048c  7102c  7103c  7104c  7108c  7109c  7339c  7365c  7524c
driver_state_def
  GLE_HPGL(gle_hpgl_out)             7021d  7026d  7094d
driverkey
  INITLOAD(bootdammodule)             576d   637c   639c   641c   664c   666c   677c   772c   774c
drivetype
  MINIT(xminit)                      1363d  1526c  1594c  1595c  1597c  1599c
drp
  INITLOAD(loader)                    909d  1258c  1264c  1283c  1323c  1325c  1376c  1408c  1439c  1521c  1580c
  LIBRARIAN                           323c   395c  1275c  1383c  1681c  1936c  1958c  2111c  2530c  2610c  2749c  2880c  2883c
drs_line
  IOLIB(serial_0)                    2248c  2268c  2311c  2331c  2383c  2431c
  KERNEL(iodeclarations)     ***      411d
drv
  F9885(f9885dvr)                      58d   164c
drv_misc
  BUBBLES(bubble)                      67c   130c   162c
  EDRIVER(edriver)           ***      142c
  KERNEL(iodeclarations)              503d   517d   531d
drv_table_type
  DC_DRV(init_dc)            ***      585d
  DI_DRV(init_discint)       ***      192d
  G_DRV(init_gpio)           ***      209d
  H_DRV(init_hpib)           ***      208d
  KERNEL(general_0)                   742d   743d
  KERNEL(iodeclarations)              443d   539d
  RS_DRV(init_rs)            ***      203d
drv_tmp_ptr
  DC_DRV(intdc)              ***      534c
  DISCHPIB(dischpib)         ***      466c
  IOLIB(general_4)                    1275c  1419c
  KERNEL(iodeclarations)     ***      571d
  SRM_DRV(srm)                        1201c  1342c
drvasm
  AMIGO(amigodvr)            ***      425d
  CS80(cs80dvr)              ***      972d
  DISCHPIB(dischpib)         ***      189d
drve
  F9885(f9885dvr)                      63d   131c   201c
ds
  CTABLE(ctr)                          323d   329d   330d   331d   332d   333d   334d   335d   336d   337d   338d   339d   506d   508c
                                       637d   639c   655d   658c   686d   694c   698d   707c   711d   716c   720d   733c   737d   752c
                                       756d   768c   772d   787c   791d   794c   795c   796c   797c   798c   805d   809c   812d   816c
  INITLOAD(bootdammodule)    ***      586d
ds9895_map
  AMIGO(csamigo)                       193d   211c
```

```
ds_type
  CTABLE                     ***     1268d
  CTABLE(ctr)                         291d   323d   329d   330d   331d   332d   333d   334d   335d   336d   337d   338d   339d   506d
                                      637d   655d   686d   698d   711d   720d   737d   756d   772d   791d   805d   812d
dsa
  HEAPI(hpm)                           76d    77d   122d   128c
dsap
  HEAPI(hpm)                           76d   129c   138c   196c   219c   265c   294c
dsize
  INITLOAD(bootdammodule)             584d   612d   638c   665c
  LIBRARIAN                  ***     2334d
  LIFDAM(lifmodule)                    48d   255c   283c   284c   294c   504c   505c   509c   567c   571c
dsj
  AMIGO(amigodvr)                     473c   477c   521c   528c   615c   802c   941c
  AMIGO(csamigo)                      145d   229d   236c
  MINIT(hminit)                       361c   368c   395c   400c   405c   426c   441c   449c   474c
dsj_byte
  AMIGO(csamigo)                      231d   235c   236c
dsj_sec
  AMIGO(csamigo)                      233d   235c
dsr_line
  IOLIB(serial_0)                    2388c  2436c
  KERNEL(iodeclarations)     ***      410d
dstart
  INITLOAD(bootdammodule)             584d   611d   638c   665c
  LIFDAM(lifmodule)                    45d   255c   284c   291c   503c   509c   567c   582d   593c   604c   622c
dstatus
  ETU                                  65d   254c   256c   261c   362c   896c
dt
  CS80(cs80)                 ***       99d
  CS80(cs80dsr)                       922c   941c
  CS80(cs80dvr)              ***     1059c
  MINIT(qminit)                      1013c  1063c  1074c  1080c  1162c
  TAPEBKUP                           1301c  1318c  1442c  1470c
  TAPEBKUP(cs80tbdvr)        ***      489c
dtid
  INITLOAD(loader)           ***      838d
  LIBRARIAN                          1286c  1324c  1352c  1386c  2726c  2727c  2728c  2894c  2895c  3039c  3067c  3068c  3258c  3259c
                                     3282c  3283c  3312c  3328c  3348c  3361c  3374c  3428c  3438c  3452c  3471c  3605c
  UCSD_DAM(ucsdmodule)                146c   248c   277c   314c   342c   414c   452c   540c   544c   592c
dtp
  UCSD_DAM(ucsdmodule)                 49d   654c   657c
dtr_line
  IOLIB(serial_0)                    2246c  2266c  2309c  2329c  2378c  2426c
  KERNEL(iodeclarations)     ***      413d
```

```
du
  AMIGO(csamigo)                ***    268c
  CS80(cs80)                           393c   451c   579c
  CS80(cs80dsr)                 ***    622c
  CTABLE                              1191d  1243d  1247c  1274c  1278c  1318c  1319c  1345c  1348c  1349c  1360c  1362c  1469c  1505c
                                      1540c  1554c  1574c  1596c  1600c  1623c
  CTABLE(brstuff)                      934d  1011c  1016c
  CTABLE(ctr)                          325d   329d   330d   331d   332d   333d   334d   335d   336d   337d   415c   475c   497c   655d
                                       657c   658c   668d   672c   673c   686d   691c   694c   698d   704c   707c   711d   715c   716c
                                       720d   726c   733c   737d   743c   752c   756d   766c   768c   772d   785c   787c   791d   794c
                                       795c   796c   797c   798c   882c
  CTABLE(options)                      114d   118d   121d   123d   125d   127d
  CTABLE(scanstuff)             ***   1116c
  F9885(f9885dvr)                      126c   131c   148c   164c   201c
  INIT(initunits)               ***   2208c
  INITLOAD                      ***   1681c
  INITLOAD(mini)                       489c   492c
  INITLOAD(sysglobals)          ***    148d
  MINIT(cs80ir)                 ***    585c
  MINIT(qminit)                 ***   1124c
  MIUI                                  36c   201c   350c   351c
  SRMDAM(srmdammodule)          ***    417c
  SRM_DRV(srm)                  ***   1502c
  TAPEBKUP                            1010c  1092c
  TAPEBKUP(cs80tbdvr)                  740c   742c
  TAPEBKUP(cs80tbr)                     50c   240c   245c
dumbuf
  SRM_DRV(srm)                        1125d  1193c  1217c  1304c  1310c  1312c  1321c  1322c  1329c  1331c  1337c  1339c  1351c  1388c
                                      1389c  1390c  1392c  1428c  1429c  1430c  1431c  1456c  1457c  1458c  1459c
dummy
  CRT(crt)                      ***    222d
  DGL_C_OUT(dgl_confg_out)            11087d 11097c 11098c 11120c 11121c
  DI_DRV(init_discint)          ***    209d
  DMA_DRV(init_dma)                    215d   234c
  EDRIVER(edriver)                      98d   166c   177c
  GCRT(crtb)                           493d   518c   519c   531d   541c   542c
  GLE_HPGL(gle_hpgl_out)              7052d  7542c  7543c  7549c  7550c  7551c  7552c  7553c  7554c  7555c  7556c  7557c  7558c
  G_DRV(init_gpio)              ***    227d
  H_DRV(init_hpib)              ***    225d
  INIT(misc)                           301d   359c
  IOLIB(serial_0)                     2234d  2253c  2254c  2255c  2274c  2275c  2276c  2297d  2316c  2317c  2318c  2337c  2338c  2339c
                                      2361d  2411c  2412c  2459c  2460c
  IOLIB(serial_3)                     2531d  2563c  2564c  2565c  2566c  2575c  2576c  2577c  2601d  2663c  2664c  2665c  2685d  2715c
                                      2716c  2717c  2739d  2781c  2782c  2783c
  KERNEL(general_0)                    814d   819d   824d   829d   834d   839d   844d   967d  1032c  1033c  1035c  1073c  1075c  1122c
                                      1138c  1146c  1147c
  LIBRARIAN                            437d   453d   822c
  MIUI                                 399d   434d
  MOREFSYS(mfs)                        670d   673c
dummy1
  DGL_C_OUT(dgl_confg_out)            11167d 11277c
  LIFDAM(lifmodule)                     46d   254c   498c
dummy2
  LIFDAM(lifmodule)                     47d   254c   498c
dummy3
  LIFDAM(lifmodule)                     50d   254c   498c
dummy4
  LIFDAM(lifmodule)                     57d   524c
dummy_byte
  MINIT(iramigo)                       211d   217c   218c
dummy_char
  DISCHPIB(dischpib)                   272d   283c   331d   342c
dummy_driver
  KERNEL(general_0)                    809d   816c   821c   826c   831c   836c   841c   846c  1179c  1180c  1281c

dummy_driver_a
  KERNEL(general_0)                    819d  1181c  1183c  1187c  1188c  1190c
dummy_driver_b
  KERNEL(general_0)                    839d  1191c  1192c
dummy_driver_ba
  KERNEL(general_0)                    844d  1193c
dummy_driver_c
  KERNEL(general_0)                    814d  1182c  1189c
dummy_driver_dw
  KERNEL(general_0)                    829d  1186c
dummy_driver_w
  KERNEL(general_0)                    824d  1184c
dummy_driver_wa
  KERNEL(general_0)                    834d  1185c
dummy_drivers
  DC_DRV(init_dc)               ***    613c
  DI_DRV(init_discint)          ***    218c
  G_DRV(init_gpio)              ***    236c
  H_DRV(init_hpib)              ***    234c
  KERNEL(general_0)                    743d  1178c  1202c  1231c
  RS_DRV(init_rs)               ***    229c
dummy_dsj
  AMIGO(amigodvr)                      508d   521c
dummy_isc
  DC_DRV(intdc)                        265d   269c   274c   282c   284c   285c   290c   293c
dummy_on_off
  DGL_RAS(dgl_raster)                 17286d 17587c
dummy_pr
  DC_DRV(intdc)                 ***    536c
  DMA_DRV(init_dma)                    185c   202c   222c   225c
  IOLIB(general_4)              ***   1277c
  KERNEL(general_0)                   1214c  1269c
  KERNEL(iodeclarations)        ***    468d
  SRM_DRV(srm)                  ***   1203c
dummy_sl
  DMA_DRV(init_dma)                    223c   226c
  IOLIB(general_4)              ***   1276c
  KERNEL(general_0)                   1213c  1268c
  KERNEL(iodeclarations)        ***    469d
  SRM_DRV(srm)                  ***   1202c
dummy_tfr_1
  KERNEL(iodeclarations)        ***    549d
dummy_tfr_2
  KERNEL(iodeclarations)        ***    553d
dummy_xxx
  GLE_TYPES(gle_types)                1062d  1212d
dummybatcmd
  SYSDEVS(sysdevs)                     445d   607c
dummybatread
  SYSDEVS(sysdevs)                     447d   606c
dummyboolproc
  SYSDEVS(sysdevs)                     410d   619c
dummyc
  PRINTER(prtdvr)                      153d   158c
dummycleario
  INIT                                2530c  2544c
dummyclockio
  SYSDEVS(sysdevs)                     455d   610c
dummyclockreq
  SYSDEVS(sysdevs)                     453d   609c
dummyclocksys
  SYSDEVS(sysdevs)              ***    451d
dummycmdchar
  M68KSYS(ci)                          145d  1176c
```

```
dummycrtll
  DGL_C_OUT(dgl_config_out)    ***  11186c
  SYSDEVS(sysdevs)                   378d   487d   629c
dummydataproc
  HPHIL(hphil)                        71d   195c   200c   286c   310c   326c   368c
dummydbcrt
  SYSDEVS(sysdevs)                   489d   630c
dummydebug
  INIT                              2529c  2543c
dummyhphilcmd
  SYSDEVS(sysdevs)                   593d   653c
dummykbd
  SYSDEVS(sysdevs)                   402d   604c   612c   615c   651c
dummykbdwait
  KEYS(keys)                         574d   579c
dummykey
  KEYS(keys)                         407d   411c   412c
dummyopsproc
  HPHIL(hphil)                        73d   194c
dummyout2
  SYSDEVS(sysdevs)                   400d   600c   601c
dummyproc
  SYSDEVS(sysdevs)                   398d   622c   623c   624c   625c   626c   627c   645c   646c
dummyregl
  SYSDEVS(sysdevs)                   405d   603c   614c
dummystatus56
  A804XDVR(a804xdvr)                  71d   405c   406c
dummytimerio
  SYSDEVS(sysdevs)                   477d   611c
dummytm
  SYSDEVS(sysdevs)                   407d   618c   628c
dummyword
  DC_DRV(init_dc)                    603d   678c   679c   680c   703c   704c
  DC_DRV(intdc)                      264d   273c   276c   277c   358d   389c   391c
  RS_DRV(init_rs)              ***   220d
dump_c
  C_HOOK                              60d   137c
dump_graphics
  DGL_RAS(dgl_raster)               17306d 17449c 17514c
dumpa
  CRT(crt)                           180d   697c
dumpalpha
  KEYS(keys)                         394d   446c   496c   510c
dumpalphahook
  CRT(crt)                     ***   697c
  GCRT(crtb)                   ***   511c
  KEYS(keys)                   ***   395c
  SYSDEVS(sysdevs)                   122d   624c
dumped
  LIBRARIAN                          304d   322c   402c  1440c
dumpg
  CRT(crt)                           216d   698c
  GCRT(crtb)                         134d   511c   512c
dumpgat
  DGL_RAS(dgl_raster)               17364d 17454c 17519c
dumpgraphics
  KEYS(keys)                         398d   445c   497c   512c
dumpgraphicshook
  CRT(crt)                     ***   698c
  C_HOOK                       ***   137c
  DGL_RAS(dgl_raster)          ***  17444c
  GCRT(crtb)                   ***   512c
  KEYS(keys)                   ***   399c
  SYSDEVS(sysdevs)                   123d   625c
```

```
dumpmod
  LIBRARIAN                          316d   402c
dumprefs
  LIBRARIAN                         2266d  2294c  2327c
dumptext
  LIBRARIAN                         2227d  2254c  2548c  2599c
dumwindow
  ETU                                864d   888c   923c
dup
  UCSD_DAM(ucsdmodule)               353d   357c   398d   400c
dupindex
  UCSD_DAM(ucsdmodule)               323d   328c   342c   344c
dupinx
  UCSD_DAM(ucsdmodule)               405d   408c   423c
duplicatelink
  INITLOAD(sysglobals)         ***   137d
  SRMDAM(srmdammodule)              1660c  1766c
duration
  A804XDVR(a804xdvr)                 157d   160c
  SYSDEVS(sysdevs)                   367d   416d   417c
dutch_kbd
  A804XDVR(a804xdvr)           ***   188d
  NONUSKBD2                          198c   212c   214c
  SYSDEVS(sysdevs)             ***   147d
dv
  AMIGO(csamigo)                     351c   372c
  CS80(cs80)                         451c   580c
  CS80(cs80dsr)                ***   936c
  CTABLE                            1191d  1243d  1248c  1334c  1363c  1470c  1554c  1560c  1561c  1574c  1578c  1579c  1596c  1600c
  CTABLE(brstuff)                    934d  1012c  1017c
  CTABLE(ctr)                        334d   336d   337d   416c   476c   497c   737d   744c   747c   752c   756d   767c   768c   772d
                                     786c   787c   882c
  CTABLE(options)                    114d   118d   121d   123d   125d   127d
  CTABLE(scanstuff)            ***  1117c
  EPROMS(eproms)               ***    82c
  INIT(initunits)              ***  2209c
  INITLOAD                     ***  1682c
  INITLOAD(sysglobals)         ***   149d
  MINIT(cs80ir)                ***   586c
  MINIT(qminit)                ***  1017c
  MIUI                               36c   178c   202c
  TAPEBKUP                          1010c  1092c
  TAPEBKUP(cs80tbdvr)                493c   740c   742c
  TAPEBKUP(cs80tbr)                   51c   239c   244c
dvid
  INITLOAD(loader)             ***   832d
  LIBRARIAN                         3157c  3158c
  M68KSYS(ci)                  ***   776d
  UCSD_DAM(ucsdmodule)                96c   107c   110c   467c   487c   515c
```

```
dvrtemp
  AMIGO(amigodvr)                598c   634c   646c   648c   675c   702c   720c   766c   777c   799c   938c
  CS80(cs80dvr)                 1090c  1101c  1134c  1231c  1342c  1367c  1368c  1382c  1387c
  CTABLE(ctr)            ***     480c
  DISCHPIB(bkgnd)                115c   129c   130c   133c   138c   149c   157c   158c   165c   170c
  DISCHPIB(dischpib)             447c   497c   508c   526c
  INIT(initunits)        ***    2213c
  INITLOAD               ***    1686c
  INITLOAD(sysglobals)   ***     153d
  MINIT(cs80ir)                  759c   806c
  MINIT(hminit)                  308c   500c   501c
  MINIT(qminit)                  984c   996c  1133c  1276c  1277c
  PRINTER(prtdvr)        ***     276c
  SRMDAM(srmdammodule)           337c   451c   744c   867c   890c   891c
  TAPEBKUP(cs80tbdvr)            357c   358c   735c
  TAPEBKUP(cs80tbr)      ***     284c
dvrtemp2
  CS80(cs80dsr)                  627c   925c   929c   933c
  CS80(cs80dvr)                 1192c  1228c  1236c
  CTABLE(ctr)            ***     481c
  INITLOAD               ***    1687c
  INITLOAD(sysglobals)   ***     154d
dwanted
  ETU                             54d   258c   896c
dx
  DGL_HPGLI(dgl_hpgli)          18023d 18029c 18030c 18045d 18080c 18083c 18085c 18090c 18101c 18102c
  DGL_KNOB(dgl_knob)            18045d 18112c 18115c 18117c 18122c 18135c 18140c 18142c 18145c
  DGL_POLY(dgl_poly)            20417d 20452c 20457c 20614c 20620c 20621c 20629c 20746c
  GEN(dgl_gen)                   3027d  3030d  3033d  3035d  3275d  3286c  3296c  3303d  3308c  3314d  3321c  3328d  3335c
  GLE_SMARK(gle_smark)           5098d  5158c  5160c
  GLE_STEXT(gle_stext)           4027d  4037c  4038c  4039c  4046c  4047c  4048c
  LIB(dgl_lib)                  20070d 20145d 20152c 20154c 20157c 20172d 20180c 20187c 20188c 20189c 20194c 20196c 20197c 20204c
                                20205c 20207c 20249d 20254c 20258c 20266d 20272c 20277c 20785d 20799c 20803c 20804c
dxmax
  GEN(dgl_gen)                   3023d  3603d  3616c  3623c
  LIB(dgl_lib)                  20052d 20488d 20505c 20512c
dxmin
  GEN(dgl_gen)                   3023d  3603d  3615c  3623c
  LIB(dgl_lib)                  20052d 20488d 20505c 20511c
dxunits
  DGL_VARS(dgl_vars)     ***     1081d
  GEN(dgl_gen)                   3493c  3504c  3517c  3623c  3628c
dy
  CLOCK(clock)                    101d   106c   109c
  DGL_HPGLI(dgl_hpgli)          18023d 18029c 18030c 18045d 18080c 18083c 18086c 18091c 18101c 18102c
  DGL_KNOB(dgl_knob)            18045d 18113c 18115c 18118c 18123c 18136c 18141c 18142c 18145c
  DGL_POLY(dgl_poly)            20417d 20452c 20457c 20613c 20621c 20628c 20629c 20746c
  GEN(dgl_gen)                   3027d  3030d  3033d  3035d  3275d  3287c  3297c  3303d  3309c  3314d  3323c  3328d  3336c
  GLE_SMARK(gle_smark)           5099d  5159c  5161c
  GLE_STEXT(gle_stext)           4027d  4041c  4042c  4043c  4050c  4051c  4052c
  LIB(dgl_lib)                  20070d 20145d 20151c 20154c 20155c 20172d 20180c 20185c 20186c 20189c 20194c 20196c 20197c 20203c
                                20204c 20207c 20249d 20255c 20259c 20266d 20273c 20278c 20785d 20799c 20803c 20805c
dyes
  DGL_VARS(dgl_vars)     ***     1347d
dymax
  GEN(dgl_gen)                   3024d  3603d  3618c  3624c
  LIB(dgl_lib)                  20053d 20488d 20505c 20515c
dymin
  GEN(dgl_gen)                   3024d  3603d  3617c  3624c
  LIB(dgl_lib)                  20053d 20488d 20505c 20514c
dyunits
  DGL_VARS(dgl_vars)     ***     1082d
  GEN(dgl_gen)                   3493c  3517c  3528c  3624c  3628c
```

```
e
  AMIGO(amigodvr)                 889c   914c
  AMIGO(csamigo)         ***      111d
  M68KSYS(ci)                     656d   701c   702c   703c
e_rcrd
  AMIGO(amigodvr)                 796d   818c   819c   827c   830c   839c   852c   866c   917c   918c   921c
eamode
  LIBRARIAN                       435d   629c   659c   696c   724c   725c   840c   885c   891c   900c   929c   932c   933c   939c
                                  943c   961c   963c  1011c  1013c  1031c  1101c  1105c  1125c  1133c
eareg
  LIBRARIAN                       436d   607c   630c   631c   632c   633c   634c   635c   637c   664c   669c   696c   724c   725c
                                  842c   886c   894c   932c   933c   939c   943c   954c   957c   962c   966c   967c   975c   980c
                                 1019c  1036c  1108c  1126c  1128c  1130c  1134c  1136c
eb0
  CS80(cs80)                      131d   284d
  CS80(cs80dsr)          ***      657c
  TAPEBKUP(cs80tbdvr)             537c   765c
eb1
  CS80(cs80)                      132d   285d
eb11
  CS80(cs80)                      142d   288d
eb13
  CS80(cs80)                      144d   289d
eb14
  CS80(cs80)                      145d   290d
eb15
  CS80(cs80)                      146d   291d
eb16
  CS80(cs80)                      148d   293d
eb18
  CS80(cs80)                      150d   295d
eb20
  CS80(cs80)                      152d   296d
eb21
  CS80(cs80)                      153d   297d
eb23
  CS80(cs80)                      155d   298d
eb25
  CS80(cs80)                      157d   300d
eb29
  CS80(cs80)                      161d   301d
eb3
  CS80(cs80)                      134d   286d
eb38
  CS80(cs80)                      171d   303d
eb39
  CS80(cs80)                      172d   304d
eb4
  CS80(cs80)                      135d   287d
eb42
  CS80(cs80)                      175d   306d
eb45
  CS80(cs80)                      178d   307d
eb46
  CS80(cs80)                      179d   308d
eb47
  CS80(cs80)                      180d   309d
eb53
  CS80(cs80)                      187d   314d
  CS80(cs80dsr)          ***      724c
  TAPEBKUP(cs80tbdvr)             606c   838c
```

```
eb54
  CS80(cs80)                          188d   315d
  CS80(cs80dsr)              ***      725c
  TAPEBKUP(cs80tbdv*)                 607c   839c
eb56
  CS80(cs80)                          190d   316d
  CS80(cs80dsr)              ***      727c
  TAPEBKUP(cs80tbdv*)                 609c   841c
eb60
  CS80(cs80)                          194d   319d
  CS80(cs80dsr)              ***      731c
  TAPEBKUP(cs80tbdv*)                 613c   845c
eb62
  CS80(cs80)                          196d   320d
  CS80(cs80dsr)              ***      733c
  TAPEBKUP(cs80tbdv*)                 615c   847c
eb63
  CS80(cs80)                          197d   321d
  CS80(cs80dsr)                       657c   734c
  TAPEBKUP(cs80tbdv*)                 537c   616c   765c   848c
eb_scan
  CS80(cs80dsr)                       641d   657c   658c   661c   662c   664c
  TAPEBKUP(cs80tbdv*)                 521d   537c   538c   541c   542c   544c   701d   765c   766c   769c   770c   772c
ebadargs
  EDRIVER(edriver)                     34d   116c
eblank
  EDRIVER(edriver)                     33d   319c
ebrate
  EDRIVER(edriver)                     47d   334d   340c
  ETU                                 591c  1071c
eccode
  EDRIVER(edriver)                    105d   107c
ecfail
  EDRIVER(edriver)                     34d   291c
  ETU                                 582c   595c
echeck
  EDRIVER(edriver)                     33d   320c
  ETU                        ***      572c
echo
  DGL_HPGLI(dgl_hpgli)               18017d 18031c 18036d 18049c 18051c 18054c 18061c 18069c 18087c 18105c 18112c 18114c
  DGL_KNOB(dgl_knob)                 18024d 18032c 18036d 18050c 18052c 18055c 18062c 18064c 18070c 18119c 18133c 18151c 18158c 18160c
  DGL_VARS(dgl_vars)                  1141d  1144d
  KEYS(keys)                           456c   467c
  LIB(dgl_lib)                       20076d 20080d 20934d 20943c 20951d 20996c 21017c
  PRINTER(prtdvr)                      157c   181c   184c
  SYSDEVS(sysdevs)                     238d   528c   533c   542c   546c   560c   564c   569c   638c
echo_cursor
  LIB(dgl_lib)                       20172d 20232c 20234c
echo_gcb
  DGL_KNOB(dgl_knob)                 18047d 18068c 18091c 18120c 18125c 18152c 18155c
echo_mult
  GLE_KNOB(gle_knob_in)              18017d 18171c 18189c
echo_rate
  GLE_KNOB(gle_knob_in)              18016d 18137c 18138c 18139c 18140c 18171c 18270c
echoerror
  DGL_HPGLI(dgl_hpgli)               18044d 18053c 18057c 18119c
  DGL_KNOB(dgl_knob)                 18044d 18054c 18058c 18165c
ecode
  EDRIVER(edriver)                     79d   107c   128c   313d   322c   329d   331c   337d   340c   346d   348c
  ETU                                 639d   641c   642c
  INIT(ldr)                          2408d  2489c  2491c  2492c  2504c
  INITLOAD(bootdammodule)             616d   620c   621c   622c
eci_type
  INITLOAD(mini)                      337d   341d
```

```
ecwrite
  EDRIVER(edriver)                     33d   318c
  ETU                                 578c   593c
edefs
  INITLOAD(loader)                   1067d  1127c  1133c
edge
  DGL_HPGL(dgl_hpgl)                 17227d 17228d 17229d 17230d 17231d 17232d 17233d 17234d 17235d 17236d 17237d 17238d 17239d 17240d
                                     17241d 17242d
  DGL_INQ(dgl_inq)           ***      6078c
  DGL_POLY(dgl_poly)                 20159c 20186c
  DGL_RAS(dgl_raster)                17562d 17563d 17564d 17565d 17566d 17567d 17568d 17569d 17570d 17571d 17572d 17573d 17574d 17575d
                                     17576d 17577d
  DGL_VARS(dgl_vars)         ***      1035d
edge_a
  DGL_POLY(dgl_poly)                 20513d 20532c
edge_b
  DGL_POLY(dgl_poly)                 20514d 20526c
edge_index
  DGL_POLY(dgl_poly)                 20390d 20484c 20485c 20503c 20504c 20536c 20539c 20542c 20673c 20680c 20684c 20726c 20736c
edge_polygon
  DGL_POLY(dgl_poly)                 20192d 21060c 21105c 21136c 21167c
edge_vertex
  DGL_POLY(dgl_poly)                 20388d 20484c 20485c 20536c 20673c 20680c 20684c
edheader
  ETU                                 865d  1075c  1076c  1077c  1078c  1079c  1080c  1081c  1082c
edi_clr
  DI_DRV(extdi)              ***      168d
  DI_DRV(init_discint)       ***      232c
edi_enc
  DI_DRV(extdi)              ***      172d
  DI_DRV(init_discint)       ***      227c
edi_init
  DI_DRV(extdi)              ***      155d
  DI_DRV(init_discint)       ***      219c
edi_isr
  DI_DRV(extdi)              ***      156d
  DI_DRV(init_discint)       ***      220c
edi_ppoll
  DI_DRV(extdi)              ***      167d
  DI_DRV(init_discint)       ***      230c
edi_rdt
  DI_DRV(extdi)              ***      157d
  DI_DRV(init_discint)       ***      221c
edi_rds
  DI_DRV(extdi)              ***      161d
  DI_DRV(init_discint)       ***      225c
edi_rdw
  DI_DRV(extdi)              ***      159d
  DI_DRV(init_discint)       ***      223c
edi_send
  DI_DRV(extdi)              ***      166d
  DI_DRV(init_discint)       ***      229c
edi_set
  DI_DRV(extdi)              ***      169d
  DI_DRV(init_discint)       ***      231c
edi_test
  DI_DRV(extdi)              ***      170d
  DI_DRV(init_discint)       ***      233c
edi_tfr
  DI_DRV(extdi)              ***      165d
  DI_DRV(init_discint)       ***      228c
edi_wtb
  DI_DRV(extdi)              ***      158d
  DI_DRV(init_discint)       ***      222c
```

```
edi_wtc
  DI_DRV(extdi)                 ***     163d
  DI_DRV(init_discint)          ***     226c
edi_wtw
  DI_DRV(extdi)                 ***     160d
  DI_DRV(init_discint)          ***     224c
edit
  M68KSYS(ci)                           661d   716c   717c   718c   720c   721c   722c
editor
  M68KSYS(ci)                            39d   720c  1012c  1118c
edriver
  EDRIVER(edriver)              ***      23d
  ETU                           ***      26d
efttable
  ASCII                         ***     321c
  ETU                                   320c   903c
  INIT(misc)                            707c   710c   715c   718c   723c   727c
  INITLOAD                             1706c  1707c  1708c
  INITLOAD(bootdammodule)               670c   709c
  INITLOAD(sysglobals)          ***     264d
  LIFDAM(lifmodule)                     421c   543c   544c   960c  1004c
  SRMDAM(srmdammodule)          ***      92c
  UCSD_AM(ucsd_am)              ***     272c
  UCSD_DAM(ucsdmodule)                  212c   291c   546c
efttableptrtype
  INITLOAD(sysglobals)                  175d   264d
efttabletype
  INITLOAD(sysglobals)                  170d   175d
eg_clr
  G_DRV(extg)                   ***     183d
  G_DRV(init_gpio)              ***     247c
eg_init
  G_DRV(extg)                   ***     172d
  G_DRV(init_gpio)              ***     237c
eg_isr
  G_DRV(extg)                   ***     173d
  G_DRV(init_gpio)              ***     238c
eg_rdb
  G_DRV(extg)                   ***     174d
  G_DRV(init_gpio)              ***     239c
eg_rds
  G_DRV(extg)                   ***     178d
  G_DRV(init_gpio)              ***     243c
eg_rdw
  G_DRV(extg)                   ***     176d
  G_DRV(init_gpio)              ***     241c
eg_set
  G_DRV(extg)                   ***     184d
  G_DRV(init_gpio)              ***     246c
eg_test
  G_DRV(extg)                   ***     185d
  G_DRV(init_gpio)              ***     248c
eg_tfr
  G_DRV(extg)                   ***     182d
  G_DRV(init_gpio)              ***     245c
eg_wtb
  G_DRV(extg)                   ***     175d
  G_DRV(init_gpio)              ***     240c
eg_wtc
  G_DRV(extg)                   ***     180d
  G_DRV(init_gpio)              ***     244c
eg_wtw
  G_DRV(extg)                   ***     177d
  G_DRV(init_gpio)              ***     242c
```

```
egetinfo
  EDRIVER(edriver)                       42d   325d   331c
  ETU                           ***     660c
eglobal
  INITLOAD(loader)                     1070d  1127c  1133c
  M68KSYS(ci)                           190c   192c
eh_clr
  H_DRV(exth)                   ***     183d
  H_DRV(init_hpib)              ***     248c
eh_end
  H_DRV(exth)                   ***     187d
  H_DRV(init_hpib)              ***     243c
eh_init
  H_DRV(exth)                   ***     170d
  H_DRV(init_hpib)              ***     235c
eh_isr
  H_DRV(exth)                   ***     171d
  H_DRV(init_hpib)              ***     236c
eh_ppoll
  H_DRV(exth)                   ***     182d
  H_DRV(init_hpib)              ***     246c
eh_rdb
  H_DRV(exth)                   ***     172d
  H_DRV(init_hpib)              ***     237c
eh_rds
  H_DRV(exth)                   ***     176d
  H_DRV(init_hpib)              ***     241c
eh_rdw
  H_DRV(exth)                   ***     174d
  H_DRV(init_hpib)              ***     239c
eh_send
  H_DRV(exth)                   ***     181d
  H_DRV(init_hpib)              ***     245c
eh_set
  H_DRV(exth)                   ***     184d
  H_DRV(init_hpib)              ***     247c
eh_test
  H_DRV(exth)                   ***     185d
  H_DRV(init_hpib)              ***     249c
eh_tfr
  H_DRV(exth)                   ***     180d
  H_DRV(init_hpib)              ***     244c
eh_wtb
  H_DRV(exth)                   ***     173d
  H_DRV(init_hpib)              ***     238c
eh_wtc
  H_DRV(exth)                   ***     178d
  H_DRV(init_hpib)              ***     242c
eh_wtw
  H_DRV(exth)                   ***     175d
  H_DRV(init_hpib)              ***     240c
eheap
  INITLOAD(loader)                     1066d  1126c  1132c
  M68KSYS(ci)                           192c   807c
eight_diget_epsilon
  DGL_VARS(dgl_vars)            ***    1150s
  LIB(dgl_lib)                        20519c 20520c 20915c 20917c
einit
  EDRIVER(edriver)                       45d   343d   348c
  ETU                           ***     641c
eirbyte
  KERNEL(general_0)                    1207c  1265c
  KERNEL(iodeclarations)                499d   513d   527d
```

```
em
  MINIT(cs80ir)                        607d   618c   619c
emitarcef
  LIBRARIAN                            580d   588c   592c   598c   632c
emitarcisp
  LIBRARIAN                            586d   635c   664c   669c
emitdir
  LIBRARIAN                            571d   582c   607c   608c   630c   631c   665c   668c   674c   689c   692c   853c   857c   886c
                                       894c   954c   957c   962c   977c   982c  1015c  1019c  1036c  1069c  1090c  1098c  1113c  1115c
                                      1126c  1128c  1130c  1134c
emitea
  LIBRARIAN                            626d   675c   680c   689c   692c   703c   722c   726c   814c   852c   856c   862c   866c   871c
                                       877c   883c   889c   902c   907c   915c   921c   926c  1007c  1043c  1051c  1084c  1088c  1098c
                                      1113c  1115c
emitimm
  LIBRARIAN                            614d   932c   947c   992c   993c  1068c
emitincx
  LIBRARIAN                            601d   636c   641c
emitint
  LIBRARIAN                            458d   499c   502c   616c   950c
emitpostincr
  LIBRARIAN                            591d   633c  1108c
emitpredecr
  LIBRARIAN                            596d   634c  1136c
emitreglist
  LIBRARIAN                            739d   900c   914c
emitsreg
  LIBRARIAN                            818d   977c   982c
emitunop
  LIBRARIAN                            810d   864c   868c   874c   880c   909c
emptydsa
  HEAP1(hpm)                           121d   129c
enab
  EDRIVER(edriver)                      58d   211c   255c   259c
enable
  A804XDVR(a804xdvr)                   163d   165c
enable_auto_release
  CS80(cs80)                    ***    236d
enable_byte
  F9885(f9885dvr)               ***    108d
enable_report_changes
  TAPEBKUP                             1021d  1296c  1309c  1405c  1435c  1469c
end_error_link
  KERNEL(general_0)                    958d   977c
end_mode
  DC_DRV(intdc)                        450c   478c   513c
  DISCHPIB(dischpib)            ***    517c
  IOLIB(general_4)                    1423c  1546c
  KERNEL(iodeclarations)        ***    576d
  SRM_DRV(srm)                  ***   1349c
end_of_cylinder
  AMIGO(amigodvr)               ***    809c
  AMIGO(csamigo)                ***     75d
end_of_file
  CS80(cs80)                    ***    176d
  CS80(cs80dsr)                 ***    692c
  TAPEBKUP(cs80tbdvr)                  572c   804c
end_of_volume
  CS80(cs80)                    ***    177d
  CS80(cs80dsr)                 ***    679c
  TAPEBKUP(cs80tbdvr)                  559c   791c
end_set
  IOLIB(hpib_1)                        413d   496d   504c
```

```
end_x
  DGL_POLY(dgl_poly)                 20227c 20235c 20822c
  GLE_HPGL(gle_hpgl_out)              7343c  7346c  7369c  7372c
  GLE_SMARK(gle_smark)                5136c  5160c  5172c
  GLE_TYPES(gle_types)          ***   1112d
  LIB(ogl_lib)                       20128c 20139c 20240c 20576c 20606c 20633c 20663c
end_y
  DGL_POLY(dgl_poly)                 20228c 20236c 20823c
  GLE_HPGL(gle_hpgl_out)              7345c  7347c  7371c  7373c
  GLE_SMARK(gle_smark)                5137c  5161c  5173c
  GLE_TYPES(gle_types)          ***   1113d
  LIB(ogl_lib)                       20129c 20140c 20241c 20577c 20607c 20634c 20664c
endbytes
  INIT(misc)                           538d   638c   639c   640c   671c   675c   677c
endcmdproc
  HPHIL(hphil)                         282d   332c
enddefs
  INITLOAD(loader)                    1422d  1432c  1453c
ending
  DGL_POLY(dgl_poly)                 20547d 20569c
endioerrs
  INITLOAD(sysglobals)          ***    196d
endisrhook
  INITLOAD(sysglobals)          ***    253d
endline
  CONVERT(convert_text)                 47d    59c    62c    76c    77c
  UCSD_AM(ucsd_am)                     149d   163c   252c   259c
endmod
  INIT(ldr)                           2389c  2432c
  INITLOAD(loader)                    1073d  1360c  1361c  1374c  1406c  1429c  1482c  1521c  1600c
  SEGMENTER(asm)                ***    191c
endofproc
  LIBRARIAN                            298d  1214c  1238c
endp
  INITLOAD(loader)                    1568d  1570c  1571c
endrefs
  LIBRARIAN                           2736d  2772c  2774c
endscan
  EDRIVER(edriver)                     100d   115c   116c   200c   225c   228c   273c   276c
  ETU                                 1161d  1191c  1216c
endsysvars
  INITLOAD(sysglobals)          ***    294s
endtext
  LIBRARIAN                           2194d  2605c  2712c
enoeprom
  EDRIVER(edriver)                      34d   114c   190c
  ETU                           ***    642c
enoerror
  EDRIVER(edriver)                      34d   128c
  ETU                                  573c   579c   594c   642c
enotblank
  EDRIVER(edriver)                      34d   276c
enotprog
  EDRIVER(edriver)                      34d   131c   135c   139c   141c
enter_bdx_chain
  DISCHPIB(dischpib)                   436d   487c   495d   535c
enter_data
  DC_DRV(extdc)                 ***    200d
  DC_DRV(intdc)                        317c   335c   464c   494c
enter_transfer_chain
  AMIGO(amigodvr)                      544d   693c   700d   961c   964c
entries
  M68KSYS(ci)                          784d   803c   823c
```

```
entry
  INITLOAD(bootdammodule)     ***    585d
  LIBRARIAN                         2722d  2724c
  MINIT(cs80ir)               ***    547d
  MINIT(qminit)                     1187c  1203c
entrypoint
  INIT(ldr)                         2327c  2332c  2354c  2364c  2482c
  INITLOAD                          1715c  1717c  1768c  1787c
  INITLOAD(loader)                  1069d  1126c  1132c  1426c  1444c  1445c  1485c
  M68KSYS(ci)                        880c   885c   928c   929c   938c  1011c  1136c
  SEGMENTER(asm)                     150c   165c   261c
  TAIL                        ***     86c
entrysize
  LIBRARIAN                           32d  3189c  3190c
  LIFDAM(lifmodule)                   33d   504c
environ
  CRT(crt)                    ***    110d
  GCRT(crtb)                  ***     43d
  SYSDEVS(sysdevs)                   104d   111d
environc
  CRT(crt)                           110d   719c
  GCRT(crtb)                          43d   568c
environptr
  SYSDEVS(sysdevs)                   111d   116d
eod
  LIFDAM(lifmodule)                  716d   720c   736d   747c   759c   763c   783c   785c  1016d  1020c  1025c  1029c  1033c  1042c
eoi_line
  DISCHPIB(dischpib)                 302c   319c
  KERNEL(iodeclarations)      ***    391d
eoi_set
  DISCHPIB(dischpib)                 360d   370c   371c   391d   399c   400c
eol
  CONVERT(convert_text)               67c    78c   111c
  CRT(crt)                           442c   474c
  GCRT(crtb)                         312c   345c
  INIT(misc)                         168d   525c   526c   581c   598c   684c
  KEYS(keys)                  ***    122c
  LIBRARIAN                         1376c  1404c
  MOREFSYS(mfs)               ***     95c
  UCSD_AM(ucsd_am)                    80c   179c   199c   201c   251c
eop
  EDRIVER(edriver)                   312d   316c   317c   318c   319c   320c   322c
eoptype
  EDRIVER(edriver)                    33d    37d   307d
eor
  AMIGO(amigodvr)             ***    848c
eot
  GLE_KNOB(gle_knob_in)             18031c 18040c
  UNITIO(uio)                          38c    47c
eot_parm
  DC_DRV(intdc)               ***    539c
  DISCHPIB(dischpib)          ***    482c
  IOLIB(general_4)            ***   1278c
  KERNEL(iodeclarations)      ***    585d
  SRM_DRV(srm)                ***   1204c
eot_proc
  DC_DRV(intdc)                      536c   539c
  DISCHPIB(dischpib)          ***    481c
  IOLIB(general_4)                  1276c  1277c
  KERNEL(iodeclarations)      ***    584d
  SRM_DRV(srm)                      1202c  1203c
eotproc
  DISCHPIB(bkgnd)             ***     42d
  INITLOAD(sysglobals)                70d   118d
```

```
ep
  SEGMENTER(asm)                      54d   132c   267c
epart
  ETU                         ***    853d
epblank
  EDRIVER(edriver)                    52d   266c   319c
epbrate
  EDRIVER(edriver)                    52d   294c   340c
epcheck
  EDRIVER(edriver)                    52d   279c   320c
epcwrite
  EDRIVER(edriver)                    52d   203c   219c   234c   248c   318c
epend
  EDRIVER(edriver)                    30d   116c   159c   171c   172c   174c   178c
  ETU                                673c   730c   969c   971c
eperror
  EDRIVER(edriver)                    34d    40d    43d    45d    48d    79d   105d   310d   313d   326d   329d   334d   337d   343d
                                     346d
  ETU                                 74d   639d
epfail
  EDRIVER(edriver)                    34d   220c   235c   249c
epinc
  ETU                                 50d   502c   511c   517c   543c   674c   679c   967c  1191c
epinfo
  EDRIVER(edriver)                    52d   296c   331c
  ETU                                 73d   660c   661c   670c   689c   730c   731c   767c   788c   798c   944c   992c
epinforec
  EDRIVER(edriver)                    27d    43d    71d   122d   326d
  ETU                         ***     73d
epinit
  EDRIVER(edriver)                    52d   145c   348c
epoptype
  EDRIVER(edriver)                    52d    75d   312d
epprog
  EDRIVER(edriver)                    74d   322c   331c   340c   348c
epread
  EDRIVER(edriver)                    52d   195c   316c
eprog
  EDRIVER(edriver)                    36d   306d   322c
  ETU                                572c   578c   593c
eprom
  CTABLE                      ***   1260c
  CTABLE(ctr)                         260d   816c
eprom_prgmr
  KERNEL(general_0)           ***   1085c
  KERNEL(iodeclarations)      ***    332d
eprom_tm
  EPROMS(eproms)                       8d    12d
eprom_tm_name
  CTABLE(ctr)                         367d   816c
epromdata
  ETU                                 84d   495c   539c   664c   667c   669c   670c   683c   689c   690c   745c   806c   883c  1023c
                                     1180c
epromlist
  ETU                                 83d   664c   682c   683c  1233c
epromptr
  ETU                                 45d    47d    83d    84d
epromrec
  ETU                                 45d    46d
eproms
  EPROMS(eproms)              ***      4d
epsize
  ETU                                 49d   673c   674c   981c   988c   991c
```

```
epstart
  EDRIVER(edriver)                    29d   114c   116c   158c   170c   171c   190c
  ETU                                661c   672c   673c   689c   730c   731c   767c   788c   798c   949c   974c   976c   992c
eptr
  EDRIVER(edriver)                    70d    96d
eptype
  EDRIVER(edriver)                    61d    70d
epwrite
  EDRIVER(edriver)                    52d   203c   317c
erase
  GLE_RGL(gle_ras_out)          ***  8078d
erase_mode
  DGL_VARS(dgl_vars)            ***  1203d
erase_support
  GLE_HPGL(gle_hpgl_out)        ***  7699c
  GLE_TYPES(gle_types)          ***  1094d
eraseeol
  CRT(crt)                      ***   123d
  GCRT(crtb)                    ***    56d
  SYSDEVS(sysdevs)              ***    77d
eraseeos
  CRT(crt)                      ***   124d
  GCRT(crtb)                    ***    57d
  SYSDEVS(sysdevs)              ***    78d
eread
  EDRIVER(edriver)                    33d   316c
err
  LIBRARIAN                          3616d  3663c
err_aspect
  DGL_VARS(dgl_vars)            ***  1226d
  LIB(dgl_lib)                  ***  20356c
err_bad_parms
  DGL_INQ(dgl_inq)                   6047c  6073c
  DGL_POLY(dgl_poly)                20181c 20936c
  DGL_VARS(dgl_vars)            ***  1227d
  LIB(dgl_lib)                      20295c 20330c 20390c 20404c 20423c 20799c
err_dis_int
  DGL_VARS(dgl_vars)            ***  1223d
  GEN(dgl_gen)                  ***  3086c
  LIB(dgl_lib)                      20582c 20612c 20641c 20672c 20731c
err_disp_eq_loc
  DGL_VARS(dgl_vars)            ***  1230d
err_echo_dis_int
  DGL_HPGLI(dgl_hpgli)          ***  18119c
  DGL_KNOB(dgl_knob)            ***  18165c
  DGL_VARS(dgl_vars)            ***  1225d
err_loc_int
  DGL_VARS(dgl_vars)            ***  1224d
  GEN(dgl_gen)                  ***  3094c
  LIB(dgl_lib)                  ***  20946c
err_neg_points
  DGL_POLY(dgl_poly)            ***  20935c
  DGL_VARS(dgl_vars)            ***  1235d
  LIB(dgl_lib)                      20685c 20701c
err_no_ctable
  DGL_INQ(dgl_inq)              ***  6046c
  DGL_VARS(dgl_vars)            ***  1234d
err_no_display_hardware
  DGL_KNOB(dgl_knob)            ***  18078c
  DGL_VARS(dgl_vars)            ***  1232d
err_out_loc
  DGL_VARS(dgl_vars)            ***  1233d
err_out_phys
  DGL_VARS(dgl_vars)            ***  1228d

err_out_virt
  DGL_VARS(dgl_vars)            ***  1231d
  LIB(dgl_lib)                  ***  20303c
err_out_wind
  DGL_VARS(dgl_vars)            ***  1229d
err_sys_int
  DGL_VARS(dgl_vars)            ***  1222d
  GEN(dgl_gen)                  ***  3078c
  LIB(dgl_lib)                      20583c 20613c 20642c 20673c 20732c 20947c
errblk
  M68KSYS(ci)                         45d   630c  1009c  1117c
errcode
  F9885(f9885dvr)                     61d   205c   235c   236c
errfid
  M68KSYS(ci)                         47d   631c
errlnk_type
  KERNEL(iodeclarations)              376d   641d
errmsg
  INITLOAD                      ***  1794c
errnum
  M68KSYS(ci)                         45d   630c  1009c  1011c  1116c
error
  BUBBLES(bubble)                     37d    39c
  DGL_C_IN(dgl_config_in)       ***  12115c
  DGL_C_OUT(dgl_config_out)     ***  11484c
  DGL_HPGLI(dgl_hpgli)          ***  18119c
  DGL_INQ(dgl_inq)                    6046c  6047c  6073c
  DGL_KNOB(dgl_knob)                 18078c 18165c
  DGL_POLY(dgl_poly)                 20181c 20935c 20936c
  DGL_RAS(dgl_raster)                17083d 17129c 17133c 17134c 17141c 17145c 17146c 17151c 17155c 17156c 17158c 17160c
  DGL_VARS(dgl_vars)                  1126d  1131d
  ETU                                   74d   572c   573c   578c   579c   582c   591c   593c   594c   595c   660c  1071c
  GEN(dgl_gen)                        3012d  3064d  3078c  3086c  3094c
  GLE_HPGLI(gle_hpgl_in)        ***  18238d
  GLE_HPIB(gle_hpib_io)               10016d 10491c 10499c
  LIB(dgl_lib)                        20295c 20303c 20330c 20356c 20390c 20404c 20423c 20582c 20583c 20612c 20613c 20641c 20642c 20672c
                                      20673c 20685c 20701c 20731c 20732c 20799c 20946c 20947c
error_byte
  MINIT(cs80ir)                 ***    538d
error_conversion_table
  INITLOAD(mini)                      341d   379c
error_h
  DGL_RAS(dgl_raster)                17074d 17116c 17118c 17119c
error_h2
  DGL_RAS(dgl_raster)                17075d 17117c 17118c 17119c 17134c 17146c 17156c
error_i
  DGL_RAS(dgl_raster)                17076d 17121c 17123c 17129c 17133c 17134c
error_i2
  DGL_RAS(dgl_raster)                17078d 17123c 17145c 17146c
error_number
  GEN(dgl_gen)                        3012d  3064d  3069c
error_return
  DGL_C_IN(dgl_config_in)            12057c 12063c 12095c 12114c 12116c 12117c 12135c
  DGL_C_OUT(dgl_config_out)          11354c 11362c 11364c 11445c 11458c 11461c 11487c 11491c 11492c 11498c 11511c
  DGL_KNOB(dgl_knob)            ***  18078c
  DGL_POLY(dgl_poly)                 21049c 21094c
  GLE_GEN(gle_gen)              ***   2134c
  GLE_GEN1(gle_gen1)            ***   3032c
  GLE_HPGL(gle_hpgl_out)              7563c  7602c  7612c  7715c
  GLE_HPGLI(gle_hpgl_in)             18337c 18394c 18400c
  GLE_KNOB(gle_knob_in)         ***  18271c
  GLE_RGL(gle_ras_out)                8305c  8306c  8393c
  GLE_TYPES(gle_types)                1079d  1225d
  LIB(dgl_lib)                       21221c 21271c 21323c
```

```
error_s
  DGL_RAS(dgl_raster)              17077d 17120c 17122c 17145c 17146c
error_s2
  DGL_RAS(dgl_raster)              17079d 17122c 17133c 17134c
errorbit_type
  CS80(cs80)                         128d   202d
  CS80(cs80dsr)              ***     641d
  TAPEBKUP(cs80tbdvr)                521d   701d
errorbits
  CS80(cs80)                 ***     211d
  CS80(cs80dsr)                      658c   682c
  TAPEBKUP(cs80tbdvr)                538c   562c   766c   794c
errorbits_owning_parmfield
  CS80(cs80)                 ***     281d
  CS80(cs80dsr)              ***     661c
  TAPEBKUP(cs80tbdvr)                541c   769c
errorbits_requesting_release
  CS80(cs80)                 ***     325d
  CS80(cs80dsr)              ***     754c
  TAPEBKUP(cs80tbdvr)                644c   877c
errorcode
  BUBBLES(bubble)                     83c   102c   109c   116c   164c
  INIT(misc)                         210d   365d   373c   374c   377c   416c
  KERNEL(general_0)          ***     958d
  KERNEL(iodeclarations)     ***     376d
errorhandle
  MINIT(xminit)                     1418d  1634c  1683c  1729c  1763c  1803c
errorline
  LIBRARIAN                           92d   270c  2091c  2348c  2785c  3088c  3170c  3264c  3318c  3321c  3396c  3418c  3662c
errors
  F9885(f9885dvr)                     48d    61d
  LIBRARIAN                           59d  2347c  2982c  3010c  3675c
errortype
  MINIT(xminit)                     1406d  1424c  1553c
errsym
  M68KSYS(ci)                         45d   630c
errtype
  MINIT(asmr)                       1333d  1338d  1344d  1349d
ert
  MINIT(cs80ir)                      528d   778d   790c   792c   793c   794c   795c   796c   797c   798c   799c   800c
  MINIT(qminit)              ***    1191c
ert_area
  MINIT(cs80ir)                      567d   770d   798c
ert_area_type
  MINIT(cs80ir)                      550d   567d   770d
ert_message
  MINIT(cs80ir)                      568d   771d   791c   801c   811c
ert_msge
  MINIT(qminit)                     1103d  1186c  1187c
esc
  CRT(crt)                   ***     234c
  INIT(misc)                 ***     170d
  INITLOAD(loader)                  1589d  1646c  1649c
  KEYS(keys)                 ***     528c
escape_caught
  CS80(cs80dsr)                      774d   873c   957c   959c
  CS80(cs80dvr)                     1146d  1152c  1160c  1162c
escape_value
  F9885(f9885dvr)                     94d   101c
escapeco
  HEAPT(sysglobals)          ***      29d
escchar
  INIT(misc)                 ***     170d
```

```
esccode
  LIBRARIAN                           60d  3741c  3743c
  M68KSYS(ci)                        857d   859c   866c   872d   888c   893c   898c   900c
escesc
  INITLOAD(bootdammodule)            616d   643c   679c
escio
  INITLOAD(bootdammodule)            594d   597c   598c   599c   600c   601c   602c   720c   724c   765c
escn
  INITLOAD(bootdammodule)            594d   596c   603c
esctemp
  SEGMENTER(asm)                     179d   212c   218c
etu
  ETU                        ***      24d
etx
  CRT(crt)                           105d   151d   440c   475c
  GCRT(crtb)                          84d   310c   346c
  INIT(misc)                 ***     161d
  KEYS(keys)                         120c   529c
  LIBRARIAN                  ***    1398c
  SYSDEVS(sysdevs)           ***      98d
evalgvr
  INITLOAD(loader)                  1104d  1443c  1465c  1531c
even_parity
  IOLIB(serial_3)                   2752c  2773c
  KERNEL(iodeclarations)     ***     403d
evu
  TAPEBKUP(cs80tbdvr)                708d   717c   718c   719c
evu_byte
  CS80(cs80)                 ***     124d
evu_type
  CS80(cs80)                         120d   208d
  TAPEBKUP(cs80tbdvr)        ***     711d
  TAPEBKUP(cs80tbr)                  226d   229d
ewrite
  EDRIVER(edriver)                    33d   317c
excess
  LIBRARIAN                  ***    3178d
exchangepack
  SRMDAM(srmdammodule)       ***    1380c
  SRM_DRV                    ***    1825d
  SRM_DRV(srm)               ***    1067d
exclusive_share_code
  SRMDAM(srmdammodule)               239c   288c   620c  1164c  1424c
  SRM_DRV(srm)               ***     162d
excp_line
  INIT(misc)                         368d   417c
excp_pc
  INIT(misc)                         369d   418c
executable
  INITLOAD(loader)                   948s  1439c  1526c
  LIBRARIAN                          356c  1280c  1699c  1941c  1967c  2530c  2621c  2754c
execute
  M68KSYS(ci)                        987d  1019c  1133c  1137c  1140c
execution_sec
  CS80(cs80)                         277d   466c   494c
  CS80(cs80dvr)              ***    1116c
  MINIT(cs80ir)                      619c   757c   804c
  TAPEBKUP(cs80tbr)          ***     281c
exist
  SRMDAM(srmdammodule)       ***     400c
  SRM_DRV(srm)               ***     789d
exists_proc
  SEGMENTER(asm)                     104d   106c   285c
  SEGMENTER(segmenter)       ***      36d
```

```
expand_screen
  DGL_C_OUT(dgl_config_out)          11171d 11339c 11348c
expiwait_unlock
  SRM_DRV                    ***      2056c
  SRM_DRV(srm)               ***       770d
expodigit
  MOREFSYS(mfs)                        110d   147c   173c
exponent
  MOREFSYS(mfs)                         35d   409c   473c   484c   489c   501c   586c   587c   593c   606c   613c   616c   620c   622c
                                       625c   629c   631c   633c   634c   635c   636c   637c   642c   646c   649c
expostate
  MOREFSYS(mfs)                        162d   206c   251c   289c
expsign
  MOREFSYS(mfs)                        400d   474c   477c   489c
ext
  INITLOAD(loader)           ***      1038d
  LIBRARIAN                             445d   481c   482c   491c   529c   530c   531c   604c   688c   689c   692c   781c   790c   799c
                                        820c   946c   977c   982c  1062c  1063c  1222c  1233c  1767c  1771c  1781c  1787c  1789c
ext1
  SRMDAM(srmdammodule)                 1026d  1057c  1059c  1088c  1125c  1136c  1148c  1336d  1415c  1417c  1422c
  SRM_DRV(srm)                         1052d  1739d  1766c
ext2
  SRM_DRV(srm)                         1053d  1740d  1768c
extaddr
  INIT(ldr)                  ***      2395c
  INITLOAD(loader)                     1020d  1187c  1286c  1288c  1289c  1290c  1291c  1309c  1348c
  LIBRARIAN                             397c  1356c  1572c  1691c  1692c  1767c
extblock
  INITLOAD(loader)                      955d  1288c
  LIBRARIAN                             373c  1982c  2883c
extblocks
  LIBRARIAN                            1853d  1984c  1985c  1986c
extdc
  DC_DRV(extdc)              ***       181d
  DC_DRV(init_dc)            ***       597d
  DC_DRV(intdc)              ***       261d
extdi
  DI_DRV(extdi)              ***       135d
  DI_DRV(init_disc_int)      ***       203d
extend
  LIBRARIAN                             510d   587c   603c   623c   638c   639c   640c   936c  1037c  1059c  1155c
extended_features_mode
  MINIT(midecs)              ***        35d
  MINIT(qminit)              ***      1068c
  MINIT(xminit)              ***      1504c
  MIU!                                  155c   156c
extension
  INITLOAD(bootdammodule)               566d   671c   673c
  KEYS(keys)                            205c   206c   208c   428c
  LIFDAM(lifmodule)                      67d   612c   618c   944c   960c   961c  1004c  1005c
  NONUSKBD1(non_us_kbd1)                 60c    67c
  SYSDEVS(sysdevs)           ***       266d
extensize
  SRMDAM(srmdammodule)                   52d  1059c  1088c  1125c  1136c  1148c  1318c  1417c  1422c
```

```
external
  DC_DRV(extdc)              ***       181d
  DISCHPIB(dischpib)         ***       216d
  DI_DRV(extdi)              ***       135d
  GCRTI(crtb)                          117d   118d   119d   120d   121d   122d   123d   124d   125d   127d   128d   129d   130d   131d
                                       132d
  GLE_HPIB(gle_hpib_io)               10059d 10061d
  G_DRV(extg)                ***       153d
  HEART(sysglobals)          ***        26d
  H_DRV(exth)                ***       151d
  INIT(initunits)                     2149d  2153d  2157d  2160d  2162d  2165d
  INITLOAD(bootdammodule)              584d   586d   587d   589d   590d   592d
  INITLOAD(loader)                    1105d  1109d  1176d  1216d
  IOLIB(general_4)           ***      1260d
  IOLIB(hpib_1)                        428d   430d
  KERNEL(iocomasm)           ***       687d
  MINIT(asmr)                         1344d  1349d
  MOREFSYS                   ***       682d
  MOREFSYS(mfs)                        41d   555d   557d
  RS_DRV(rs)                 ***       156d
  SEGMENTER(asm)             ***        42d
  UCSD_DAM(ucsdmodule)                  55d    56d
extg
  G_DRV(extg)                ***       153d
  G_DRV(init_gpio)           ***       221d
exth
  H_DRV(exth)                ***       151d
  H_DRV(init_hpib)           ***       219d
extleft
  A804XDVR(a804xdvr)                    68d    77c   336c   346c   350c   368c   402c
extlist
  INITLOAD(loader)           ***      1173d
extmodptr
  INITLOAD(loader)                    1301d  1303c  1304c  1313c
extra
  INIT(ldr)                           2373d  2375c  2376c  2380c
  INITLOAD(loader)                    1083d  1136d  1153c  1154c  1158c  1159c  1164c  1165c  1590d  1602c  1603c  1613c
extraexponent
  MOREFSYS(mfs)                        401d   410c   445c   452c   500c   501c
extright
  A804XDVR(a804xdvr)                    68d    77c   336c   346c   352c   368c   402c
extsize
  INITLOAD(loader)                     956d  1285c  1286c  1287c  1288c  1291c
  LIBRARIAN                            372c   373c   398c   399c   557d   623c  1688c  1689c  1983c  2751c
extsiztype
  LIBRARIAN                            548d   557d
extstate
  A804XDVR(a804xdvr)                   327d   331c   335c   336c   343c   368c   369c   372c
exttable
  INITLOAD(loader)           ***      1172d
```

```
  AMIGO(amigodvr)                    664c   887c   900c
  AMIGO(csamigo)          ***        112d
  CTABLE(ctr)                        346d   633c   639c   645c   651c   658c   664c   673c   682c   694c   707c   716c   733c   752c
                                     768c   787c   809c   816c   834d   837c   842c
  GLE_KNOB(gle_knob_in)            18033d 18036c 18041c 18046d 18050c 18054c 18060d 18064c 18067c 18068c
  GLE_RGL(gle_ras_out)               8161d  8162d  8163d  8164d  8165d  8166d  8167d  8168d  8169d  8170d  8171d  8172d  8173d  8174d
                                     8175d  8176d  8177d
  HEAPT(hpm)                         135d   147c   148c   152c   158c   159c   161c   162c   164c   165c   166c   167c   169c   170c
                                     176c   179c   181c   184c   185c   186c   187c
  INIT(fs)                           763d   764d   765d   766d   767d   768d   769d   770d   772d   773d   774d   776d   777d   778d
                                     779d   780d   781d   783d   784d   786d   787d   789d   793d   797d   798d
  INIT(initunits)                   2141d  2160d  2165d  2232d  2256d
  INIT(misc)                         213d   263d   267c   268c   269c   272c   426d   431c   446c   447c   458c
  INITLOAD(bootdammodule)            543d   544d   607d   692c   696c   697c   699c   700c   705c   707c   709c   710c   711c   712c
                                     713c   714c   747d   752c   773c
  INITLOAD(loader)                  1118d  1120c
  INITLOAD(sysglobals)    ***        140d
  LIBRARIAN                          106d   116c   118c   119c   121c   131d   133c   137d   139c   221d   224c   231c   232c   233c
                                     235c   236c   237c   238c  2336d  2340c  2740d  2742c
  LIFDAM(lifmodule)                   28d   195d   222c   223c   309c   394c   488c   593c   595c   606c   607c   636c   646c   915c
                                     916c   917c   918c   919c   920c   922c   923c   931c   935c   936c   937c   954c   958c   959c
                                     960c   961c   969c   970c   973c   975c   980c   995c  1001c  1004c  1005c  1006c  1022c  1054c
                                    1065c  1072c  1077c  1086c  1088c  1092c  1108c  1110c  1127c  1128c  1130c  1133c  1136c  1141c
                                    1149c  1150c  1152c  1153c  1155c  1159c  1162c  1164c  1165c
  LOCKMOD(lockmodule)                 30d    31d    32d    37d    40c    49c    62d    65c    72c    78d    81c    90c
  M68KSYS(ci)                        396d   413c   414c   439c   451c   502c   514c   523c   537c   551c   586d   591d   595c   596c
                                     598c   602c   603c   619c   620c   657d   661d   664c   666c   667c   699c   702c   703c   790d
                                     818c   825c   954d   956c   962c   963c
  MINIT(hminit)           ***        480c
  MINIT(iramigo)                     226d   234c
  MIUI                                15d   177c   198c   334c   350c   475c
  MOREFSYS(mfs)                       45d    67c    78c   102c   104c
  SRMAM(srmammodule)                  50d    60c    97d   107c   144d   149c
  SRMDAM(srmdammodule)                43d   119d   122c   133d   140c   142c   446d   452c   456d   476c   478c   499d   519c   521c
                                     562d   582c   645d   670c   711c   875d   880c   882c   901d   917c   956d   973c   975c   980c
                                     986d   996c  1013d  1038c  1079c  1119c  1159c  1164c  1170d  1183c  1185c  1187c  1210d  1222c
                                    1224c  1226c  1245d  1251c  1253c  1263d  1271c  1286c  1310d  1316c  1329d  1342c  1355c  1359c
                                    1360c  1372c  1384c  1394c  1424c  1430c  1438d  1464c  1472d  1475c  1496d  1499c  1503c  1519d
                                    1539c  1624d  1643c  1690c  1700c  1703c  1706c  1712c  1715c  1723c  1726c  1738c  1741c  1750c
                                    1753c  1760c  1782c  1784c  1766c  1770c  1773c  1779c  1781c  1783c  1785c  1787c  1789c  1791c
                                    1796c
  UCSD_DAM(ucsdmodule)                32d    66d   130c   160c   173c   210c   212c   217c   243c   268c   288c   310c   329c   363c
                                     406c   409c   439c   460c   476c   477c   511c   532c   576c   631c   650c   654c   657c   661c
  UNITIO(uio)                         40d    43c    48c    55d    59c    65c    67c    76d    80c    86c    93d    96c    97c   102d
                                     106c   109c   110c   111c
f0
  CTABLE(brstuff)         ***        946d
f1
  CTABLE(brstuff)         ***        946d
f2
  CTABLE(brstuff)         ***        946d
f3
  CTABLE(brstuff)         ***        946d
f4
  CTABLE(brstuff)         ***        946d
f5
  CTABLE(brstuff)         ***        946d
f6
  CTABLE(brstuff)         ***        946d
f7
  CTABLE(brstuff)                    946d  1001c
f9885_tm_name
  CTABLE(ctr)                        363d   694c
```

```
f9885dvr
  F9885(f9885dvr)         ***         34d
f9885init
  F9885                   ***         31d
f9885io
  F9885(f9885dvr)                     40d    45d
f_pwr_on
  INITLOAD(mini)                     398c   459c
  MINIT(mminit)           ***        171c
faccess
  INIT(fs)                           745d   764d   765d
failcode
  ETU                                532d   546c
fakeaddr
  HEAPT(hpm)                         135d   141c   149c
fakeone
  HEAPT(hpm)                          78d   122d   141c   142c   143c   161c
fanonctr
  INITLOAD(sysglobals)    ***        104d
  SRMDAM(srmdammodule)               163c  1043c
fanonfile
  INIT(fs)                ***        770d
fanonymous
  ETU                     ***        323c
  INITLOAD(sysglobals)    ***         81d
  LIFDAM(lifmodule)                  636c   646c   935c   969c   973c  1065c  1086c
  SRMDAM(srmdammodule)               144c  1077c  1283c  1304c  1364c  1454c  1737c  1743c  1759c
  UCSD_DAM(ucsdmodule)                76c   210c
fast
  EDRIVER(edriver)                    56d   157c   210c
fastburn
  ETU                                 71d   559c   580c   587c   591c   793c  1071c  1232c
fastmove
  INITLOAD(loader)                  1273c  1456c  1490c  1563c  1618c
  LIBRARIAN                         1644c  1784c  1903c  2075c  2237c  2274c  2297c  2328c  2582c  2840c  3095c
  M68KSYS(ci)                        738c   740c
fatal_ioresult
  MIUI                                23d   214c   363c   382c
  TAPEBKUP                           987d  1101c  1205c  1402c
fatal_message
  TAPEBKUP                           996d  1089c  1311c  1313c  1360c  1471c
faultlog
  CS80(cs80)              ***        216d
fb0
  INITLOAD(sysglobals)    ***        108d
fb1
  INITLOAD(sysglobals)    ***        108d
fb_ptr
  GLE_RGL(gle_ras_out)               8101d  8236c  8386c
fb_ptr_ptr
  GLE_RGL(gle_ras_out)               8102d  8385c  8386c
```

```
fblksize
  ASCII(asciimodule)          ***     10d
  INIT(misc)                          547c   559c   563c   567c   589c   592c   595c   626c   627c   629c   631c   636c   639c   646c
                                      649c   659c   663c   664c   667c   668c
  INITLOAD                    ***    1697c
  INITLOAD(loader)                    813d   839d  1120c
  INITLOAD(sysglobals)                 21d   129d
  LIBRARIAN                           133c   139c
  M68KSYS(ci)                 ***     816c
  UCSD_AM(ucsd_am)                     43d    62c    63c    75c    76c    78c    79c    82c    84c    92c    94c   109c   119c   139c
                                      142c   144c   162c   165c   184c   205c   221c   241c   247c
  UCSD_DAM(ucsdmodule)                 39d   100c   162c   163c   218c   335c   365c   368c   370c   371c   416c   417c   418c   419c
                                      483c   488c   495c   501c   503c   504c   517c   518c   519c   521c   547c   549c   550c   582c
                                      584c   604c   606c
  UNITIO(uio)                          65c    86c
fblockio
  INIT(fs)                    ***     789d
fbp
  INIT(ldr)                   ***    2378c
  INITLOAD(loader)                     912d  1141c  1390c  1391c  1437c  1438c  1605c  1610c
  LIBRARIAN                           1296c  1299c  1393c  2258c  2299c  2771c  2817c  2867c  3018c
fbufchanged
  ASCII(asciimodule)                   39c    48c    69c    94c   117c   136c   253c   265c   271c   298c
  ETU                         ***     324c
  INIT(misc)                          544c   550c   632c   650c   665c   678c
  INITLOAD(sysglobals)        ***      89d
  LIFDAM(lifmodule)           ***     298c
  UCSD_AM(ucsd_am)                     64c   100c   140c
  UCSD_DAM(ucsdmodule)        ***      84c
fbuffer
  ASCII(asciimodule)                   37c    50c    74c    94c   115c   116c   159c   203c   223c   251c
  INIT(misc)                          269c   548c   564c   567c   598c   600c   629c   631c   646c   649c   659c   664c   675c   677c
  INITLOAD(sysglobals)        ***     129d
  UCSD_AM(ucsd_am)                     63c    76c    80c    83c    84c    92c    98c   117c   165c   203c   234c   245c
  UCSD_DAM(ucsdmodule)                500c   501c   503c   504c
fbuffered
  ETU                                 321c   902c
  INIT(misc)                  ***     471c
  INITLOAD(bootdammodule)     ***     713c
  INITLOAD(loader)            ***    1607c
  INITLOAD(sysglobals)        ***      80d
  LIFDAM(lifmodule)                   298c   920c
  SRMDAM(srmdammodule)        ***     948c
  UCSD_DAM(ucsdmodule)                 84c   164c
fbufferref
  INIT(fs)                    ***     776d
fbufvalid
  INITLOAD(sysglobals)        ***      85d
  LIFDAM(lifmodule)           ***     289c
  UCSD_DAM(ucsdmodule)        ***      78c
fbusy
  AMIGO(amigodvr)             ***     587c
  BUBBLES(bubble)                     115c   116c
  CS80(cs80dvr)               ***    1373c
  F9885(f9885dvr)             ***     272c
  GLE_KNOB(gle_knob_in)       ***   18068c
  INITLOAD(mini)              ***     500c
  INITLOAD(sysglobals)        ***     116d
  KEYS(keys)                  ***      83c
  M68KSYS(ci)                         120c   140c   141c   382c
  UNITIO(uio)                         110c   111c
fclose
  INIT(fs)                    ***     767d
```

```
fcloseit
  INIT(fs)                    ***     766d
fd
  F9885(f9885dvr)                      53d   111d   115d
fdate
  INITLOAD(bootdammodule)     ***     563d
  LIFDAM(lifmodule)                    64d   372c   548d   550c   551c   552c   555c   556c   557c   602c   617c   651c   652c   940c
                                      941c  1008c  1093c  1094c
fdirectory
  INIT(ldr)                           2436c  2437c  2446c  2447c  2452c
  INITLOAD(loader)                    1053d  1144c  1145c  1146c  1152c  1161c  1163c  1318c  1621c  1622c
  LIBRARIAN                           1286c  1324c  1352c  1386c  2854c  2855c  2959c  2967c  3020c  3022c  3029c  3036c  3039c  3064c
                                      3065c  3068c  3087c  3097c  3255c  3256c  3279c  3280c  3312c  3326c  3328c  3348c  3361c  3372c
                                      3374c  3428c  3433c  3438c  3452c  3465c  3471c  3490c  3586c  3605c  3623c  3631c  3632c  3642c
                                      3652c  3735c
  SEGMENTER(asm)                       194c   195c
fdp
  INITLOAD(loader)                     910d  1144c  1152c  1159c  1161c
  LIBRARIAN                           2854c  2859c  2888c  3106c  3112c  3153c
feft
  ETU                                 320c   903c
  INITLOAD(sysglobals)        ***      77d
  LIFDAM(lifmodule)                   937c   959c  1110c
  SRMDAM(srmdammodule)                934c   938c   939c   940c  1045c  1047c  1090c  1387c  1418c  1452c
  UCSD_DAM(ucsdmodule)                212c   291c
feof
  ETU                                 327c   468c   472c   481c  1034c  1045c
  INIT(fs)                    ***     779d
  INITLOAD(sysglobals)        ***      87d
  LIFDAM(lifmodule)           ***     290c
  UCSD_DAM(ucsdmodule)        ***      79c
feoln
  ASCII(asciimodule)                  207c   208c
  ETU                                 327c   459c   461c  1034c
  INIT(fs)                    ***     780d
  INIT(misc)                          525c   606c   684c   685c
  INITLOAD(sysglobals)        ***      86d
  KEYS(keys)                  ***     127c
  M68KSYS(ci)                         348c   351c
  UCSD_AM(ucsd_am)                    152c   230c   253c   254c
feot
  AMIGO(amigodvr)             ***     600c
  CRT(crt)                            446c   496c
  CS80(cs80dvr)               ***    1340c
  DISCHPIB(bkgnd)                      42d   139c
  F9885(f9885dvr)             ***     298c
  GCRT(crtb)                          316c   364c
  GLE_KNOB(gle_knob_in)       ***   18040c
  INIT_OAD(mini)              ***     526c
  INIT_OAD(sysglobals)        ***     118d
  KEYS(keys)                  ***     136c
  UNITIO(uio)                 ***      47c
fetchbuffer
  INIT(misc)                          554d   593c   644c   658c   673c
fetchdir
  UCSD_DAM(ucsdmodule)                 70d   121c   128c   463c   654c   655c   660c
fetchvolume
  UCSD_DAM(ucsdmodule)                124d   127c   133c   211c   269c   331c   479c   512c   533c   578c
fextra
  INIT_OAD(sysglobals)        ***     123d
fextra2
  INIT_OAD(sysglobals)        ***     124d
```

```
ff
  INIT(misc)                  ***    167d
  KEYS(keys)                         479c   508c
ffib
  ETU                                418d   434c   436c   458c   479c
ffpw
  INITLOAD(sysglobals)        ***    101d
  SRMDAM(srmdammodule)               128c   719c   860c  1671c  1672c  1680c  1807c
fget
  INIT(fs)                    ***    777d
fgetxy
  ETU                                588c  1179c
  INIT(fs)                    ***    798d
  M68KSYS(ci)                 ***    128c
  MINIT(mminit)               ***    157c
  MIUI                               246c   260c
  TAPEBKUP                          1133c  1147c
fgotoxy
  ETU                                297c   300c   558c   590c   719c   727c   743c   758c   772c   873c   913c   932c  1029c  1150c
                                    1174c  1208c  1240c  1280c
  INIT(fs)                    ***    797d
  LIBRARIAN                           93c   100c   102c  1424c  1442c  2097c  2928c  2983c  3080c  3132c  3180c  3196c  3202c  3208c
                                    3209c  3216c  3234c  3250c  3331c  3333c  3343c  3362c  3367c  3374c  3392c  3427c  3446c  3484c
                                    3520c  3545c  3584c  3597c  3716c  3718c
  M68KSYS(ci)                        129c   134c   137c   664c   667c   674c   682c   687c   714c
  MIUI                               248c   255c   261c   270c
  TAPEBKUP                          1135c  1142c  1148c  1157c
fhimask
  A804XDVR(a804xdvr)          ***    385c
  MINIT(mminit)                      154c   163c   167c
  SYSDEVS(sysdevs)            ***    166d
fhpopen
  INIT(fs)                    ***    764d
fhpreset
  INIT(fs)                    ***    765d
fhs_tfr
  KERNEL(iodeclarations)      ***    563d
fib
  CTABLE(ctr)                 ***    834d
  ETU                                 56d   306d
  GLE_KNOB(gle_knob_in)       ***  18033d
  INIT(fs)                           763d   764d   765d   766d   767d   768d   769d   770d   772d   773d   774d   776d   777d   778d
                                     779d   780d   781d   783d   784d   786d   787d   789d   793d   797d   798d
  INIT(initunits)                   2141d  2160d  2165d  2232d  2256d
  INIT(ldr)                   ***   2377c
  INIT(misc)                         213d   263d   426d
  INITLOAD(bootdammodule)            543d   607d
  INITLOAD(loader)                  1118d  1604c
  INITLOAD(sysglobals)               64d    72d   140d
  LIBRARIAN                         131d   137d  2740d  2830d  3076d
  LIFDAM(lifmodule)                  28d   195d
  LOCKMOD(lockmodule)                30d    31d    32d    37d    62d    78d
  M68KSYS(ci)                       790d   954d
  MIUI                              376d   459d
  SRMAM(srmammodule)                 50d    97d   144d
  SRMDAM(srmdammodule)               43d   119d   133d   456d   499d   562d   645d   875d   901d   956d   986d  1013d  1170d  1210d
                                    1245d  1263d  1310d  1329d  1441d  1443d  1472d  1496d  1519d  1624d
  UCSD_DAM(ucsdmodule)               32d    66d
  UNITIO(uio)                 ***    40d
```

```
fibp
  AMIGO(amigodvr)             ***    430d
  ASCII(asciimodule)                   6d    13d
  BUBBLES(bubble)                     29d    53d
  CRT(crt)                    ***    402d
  CS80(cs80dvr)                      987d  1214d
  DISCHPIB(bkgnd)             ***     43d
  EPROMS(eproms)                       8d    12d
  ETU                         ***    418d
  F9885(f9885dvr)             ***     40d
  GCRT(crtb)                  ***    272d
  GLE_KNOB(gle_knob_in)             18031c 18068c
  INIT(initunits)                   2139d  2148d  2152d  2156d  2228d
  INIT(ldr)                         2296c  2298c  2304c  2306c
  INIT(misc)                         483d   515d   534d
  INITLOAD(bootdammodule)            544d   747d
  INITLOAD(loader)            ***    912d
  INITLOAD(mini)                     309d   384d
  INITLOAD(sysglobals)                64d    68d    70d    74d
  KEYS(keys)                  ***     72d
  LIBRARIAN                   ***   2754c
  LIFDAM(lifmodule)                  246c   251c   285c   315c   317c   319c   352c   390d   407c   528c   530c  1122c  1144c
  M68KSYS(ci)                         89c   117c   119c   306d   547c   548c   567c   731d
  MINIT(bminit)               ***     64d
  MOREFSYS(mfs)               ***    520c
  PRINTER(prtdvr)                     40d    47d
  SRMAM(srmammodule)                  39d   177d
  SRMDAM(srmdammodule)               1228c  1230c  1234c  1236c  1240c
  SWVOL                               22c    24c
  SYSDEVS(sysdevs)                   372d   376d   407d   432d   484d
  UCSD_AM(ucsd_am)            ***     40d
  UCSD_DAM(ucsdmodule)                74c   155c   477c   498c   500c   503c   504c   572d   623c   630c   675c   676c
  UNITIO(uio)                         38c   110c   111c
fibptr
  AMIGO(amigodvr)             ***    601c
  DISCHPIB(bkgnd)                     43d   139c
  HEAPT(sysglobals)           ***     30d
  SRMDAM(srmdammodule)               1441d  1445c  1459c  1466c
fid
  CTABLE                      ***   1229d
  ETU                                 64d   305d   349d   350d   844d   845d
  INIT(fs)                           753d   757d   760d
  INIT(ldr)                         2274d  2281d  2351d  2406d
  INITLOAD(loader)                  1093d  1583d
  INITLOAD(sysglobals)                43d   128d   267d
  LIBRARIAN                   ***   2921d
  LIFDAM(lifmodule)           ***    163d
  M68KSYS(ci)                         41d    47d    57d    72d    73d    93d   100d   398d   662d   672d   843d   912d   914d   988d
                                     999c  1000c  1017d
  MIUI                        ***    126d
  SEGMENTER(asm)                     175d   230d   255d   278d   289d   316d
  SEGMENTER(segmenter)                27d    28d    32d    33d
  SRMDAM(srmdammodule)               1173d  1634d
  SRM_DRV                            1846d  1856c  1882d  1893c  1942d  1954c  1996d  2010c  2023d  2035c  2046d  2055c  2082d  2096c
  SRM_DRV(srm)                       1034d  1071d  1075d  1086d  1096d  1102d  1106d  1109d  1682d  1690c
  TAPEBKUP                    ***   1056d
fid1
  SRM_DRV                            1826d  1836c
  SRM_DRV(srm)                ***   1068d
fid2
  SRM_DRV                            1827d  1837c
  SRM_DRV(srm)                ***   1069d
```

fidleng		
INITLOAD(sysglobals)		20d 43d
SRMDAM(srmdammodule)	***	1682c
fidptr		
SRMDAM(srmdammodule)		1173d 1193c 1194c 1195c
field		
M68KSYS(ci)		209d 225c
fieldwidth		
MIUI		228d 261c
TAPEBKUP		1115d 1148c
file_code		
SRMDAM(srmdammodule)		603c 604c 924c 934c 935c 1109c
SRM_DRV(srm)		278d 495d 663d 1762c
file_header_type		
SRM_DRV(srm)		266d 377d 400d 423d 494d 520d 521d 538d 539d 647d 696d 1512d
file_id		
SRMDAM(srmdammodule)		819c 848c 927c 931c 1101c
SRM_DRV		1856c 1893c 1954c 2010c 2035c 2055c 2096c
SRM_DRV(srm)		456d 572d 630d 659d 678d 712d 734d 753d 769d 802d 1690c
file_id_1		
SRM_DRV	***	1836c
SRM_DRV(srm)	***	555d
file_id_2		
SRM_DRV	***	1837c
SRM_DRV(srm)	***	556d
file_id_type		
SRMDAM(srmdammodule)		657d 658d 1338d
SRM_DRV		1784d 1789d 1826d 1827d 1846d 1882d 1906d 1942d 1969d 1996d 2023d 2046d 2082d
SRM_DRV(srm)		198d 268d 456d 474d 476d 555d 556d 572d 630d 659d 678d 712d 734d 753d 769d 802d 1009d 1017d 1025d 1034d 1036d 1038d 1045d 1059d 1064d 1068d 1069d 1071d 1075d 1081d 1086d 1093d 1096d 1102d 1106d 1109d 1514d 1551d 1589d 1627d 1682d 1703d 1705d 1732d
file_index		
SRM_DRV(srm)		374d 1566c
file_info		
SRMDAM(srmdammodule)		422c 1103c 1445c 1485c
SRM_DRV(srm)	***	578d
file_info_type		
SRM_DRV(srm)		274d 386d 578d
file_init		
DGL_C_OUT(dgl_confg_out)	***	11443c
GLE_FILE(gle_file_io)		9017d 9049d
file_inq_timeout		
DGL_C_OUT(dgl_confg_out)	***	11435c
GLE_FILE(gle_file_io)		9019d 9037d
file_iocb		
DGL_C_OUT(dgl_confg_out)	***	11035d
GLE_FILE(gle_file_io)		9009d 9010d
file_iocb_ptr		
DGL_C_OUT(dgl_confg_out)		11370d 11402d
GLE_FILE(gle_file_io)		9009d 9055c 9065c 9075c
file_iocb_space		
DGL_C_OUT(dgl_confg_out)		11035d 11431c
file_len		
MIUI		412d 446d
file_name		
DGL_C_OUT(dgl_confg_out)	***	11439c
GLE_FILE(gle_file_io)		9011d 9057c
MIUI		409d 443d
SRMDAM(srmdammodule)		127c 151c 156c 165c 431c 432c 602c 767c 770c 787c 857c 1004c 1191c 1195c 1196c
SRM_DRV(srm)		275d 290d
file_name_header		
SRM_DRV		1922c 1983c
SRM_DRV(srm)		377d 400d 423d 494d 647d 696d 1573c 1611c 1644c 1756c
file_name_header1		
SRM_DRV(srm)	***	538d
file_name_header2		
SRM_DRV(srm)	***	539d
file_name_set		
SRMDAM(srmdammodule)		120d 463d 506d 572d 966d 994d 1178d 1181d 1217d 1220d 1249d 1334d
SRM_DRV(srm)		289d 926d 927d
file_password		
SRM_DRV(srm)		458d 1692c
file_set_timeout		
DGL_C_OUT(dgl_confg_out)	***	11436c
GLE_FILE(gle_file_io)		9020d 9043d
file_term		
DGL_C_OUT(dgl_confg_out)		11382c 11453c
GLE_FILE(gle_file_io)		9021d 9072d
file_type		
MIUI		410d 444d
file_write		
DGL_C_OUT(dgl_confg_out)	***	11433c
GLE_FILE(gle_file_io)		9018d 9062d
fileblock		
INITLOAD(loader)		1029d 1343c 1539c 1544c
LIBRARIAN		1296c 1299c 1394c 2632c 2633c 2771c 2817c
filedirectory		
INITLOAD(loader)		879d 1033d
filedirptr		
INITLOAD(loader)		879d 910d 1053d
filefib		
INITLOAD(loader)		1028d 1342c 1437c 1438c 1538c 1543c
LIBRARIAN		1296c 1299c 1393c 2258c 2299c 2771c 2817c
fileid		
AMIGO(amigodvr)	***	624c
BUBBLES(bubble)		77c 94c
CS80(cs80dvr)	***	1395c
EPROMS(eproms)	***	57c
F9885(f9885dvr)	***	286c
INIT(misc)		268c 452c
INITLOAD(bootdammodule)		710c 711c
INITLOAD(mini)	***	514c
INITLOAD(sysglobals)	***	107d
LIFDAM(lifmodule)		248c 291c 408c 646c 915c
M68KSYS(ci)		738c 740c 956c
MINIT(bminit)	***	77c
MIUI	***	488c
SRMAM(srmammodule)		67c 75c 84c 114c 122c 131c
SRMDAM(srmdammodule)		332d 334c 335c 337c 340c 342c 931c 1279c 1281c 1285c 1319c 1347c 1349c 1352c 1380c 1389c 1411c 1426c 1427c 1429c 1449c 1462c 1477c 1483c 1508c 1510c 1677c 1804c
UCSD_DAM(ucsdmodule)		80c 162c 163c 330c 335c 502c 585c
fileinfopack		
SRMDAM(srmdammodule)		421c 1102c 1462c 1483c
SRM_DRV	***	1845d
SRM_DRV(srm)	***	1070d

```
filekind
  ETU                          ***    309d
  INIT(fs)                            754d   756d   758d   770d
  INIT(misc)                          182d   695d
  INITLOAD                     ***   1703d
  INITLOAD(bootdammodule)             610d   653d
  INITLOAD(loader)             ***    830d
  INITLOAD(sysglobals)                 46d    78d   168d   169d   170d
  LIBRARIAN                    ***   2923d
  LIFDAM(lifmodule)                   135d   164d   540d   952d
  M68KSYS(ci)                         400d   914d
  MIUI                         ***    128d
  SRMDAM(srmdammodule)                 86d    88d
  TAPEBKUP                     ***   1058d
  UCSD_DAM(ucsdmodule)         ***     56d
filemoved
  ETU                                 847d  1022c  1064c  1116c  1124c
filename
  ETU                                 305d   316c   349d   361c
  INIT(ldr)                          2273d  2372d  2379c
  M68KSYS(ci)                          55d    72d    73d    93d    95c   100d   102c   602c   620c   621c   666c   667c   693c   703c
                                      728c  1010c  1012c  1112c  1118c  1120c  1130c
  MINIT(xminit)                      1395d  1511c  1512c
  SEGMENTER(asm)                      175d   193c   230d   251c   255d   275c   278d   286c   289d   313c   316d   330c
  SEGMENTER(segmenter)                 27d    28d    32d    33d
filename1
  ETU                                 844d   878c   879c   880c   881c   882c   893c   918c
fileopen
  ETU                                  59d   401c   406c   886c   905c   927c  1228c
fileptr
  INITLOAD(loader)                    874d   904d
filer
  M68KSYS(ci)                          39d   721c  1120c
filesize
  LIFDAM(lifmodule)                   387d   437c   442c   443c   450c   451c
filestart
  LIFDAM(lifmodule)                  1017d  1021c  1038c  1045c
filetogo
  INIT(ldr)                          2274d  2406d  2415c  2434c  2435c  2444c  2445c  2501c
  INITLOAD(loader)                   1093d  1583d  1608c  1615c  1616c  1617c  1618c  1619c
  M68KSYS(ci)                         912d   918c   936c
fill
  SRM_DRV(srm)                 ***    250d
fill1
  SRM_DRV(srm)                 ***    207d
fill2
  SRM_DRV(srm)                 ***    208d
fill_index_color
  GLE_GEN(gle_gen)             ***   2192c
  GLE_HPGL(gle_hpgl_out)       ***   7552c
  GLE_RGL(gle_ras_out)                8415c  8481c
  GLE_TYPES(gle_types)         ***   1038d
fillchar
  MOREFSYS(mfs)                ***    649c
  SRM_DRV(srm)                 ***   1426c
fillcount
  CRT(crt)                     ***    128d
  GCRT(crtb)                   ***     61d
  SYSDEVS(sysdevs)             ***     81d
filler
  LIFDAM(lifmodule)                    55d   522c
  MOUSE(mouse)                 ***     44d
  SRM_DRV(srm)                        292d   317d

filler1
  SRM_DRV                      ***   1932c
  SRM_DRV(srm)                        269d   336d   375d   384d   397d   408d   652d   722d  1496c  1522c  1567c  1604c
filler2
  SRM_DRV                      ***   1928c
  SRM_DRV(srm)                        378d   401d   648d  1571c  1609c
filler3
  SRM_DRV                            1929c  1955c  2012c  2013c  2098c  2099c
  SRM_DRV(srm)                        612d   649d   679d   714d   804d
filler4
  SRM_DRV                      ***   1931c
  SRM_DRV(srm)                 ***    651d
filler5
  SRM_DRV(srm)                        459d  1693c
filler8
  SRM_DRV                      ***   2102c
  SRM_DRV(srm)                 ***    807d
filname
  INITLOAD(loader)                    820d   838d
  LIBRARIAN                          2894c  3270d  3299d  3305d
find
  M68KSYS(ci)                         587d   596c   603c   610c   613c   620c
find_proc
  SEGMENTER(asm)                      109d   116c   133c   281c   284c
  SEGMENTER(segmenter)         ***     35d
findblock
  UCSD_DAM(ucsdmodule)                326d   335c   343c
findcard
  ETU                                 696d   762c  1234c
finddup
  UCSD_DAM(ucsdmodule)                323d   336c   342c
findem
  M68KSYS(ci)                         590d   611c   616c
findfile
  LIFDAM(lifmodule)                   630d   658c   671c   975c   980c   998c  1068c  1073c  1089c  1110c
  UCSD_DAM(ucsdmodule)                323d   328c   345c   357c   400c   408c
findmod
  LIBRARIAN                          3270d  3390c  3416c  3447c  3485c
findold
  UCSD_DAM(ucsdmodule)                264d   267c   279c   287c   306c   432c   444c   448c
findunit
  INIT(fs)                     ***    761d
findvolume
  ETU                                 235c   248c   319c
  INIT(fs)                     ***    759d
  INIT(ldr)                    ***   2317c
  MIUI                         ***    142c
  TAPEBKUP                     ***   1072c
findx
  SRMDAM(srmdammodule)               1522d  1534c  1543c  1544c  1546c  1548c  1551c  1589c
finfo
  ETU                                 219d   226c   352d   359c   364c
finish_kbd
  A804XDVR(a804xdvr)           ***    190d
  NONUSKBD2                           34d    36d   128c   152c
  SYSDEVS(sysdevs)             ***    146d
finish_polygon
  DGL_POLY(dgl_poly)                21011d 21058c 21103c 21134c 21165c
finishboot
  LIBRARIAN                          3509d  3624c
finishfib
  LIFDAM(lifmodule)                   913d   947c   955c
```

```
finitb
  ETU                                ***    923c
  INIT(fs)                           ***    763d
  INIT(ldr)                          ***   2378c
fintdata
  MIUI                               ***    334c
finx
  UCSD_DAM(ucsdmodule)                      184d   195c   199c   239c   254c   379d   387c
first
  LIBRARIAN                                2903d  2912c  2913c
first_extent
  SRM_DRV(srm)                              498d  1766c
first_harddisc_lun
  CTABLE                                   1522c  1813c
  CTABLE(options)                            70d   200d
first_index
  DGL_POLY(dgl_poly)                      20429d 20660c 20677c
first_lun
  CTABLE(ctr)                               308d   887d   893c   896c   897c   907c   909c   912c
first_lun_ok
  CTABLE(ctr)                               889d   892c   894c   910c
firstbytes
  INIT(misc)                                538d   595c   596c   597c   599c   600c   601c   602c   603c   638c   639c   640c   642c   646c
                                            649c   659c   664c   668c   675c   677c
firstchar
  MOREFSYS(mfs)                              56d    69c    80c   101c   103c   378c
firstline
  LIBRARIAN                                 424d  1249c  1255c  1261c  1262c  1263c
firstmodname
  LIBRARIAN                                3299d  3312c  3327c  3328c  3339c  3435c  3438c  3452c  3460c  3471c
firstmodnum
  LIBRARIAN                                3300d  3313c  3314c  3329c  3336c  3424c  3426c  3440c  3453c  3460c  3472c  3493c  3497c
firstoutblock
  LIBRARIAN                          ***    69d
firstpos
  INIT(misc)                                538d   594c   595c   598c   600c   636c   637c   639c   646c   649c
firstptr
  LIBRARIAN                                1478d  1499c  1604c  1609c  1613c  1618c  1624c  1625c
firsttime
  GLE_HPIB(gle_hpib_io)                   10020d 10248c 10336c
  IOLIB(hpib_1)                             424d   570c   658c
firstword
  SRM_DRV(srm)                       ***    217d
fisafit
  SRMDAM(srmdammodule)                     1636d  1641c  1668c  1801c
fisnew
  INIT(misc)                         ***    467c
  INITLOAD(bootdammodule)            ***    712c
  INITLOAD(sysglobals)               ***     82d
  LIFDAM(lifmodule)                         287c   917c   919c   969c   970c   975c   980c   995c  1088c  1110c  1127c  1128c
  SRMDAM(srmdammodule)                      141c  1272c  1283c  1289c  1300c  1304c  1351c  1453c  1721c  1734c  1759c
  UCSD_DAM(ucsdmodule)                       76c   249c   290c   312c   344c   408c   422c
fistexivar
  GLE_KNOB(gle_knob_in)              ***  18038c
  INIT(misc)                                469c   472c   682c
  INITLOAD(sysglobals)               ***     79d
  LIFDAM(lifmodule)                  ***    922c
  SRMDAM(srmdammodule)                      919c   949c
  UCSD_DAM(ucsdmodule)               ***    165c
  UNITIO(uio)                        ***     45c
fixed
  LIFDAM(lifmodule)                         678d   741c   751c   836c   867c   868c   885c   891c   893c   898c
```

```
fixed_vol_byte
  MINIT(qminit)                             944d  1016c  1017c
  TAPEEKUP(cs80tbdvr)                       463d   492c   493c
fixed_volume_byte
  CS80(cs80dsr)                             783d   935c   936c
fixed_word_array
  GLE_FGL(gle_ras_out)               ***   8095d
fixlock
  ETU                                        91d   826c  1227c  1271c
fixname
  INIT(fs)                           ***    754d
  LIBRARIAN                                2930c  3082c  3140c  3148c
  M68KSYS(ci)                               103c   665c   996c  1046c
fk
  INIT(misc)                                695d   703c   705c   706c   707c
  INITLOAD                                 1703d  1707c
  INITLOAD(bootdammodule)                   610d   653d   669c   670c   708c   709c
  LIFDAM(lifmodule)                         135d   145c   147c   148c   152c   164d   168c   169c   176c   180c   187c   188c   540d   542c
                                            543c   544c   952d   954c
  M68KSYS(ci)                               400d   540c
  SRMDAM(srmdammodule)                       88d    91c    92c    93c
fkind
  ETU                                       320c   903c   930c
  INIT(fs)                           ***    758d
  INIT(misc)                         ***    473c
  INITLOAD(bootdammodule)                   653d   668c   670c   671c   673c   707c   709c   714c
  INITLOAD(loader)                   ***   1141c
  INITLOAD(sysglobals)               ***     78d
  LIBRARIAN                                2923d  2931c
  LIFDAM(lifmodule)                         922c   954c
  MIUI                                      128d   139c
  SRMDAM(srmdammodule)                      935c   949c  1451c
  TAPEEKUP                                 1058d  1069c
  UCSD_DAM(ucsdmodule)                      166c   212c   247c   291c   294c   313c
fkindIC
  INITLOAD(sysglobals)               ***     54d
fkind11
  INITLOAD(sysglobals)               ***     54d
fkind12
  INITLOAD(sysglobals)               ***     54d
fkind13
  INITLOAD(sysglobals)               ***     55d
fkind14
  INITLOAD(sysglobals)               ***     55d
fkind7
  INITLOAD(sysglobals)               ***     53d
fkind8
  INITLOAD(sysglobals)               ***     53d
fkind9
  INITLOAD(sysglobals)               ***     53d
flag
  EPROMS(eproms)                            27d    88c    89c
flagit
  CS80(cs80dvr)                            1088d  1116c
flast
  UCSD_DAM(ucsdmodule)                      185d   195c   200c   239c   245c   246c
```

```
flastpos
  ASCII(asciimodule)                  21c   22c   37c   49c   50c   59c   70c   74c   85c   94c  114c  115c  116c  159c
                                    203c  223c  251c  264c  265c  295c  305c
  ETU                               325c 1035c
  INIT(misc)                        546c  548c  559c  560c  564c  589c  594c  625c  626c  663c
  INITLOAD(sysglobals)        ***    95d
  LIFDAM(lifmodule)           ***   297c
  SRMDAM(srmdammodule)        ***  1504c
  UCSD_AM(ucsd_am)                   64c   75c  139c  142c  153c  161c  208c  260c
  UCSD_DAM(ucsdmodule)        ***    83c
fleof
  ASCII(asciimodule)                  28c   77c   78c   80c   82c   83c   95c  132c  157c  200c  221c  255c  273c  283c
                                    299c
  ETU                         ***   931c
  INIT(ldr)                        2300c 2308c
  INIT(misc)                        466c  492c  501c  546c  574c  588c  611c  622c
  INITLOAD(bootdammodule)     ***   712c
  INITLOAD(sysglobals)        ***    92d
  LIFDAM(lifmodule)                 295c  409c  917c  918c  960c 1004c 1006c 1092c
  M68KSYS(ci)                 ***    90c
  SRMDAM(srmdammodule)              933c 1279c 1347c 1412c 1488c 1508c
  UCSD_AM(ucsd_am)                   53c   60c  106c  162c  219c
  UCSD_DAM(ucsdmodule)               81c  101c  250c  294c  295c  315c  416c  418c  495c  496c
flimit
  HEAPT(hpm)                        135d  166c  167c  172c  174c  179c
flistptr
  INITLOAD(sysglobals)        ***    74d
flockable
  INITLOAD(sysglobals)        ***   114d
  LOCKMOD(lockmodule)                41c   66c   82c
  SRMAM(srmammodule)                193c  200c
  SRMDAM(srmdammodule)              936c 1277c 1345c
flocked
  ETU                         ***   326c
  INITLOAD(sysglobals)        ***   115d
  LOCKMOD(lockmodule)                44c   69c   85c
  SRMAM(srmammodule)                193c  200c
  SRMDAM(srmdammodule)              937c 1277c 1345c 1489c 1512c
floppy_unit_pairs
  CTABLE                      ***  1513c
  CTABLE(options)                    68d   71d
flpt_present
  MOREFSYS                          687c  689c
  MOREFSYS(mfs)               ***    16d
flpy_flags
  CTABLE                           1193d 1195d 1270c 1301c 1302c 1324c 1327c 1328c
  CTABLE(brstuff)             ***   933d
  CTABLE(ctr)                 ***   279d
flpy_flags_type
  CTABLE(ctr)                       265d  272d  274d  279d
flpy_msus
  CTABLE                           1203d 1270c 1321c 1322c 1442c
flpy_wrt
  INITLOAD(mini)              ***   446c
flpyinit
  MINIT(mminit)               ***   162c
flpymread
  INITLOAD(mini)              ***   426c
flpymwrite
  INITLOAD(mini)              ***   428c
flpyread
  INITLOAD(mini)              ***   440c
fltpt
  INITLOAD(loader)                  857d  934d
```

```
flun
  MIUI                               32d   34c  222d  233c  234c  235c  237c  247c  249c  374d  512c  513c  514c  519c
                                    521c  527c  529c  533c
  TAPEBKUP                         1109d 1120c 1121c 1122c 1124c 1134c 1136c
flush
  AMIGO(amigodvr)             ***   589c
  ASCII(asciimodule)                137c  298c
  BUBBLES(bubble)             ***   106c
  CRT(crt)                          101d  147d  414c
  CS80(cs80dvr)               ***  1375c
  EPROMS(eproms)              ***    77c
  F9885(f9885dvr)             ***   274c
  GCRT(crtb)                         80d  284c
  INIT(misc)                        505c  576c  578c
  INITLOAD(mini)              ***   502c
  INITLOAD(sysglobals)        ***    66d
  KEYS(keys)                  ***    82c
  LIFDAM(lifmodule)                 317c  319c  469c
  M68KSYS(ci)                 ***   736c
  MIUI                        ***   495c
  PRINTER(prtdvr)             ***   295c
  SRMAM(srmammodule)          ***   210c
  SRMDAM(srmdammodule)        ***  1503c
  SYSDEVS(sysdevs)            ***    95d
  UCSD_AM(ucsd_am)                  173c  176c
  UCSD_DAM(ucsdmodule)              155c  500c  623c  676c
  UNITIO(uio)                 ***    67c
flush_buffer
  GLE_GEN(gle_gen)            ***  2167c
  GLE_HPGL(gle_hpgl_out)      ***  7559c
  GLE_RGL(gle_ras_out)        ***  8424c
  GLE_TYPES(gle_types)        ***  1039d
  SRM_DRV                     ***  2103c
  SRM_DRV(srm)                ***   808d
flush_tapebuf
  CS80(cs80dvr)                    1144d 1175c 1277c 1305c 1321c 1379c
flushbuffer
  ASCII(asciimodule)                 32d   48c   73c   76c  136c
  INIT(misc)                        541d  558c  577c  662c
  UCSD_AM(ucsd_am)                  134d  175c  217c
flushdir
  LIFDAM(lifmodule)                 313d  475c  535c  664c  945c 1009c 1053c 1078c 1170c
flushindent
  UCSD_AM(ucsd_am)                  123d  139c  198c
flushit
  M68KSYS(ci)                        70d  571d  574c
fmaketype
  INIT(fs)                    ***   768d
  LIBRARIAN                        3142c 3149c
fmaxpos
  INIT(fs)                    ***   774d
fminit
  MIUI                        ***    12d
fminitialize
  MIUI                        ***   350c
fmodified
  ASCII(asciimodule)                 77c   95c
  ETU                               323c  404c
  INIT(misc)                        501c  622c
  INITLOAD(sysglobals)        ***    88d
  LIFDAM(lifmodule)                 290c  919c 1001c 1130c
  SRMDAM(srmdammodule)             1276c 1323c 1344c 1455c 1507c
  UCSD_AM(ucsd_am)            ***    60c
  UCSD_DAM(ucsdmodule)               79c  372c  641c
```

```
fmt
  AMIGO(csamigo)                  ***    110d
  CTABLE(brstuff)                        955d  1001c
fmt_type
  CTABLE(brstuff)                        945d   955d
fname
  INIT(fs)                        ***    757d
  INITLOAD(bootdammodule)                559d   588d   648d   655c   660c   662c
  LIB(dgl_lib)                         20021d 21193d 21212c 21213c 21232c
  LIFDAM(lifmodule)                       60d   609c   648c   935c   936c  1077c  1165c
fnh
  SRM_DRV(srm)                           1512d  1518c
fnlength
  INITLOAD(loader)                        815d   820d
  LIBRARIAN                       ***    2895c
  UCSD_DAM(ucsdmodule)                    174c   440c
fnosrmtemp
  ETU                             ***    326c
  INITLOAD(sysglobals)            ***    109d
  SRMDAM(srmdammodule)                   141c  1091c  1099c  1117c  1156c  1283c  1289c  1361c  1390c  1457c  1722c  1735c
fns
  SRMDAM(srmdammodule)                   120d   122c   463d   478c   492c   506d   521c   529c   572d   588c   966d   975c   980c   994d
                                        1001c  1007c  1249d  1253c  1257c  1334d  1360c  1367c  1392c  1397c  1404c  1419c  1420c  1424c
fns1
  SRMDAM(srmdammodule)                  1178d  1187c  1202c  1217d  1226c  1235c
fns2
  SRMDAM(srmdammodule)                  1181d  1189c  1203c  1220d  1230c  1236c
fo
  MINIT(cs80ir)                          600d   612c   613c   614c   615c   616c
foldfileid
  INITLOAD(sysglobals)            ***    121d
  SRMDAM(srmdammodule)                  1080c  1101c  1102c  1112c  1154c  1284c  1366c  1376c  1380c  1381c
font
  GLE_HPGL(gle_hpgl_out)          ***   7561c
  GLE_RGL(gle_ras_out)            ***   8427c
  GLE_STEXT(gle_stext)            ***   4008d
foptstring
  ETU                             ***    321c
  INITLOAD(sysglobals)            ***    105d
  SRMDAM(srmdammodule)                   194d   224c   225c   226c   229c   231c   234c   244c   245c   262c   525c   976c  1060c  1456c
format
  ETU                                    849d   912c   930c  1041c  1088c  1126c
  MINIT(asmr)                           1335d  1346d
  MINIT(hminit)                          359c   398c   424c
  MINIT(iramigo)                         192d   221d
  MINIT(xminit)                         1628c  1722c  1757c  1798c
format_cmd
  AMIGO(csamigo)                  ***     57d
  MINIT(iramigo)                  ***    239c
format_cmd_buf
  MINIT(iramigo)                         223d   232c   239c
format_media
  SRM_DRV(srm)                    ***    610d
formatting_option
  MINIT(cs80ir)                          559d   590d   621c
  MINIT(qminit)                          978c  1150c
formatting_option_byte
  MINIT(qminit)                          892d   975c   978c   982c  1091c  1150c
formatting_option_supported
  MINIT(qminit)                          893d   911c   972c   981c  1088c  1147c
formfeed
  PRINTER(prtdvr)                 ***     61d
fortranflag
  LIBRARIAN                              305d   335c  1231c  1439c
```

```
forwardpatches
  LIBRARIAN                               81d  1668c  1677c  2423c  2606c
fosize
  LIBRARIAN                             2830d  2831d  3076d  3077d
found
  CTABLE                                1293d  1298c  1299c  1300c
  CTABLE(ctr)                            386d   389c   391c   395c   397c   401c
  GCRT(crtb)                             532d   537c   543c   565c   567c
  LIFDAM(lifmodule)                      136d   632d   637c   646c   651c   652c   656c   657c   658c  1016d  1020c  1025c  1037c  1044c
                                        1047c
  MINIT(xminit)                         1412d  1789c  1790c
  SEGMENTER(asm)                         113d   117c   119c   123c   125c   129c
  UCSD_DAM(ucsdmodule)                   265d   271c   273c   277c   279c   324d   334c   336c   345c   348c
found_fill_line_on_edge
  DGL_POLY(dgl_poly)                  20431d 20711c 20718c 20755c
found_gator
  DGL_C_OUT(dgl_confg_out)            11025d 11079d 11092c 11100c 11106c 11117c 11122c 11131c 11268c
foundlastptr
  LIBRARIAN                             2363d  2459c  2468c  2481c
foundpatchdelta
  LIBRARIAN                             2367d  2458c  2467c  2489c  2491c  2507c
foundpatchmodptr
  LIBRARIAN                             2361d  2460c  2469c  2477c
foverfile
  INIT(fs)                        ***    769d
foverprint
  INIT(fs)                        ***    796d
foverwritten
  INITLOAD(sysglobals)            ***    112d
  SRMDAM(srmdammodule)                  1362c  1377c  1386c  1736c
fp
  AMIGO(amigodvr)                        430d   563c   587c   600c   601c   610c   624c
  ASCII(asciimodule)                       6d    13d    20c    28c    35c    37c    46c    50c    57c    66c    81c    91c   103c   127c
                                         137c   155c   166c   179c   216c   237c   262c   283c
  BUBBLES(bubble)                         29d    53d    61c
  CRT(crt)                               402d   412c   413c   415c   423c   428c   439c   442c   443c   446c   496c
  CS80(cs80dvr)                          987d  1214d  1340c  1357c  1373c  1392c  1395c
  EPROMS(eproms)                           8d    12d    46c    52c    81c    82c   104c
  F9885(f9885dvr)                         40d   254c   272c   284c   286c   298c
  GCRT(crtb)                             272d   282c   283c   285c   293c   298c   309c   312c   313c   316c   364c
  GLE_KNOB(gle_knob_in)           ***  18031c
  INIT(initunits)                       2139d  2148d  2152d  2156d  2228d
  INIT(misc)                             483d   487c   498c   503c   505c   515d   519c   521c   534d   543c   548c   556c   564c   571c
                                         578c   582c   606c   617c
  INITLOAD(loader)                ***    894d
  INITLOAD(mini)                         309d   384d   481c   500c   512c   514c   526c
  INITLOAD(sysglobals)                    68d    70d
  KEYS(keys)                              72d    83c    94s   127c   136c
  M68KSYS(ci)                            306d   332c   335c   344c   351c   352c   383c   731d   734c
  PRINTER(prtdvr)                         40d    47d   271c
  SRMAM(srmammodule)                      39d   177d   187c   196c   203c   207c
  SYSDEVS(sysdevs)                       372d   376d   407d   432d   434c   484d   486c
  UCSD_AM(ucsd_am)                        40d    50c    57c    73c    91c   106c   114c   124c   137c   150c   159c   165c   171c   176c
                                         180c
  UCSD_DAM(ucsdmodule)                   572d   575c   579c   604c   606c
  UNITIO(uio)                     ***     38c
fpage
  INIT(fs)                        ***    795d
```

```
fpeof
  AMIGO(amigodvr)                ***   610c
  ASCII(asciimodule)                    21c    78c    82c   132c
  BUBBLES(bubble)                       74c    91c
  CS80(cs80dvr)                  ***   1392c
  EPROMS(eproms)                 ***    54c
  ETU                            ***   909c
  F9885(f9885dvr)                ***   284c
  INIT(misc)                            267c   269c   272c   454c   455c   457c   462c   466c   496c   499c   615c   618c
  INITLOAD(bootdammodule)               705c   712c   757c
  INITLOAD(mini)                 ***   512c
  INITLOAD(sysglobals)           ***    93d
  LIFDAM(lifmodule)                     248c   294c   295c   409c   593c   595c   606c   607c   916c   918c  1054c
  MINIT(bminit)                  ***    76c
  MIUI                           ***   489c
  SRMDAM(srmdammodule)                  481c   527c   537c   557c   586c   598c   640c   932c  1322c  1487c
  UCSD_AM(ucsd_am)                       55c    58c
  UCSD_DAM(ucsdmodule)                   81c   100c   101c   102c   163c   294c   295c   371c   496c   536c   563c   585c
fphydata
  MIUI                           ***   475c
fpos
  ASCII(asciimodule)                     28c    60c    77c    80c    83c    93c    94c    95c   107c   112c   120c   131c   157c   158c
                                        159c   160c   168c   200c   201c   202c   203c   204c   208c   221c   222c   223c   224c   228c
                                        241c   247c   248c   251c   252c   255c   266c   273c   274c   299c
  ETU                                   322c   436c   458c   479c  1035c
  INIT(misc)                            491c   492c   496c   499c   501c   574c   587c   592c   594c   602c   610c   611c   613c   615c
                                        618c   622c   639c
  INITLOAD(sysglobals)           ***    91d
  LIFDAM(lifmodule)                     296c   593c   931c  1022c
  M68KSYS(ci)                    ***    90c
  SRMDAM(srmdammodule)                  526c   585c  1056c  1057c  1318c  1413c  1414c  1415c  1450c
  UCSD_AM(ucsd_am)                       52c    53c    55c    58c    60c    62c    64c    94c    97c   116c   118c   139c   153c   161c
                                        162c   165c   184c   202c   204c   208c   218c   221c   237c   240c   246c   252c   260c
  UCSD_DAM(ucsdmodule)                   82c   217c   366c   368c   535c
fposition
  INIT(fs)                       ***   773d
fpurgeoldlink
  INITLOAD(sysglobals)           ***   111d
  SRMDAM(srmdammodule)           ***  1237c
fput
  INIT(fs)                       ***   778d
frac
  GEN(dgl_gen)                         3169d  3175c  3177c  3178c
fracstate
  MOREFSYS(mfs)                         194d   244c   282c   322c   361c
fraction_adjust
  GLE_STEXT(gle_stext)                 4021d  4023d  4024d  4047c  4048c  4051c  4052c
frame_buffer
  DGL_C_OUT(dgl_confg_out)            11028d 11080d 11110c 11134c 11258c 11269c
frameaddr
  GCRT(crtb)                            159c   473c
  SYSDEVS(sysdevs)               ***    132d
fread
  INIT(fs)                       ***   783d
freadable
  INIT(ldr)                            2296c  2304c
  INITLOAD(sysglobals)           ***    83d
  LIFDAM(lifmodule)                     288c   315c  1122c
  M68KSYS(ci)                    ***    90c
  MOREFSYS(mfs)                  ***   520c
  UCSD_DAM(ucsdmodule)                   77c   630c   675c
freadbool
  INIT(fs)                       ***   813d
```

```
freadbytes
  INIT(fs)                       ***   786d
freadchar
  INIT(fs)                       ***   802d
freadenum
  INIT(fs)                       ***   811d
freadint
  INIT(fs)                       ***   804d
freadln
  INIT(fs)                       ***   800d
freadmode
  INITLOAD(sysglobals)           ***    85d
  LIFDAM(lifmodule)              ***   289c
  UCSD_DAM(ucsdmodule)           ***    78c
freadpaoc
  INIT(fs)                       ***   810d
freadreal
  MOREFSYS(mfs)                          18d   515d
freadstr
  INIT(fs)                       ***   808d
freadstrbool
  INIT(fs)                       ***   836d
freadstrchar
  INIT(fs)                       ***   817d
freadstrenum
  INIT(fs)                       ***   832d
freadstrpaoc
  INIT(fs)                       ***   830d
freadstrreal
  MOREFSYS(mfs)                          21d   533d
freadstrstr
  INIT(fs)                       ***   825d
freadstrword
  INIT(fs)                       ***   820d
freadword
  INIT(fs)                       ***   803d
frecsize
  INITLOAD(sysglobals)           ***    76d
free_blocks
  SRM_DRV(srm)                   ***   345d
freearray
  HEAPT(hpm)                             75d    79d   123d
freelist
  HEAPT(hpm)                             79d   123d   145c   147c   148c   185c   186c   205c   206c   223c   226c   233c   236c   241c
                                        252c
french_kbd
  A804XDVR(a804xdvr)                    191d   193d
  NONUSKBD1(non_us_kbd1)                 77d   117c   196c
  NONUSKBD2                             155c   216c
  SYSDEVS(sysdevs)               ***    149d
freptcnt
  ASCII(asciimodule)                    105c   112c   115c   116c   120c   129c   168c   169c   170c   171c   184c   185c   187c   192c
                                        196c   202c   204c   207c   218c   222c   224c   226c   239c   248c   252c   295c   305c
  ETU                            ***   322c
  INITLOAD(sysglobals)           ***    96d
  LIFDAM(lifmodule)              ***   297c
  UCSD_AM(ucsd_am)                      125c   127c   128c   129c   130c   192c   196c   199c   201c   229c   230c   248c
  UCSD_DAM(ucsdmodule)           ***    83c
frequency
  A804XDVR(a804xdvr)                    157d   160c
  SYSDEVS(sysdevs)                      367d   416d   417c
```

```
from_memory
  DC_DRV(intdc)                    426c   440c   530c
  DISCHPIB(dischpib)        ***    458c
  IOLIB(general_4)                1496c  1536c  1570c
  KERNEL(iodeclarations)    ***    565d
  SRM_DRV(srm)                    1309c  1393c
frompos
  LIFDAM(lifmodule)                386d   434c   435c   444c   445c   474c   476c   477c
fs
  CTABLE                    ***   1186d
  ETU                       ***     26d
  INIT                      ***   2514d
  INIT(fs)                  ***    737d
  INIT(initunits)           ***   2135d
  INIT(ldr)                 ***   2270d
  LIBRARIAN                 ***     27d
  LIFDAM(lifmodule)         ***     26d
  M68KSYS(ci)               ***     33d
  MINIT(mminit)             ***    111d
  MIUI                      ***     11d
  MOREFSYS(mfs)             ***     10d
  PRINTER(prtdvr)           ***     37d
  TAPEBKUP                  ***    953d
fsavepathid
  INITLOAD(sysglobals)      ***    113d
  SRMDAM(srmdammodule)             868c   870c  1205c  1239c  1240c  1305c  1382c  1428c  1433c  1678c  1696c  1707c  1716c  1728c
                                  1744c  1754c  1774c  1805c
fseek
  INIT(fs)                  ***    772d
fsegs
  INIT(fs)                  ***    758d
  LIBRARIAN                       2922d  2931c  2935c
  MIUI                             127d   139c   155c
  TAPEBKUP                        1057d  1069c
fsidc
  INITLOAD(sysglobals)      ***     12d
  M68KSYS(ci)               ***    188d
fsidctype
  INITLOAD(sysglobals)             10d    12d
fsize
  HEAPT(hpm)                       136d   165c   166c   179c   180c   181c   182c   183c
  INITLOAD(bootdammodule)          562d   674c
  LIBRARIAN                        619d   621c   623c   626d   642c  2831d  2840c  2841c  2842c  3077d  3093c  3094c  3095c
  LIFDAM(lifmodule)                 63d   425c   437c   453c   611c   860c   916c   939c  1007c  1023c  1052c  1054c
  UCSD_DAM(ucsdmodule)             186d   195c   197c   198c   219c   234c   237c   238c   239c   246c
fsp
  INIT(misc)                ***    171d
  KEYS(keys)                       362c   486c
fstart
  INITLOAD(bootdammodule)          561d   711c
  LIFDAM(lifmodule)                 62d   425c   434c   438c   614c   646c   855c   860c   915c   938c  1021c  1038c
fstartaddress
  ETU                       ***    324c
  INITLOAD(bootdammodule)   ***    705c
  INITLOAD(sysglobals)      ***     99d
  LIBRARIAN                       2756c  2760c
  LIFDAM(lifmodule)                958c   961c  1005c
  SRMDAM(srmdammodule)             944c   947c  1049c  1088c  1125c  1136c  1148c  1422c
  UCSD_DAM(ucsdmodule)             212c   292c
fstrg
  MOREFSYS(mfs)                     46d    92c    95c    97c
fstripname
  INIT(fs)                  ***    753d
```

```
ftcb
  AMIGO(csamigo)                   383d   388c   396d   401c
  MINIT(iramigo)                   200d   205c   225d   239c
ftcb_type
  AMIGO(amigodvr)           ***    715d
  AMIGO(csamigo)                    61d   144d   240d   298d   306d   330d   383d   396d   407d
  MINIT(iramigo)                   200d   210d   225d
ftid
  ETU                       ***    911c
  INIT(misc)                       440c   443c
  INITLOAD(bootdammodule)          697c   699c   700c
  INITLOAD(loader)          ***   1437c
  INITLOAD(sysglobals)      ***    102d
  LIBRARIAN                 ***   3018c
  LIFDAM(lifmodule)                223c   636c   936c  1072c  1077c  1165c
  SRMDAM(srmdammodule)             123c   127c   144c   715c   857c   858c  1272c  1458c  1607c  1608c  1669c  1670c  1679c  1806c
  UCSD_DAM(ucsdmodule)             177c   277c   342c   414c   443c   669c
ftitle
  ETU                              316c   375c   894c   922c
  INIT(fs)                         756d   757d
  INIT(misc)                       437c   443c   444c
  INITLOAD(bootdammodule)          696c   697c   699c   700c   705c
  INITLOAD(loader)          ***   1608c
  INITLOAD(sysglobals)      ***    128d
  LIBRARIAN                       2921d  2931c
  LIFDAM(lifmodule)                222c   223c   394c   488c  1162c  1164c
  M68KSYS(ci)               ***    820c
  MIUI                             126d   139c   140c   531c
  SRMDAM(srmdammodule)             674c   678c   689c   698c   714c   722c   735c   749c   752c   763c   774c   781c   794c   858c
                                   885c  1541c  1544c  1546c  1548c  1551c  1609c  1610c  1682c  1685c  1810c
  TAPEBKUP                        1056d  1069c  1070c
  UCSD_DAM(ucsdmodule)             174c   177c   476c   514c   577c   653c   665c   669c   670c
ftype
  INIT(fs)                  ***    767d
  INITLOAD(bootdammodule)          560d   589d   609d   649d   655c   668c   670c   704c   705c   709c
  LIFDAM(lifmodule)                 61d   368c   370c   374c   419c   421c   428c   465c   534c   598c   601c   610c   612c   642c
                                   644c   649c   664c   696c   712c   807c   820c   937c   954c   959c   960c  1004c  1033c  1035c
  SRMDAM(srmdammodule)              86d    92c
  SRM_DRV(srm)                    1049d  1736d  1762c
ftypeCode
  LIFDAM(lifmodule)                630d   649c
fubuffer
  INITLOAD(mini)                   412d   440c   441c   445c   446c
fudged_d
  MIUI                             319d   322c   323c   351c
```

```
funit
  AMIGO(amigodvr)              ***    563c
  ASCII(asciimodule)                   35c    46c    81c   137c
  BUBBLES(bubble)              ***     61c
  CS80(cs80dvr)                ***   1357c
  CTABLE(ctr)                  ***    841c
  EPROMS(eproms)                       46c    59c    81c    82c   104c
  ETU                                 248c   250c   257c   260c   270c   276c   280c   319c   341c   364c   366c   370c   392c   405c
                                      434c   898c   904c   926c  1052c
  F9885(f9885dvr)              ***    254c
  GLE_KNOB(gle_knob_in)        ***  18039c
  INIT(misc)                          268c   487c   498c   519c   548c   564c   578c   617c
  INITLOAD(loader)                   1390c  1391c  1438c  1609c
  INITLOAD(mini)               ***    481c
  INITLOAD(sysglobals)         ***    117d
  LIFDAM(lifmodule)                   407c   440c   469c  1122c
  LOCKMOD(lockmodule)                  49c    72c    90c
  M68KSYS(ci)                         344c   734c   820c   962c
  MINIT(bminit)                ***     75c
  MIUI                                487c   527c
  PRINTER(prtdvr)              ***    271c
  SRMAM(srmammodule)                   67c    75c    78c    84c    85c    89c    92c   114c   122c   125c   131c   132c   136c   139c
                                      149c   154c   155c   157c   171c   188c
  SRMDAM(srmdammodule)                163c  1279c  1319c  1347c  1447c
  SWVOL                        ***     22c
  UCSD_AM(ucsd_am)                     57c    91c   165c   176c
  UCSD_DAM(ucsdmodule)                 80c   160c   329c   490c   585c
  UNITIO(uio)                  ***     46c
fvb
  CS80(cs80)                   ***    109d
  CS80(cs80dsr)                ***    935c
  CS80(cs80dvr)                ***   1066c
  MINIT(qminit)                ***   1016c
  TAPEBKUP(cs80tbdvr)          ***    492c
fvid
  ETU                                 173d   179c   181c   182c   281c   316c   318c   319c   364c   898c   899c   914c
  INIT(fs)                            757d   759d
  INITLOAD(sysglobals)         ***    100d
  LIBRARIAN                          2920d  2931c
  LIFDAM(lifmodule)                   309c  1108c
  M68KSYS(ci)                  ***    820c
  MIUI                                125d   139c   142c   207c   225d   249c   250c   251c   529c
  SRMDAM(srmdammodule)                489c   524c   584c   718c   728c   979c  1006c  1040c  1201c  1233c  1234c  1256c  1299c  1358c
                                     1673c  1674c  1681c  1808c
  TAPEBKUP                           1055d  1069c  1072c  1096c  1097c  1112d  1136c  1137c  1138c
  UCSD_DAM(ucsdmodule)         ***    130c
fwaitonlock
  INITLOAD(sysglobals)         ***    110d
  LOCKMOD(lockmodule)                  48c    89c
  SRMDAM(srmdammodule)         ***   1477c
fwindow
  ETU                          ***    369c
  INITLOAD(sysglobals)         ***     73d
  LIFDAM(lifmodule)                  1133c  1152c  1153c  1155c
  M68KSYS(ci)                  ***    821c
  MIUI                         ***    528c
  SRMDAM(srmdammodule)                482c   539c   600c   996c  1193c  1194c  1195c  1228c  1230c  1234c  1236c  1240c  1692c
  UCSD_DAM(ucsdmodule)                439c   477c   511c   532c
fwrite
  INIT(fs)                     ***    784d
```

```
fwriteable
  INIT(ldr)                          2300c  2301c  2308c  2309c
  INITLOAD(sysglobals)         ***     83d
  LIFDAM(lifmodule)            ***    288c
  UCSD_DAM(ucsdmodule)         ***     77c
fwritebool
  INIT(fs)                     ***    814d
fwritebytes
  INIT(fs)                     ***    787d
  MOREFSYS(mfs)                ***    675c
fwritechar
  INIT(fs)                     ***    801d
fwriteenum
  INIT(fs)                     ***    812d
fwriteint
  INIT(fs)                     ***    806d
fwriteln
  INIT(fs)                     ***    781d
fwritepaoc
  INIT(fs)                     ***    809d
fwritereal
  MOREFSYS(mfs)                        19d   666d
fwritestr
  INIT(fs)                     ***    807d
fwritestrbool
  INIT(fs)                     ***    837d
fwritestrchar
  INIT(fs)                     ***    818d
fwritestrenum
  INIT(fs)                     ***    834d
fwritestrpaoc
  INIT(fs)                     ***    828d
fwritestrreal
  MOREFSYS(mfs)                        22d   559d
fwritestrstr
  INIT(fs)                     ***    826d
fwritestrword
  INIT(fs)                     ***    822d
fwriteword
  INIT(fs)                     ***    805d
fxpos
  CRT(crt)                             412c   413c
  GCRT(crtb)                           282c   283c
  INITLOAD(sysglobals)         ***     119d
fypos
  CRT(crt)                             412c   413c
  GCRT(crtb)                           282c   283c
  INITLOAD(sysglobals)         ***     120d
g
  DGL_RAS(dgl_raster)                17067d 17105c
  GEN(dgl_gen)                        3048d  3049d  3166d  3181c  3182c  3183c  3184c  3185c  3186c  3190d  3219c  3220c  3228c  3233c
                                      3237c
  GLE_KNOB(gle_knob_in)        ***   18210c
  GLE_RGL(gle_ras_out)                8156d  8186c  8190c  8194c  8198c  8202c
  HEART(hpm)                           135d   169c   170c   171c   172c   184c   187c
  LIBRARIAN                            153d   165c   166c   167c   170c   172c   175c   176c   185c   199c   204c
g_dummy
  DGL_C_OUT(dgl_confg_out)           11163d 11199c
g_on
  C_HOOK                               55d   121c   122c   131c
g_on36c
  DGL_C_OUT(dgl_confg_out)           11215d 11219c 11220c
g_ptr
  DGL_C_OUT(dgl_confg_out)           11162d 11196c 11197c 11199c
```

```
gai
  CTABLE(scanstuff)                  1063d  1120c
gamut
  DGL_HPGL(dgl_hpgl)                17099c 17107c 17115c
  DGL_INQ(dgl_inq)           ***     6330c
  DGL_POLY(dgl_poly)                20069c 20075c 20117c
  DGL_RAS(dgl_raster)               17108c 17178c 17189c 17211c 17232c 17625c
  GLE_HPGL(gle_hpgl_out)     ***     7686c
  GLE_TYPES(gle_types)       ***     1102d
  LIB(dgl_lib)               ***    20458c
gang_16bit_filler
  SRM_DRV(srm)                       203d   207d   250d   269d   652d   679d
gang_32bit_filler
  SRM_DRV(srm)                       202d   292d   317d   336d   375d   378d   384d   397d   401d   408d   612d   714d   722d   804d
gang_boolean_filler
  SRM_DRV(srm)                       201d   208d
gang_file_codes
  SRMDMM(srmdammodule)               86d   991d  1016d  1031d  1339d
  SRM_DRV(srm)                      212d   278d   495d   663d  1049d  1736d
gang_open_type
  SRMDMM(srmdammodule)       ***     647d
  SRM_DRV                    ***    1909d
  SRM_DRV(srm)                       224d   653d  1084d
gangcleanpack
  SRMAM(srmammodule)         ***     171c
  SRM_DRV                    ***    1864d
  SRM_DRV(srm)               ***    1072d
gator_clear
  GLE_RGL(gle_ras_out)               8009d  8223d  8480c
gator_display
  GLE_RGL(gle_ras_out)       ***     8075d
gator_fill_index_color
  GLE_RGL(gle_ras_out)               8146d  8481c
gatorcrtinit
  GCRT(crtb)                         497d   517c   569c
gatorcrttype
  DGL_C_OUT(dgl_confg_out)          11079d 11268c
  GCRT                       ***     583c
  GCRT(crtb)                          36d   527d   565c
gatorid
  DGL_C_OUT(dgl_confg_out)          11084d 11098c 11121c
  GCRT(crtb)                         528d   542c
gbase
  CRT(crt)                           207d   211c   212c   213c
  DGL_C_OUT(dgl_confg_out)          11216d 11221c 11224c 11225c 11226c
gbuffer
  CRT(crt)                           230d   234c   235c   236c   237c   240c   241c   246c   247c   254c   255c   256c
  C_HOOK                              78d    88c    89c    90c    91c    92c    93c   107c   109c
  DGL_RAS(dgl_raster)               17316d 17322c 17323c 17324c 17325c 17326c 17327c 17332c 17339c 17379d 17395c 17396c 17397c 17398c
                                    17399c 17400c 17401c 17415c 17417c
  GCRT(crtb)                          151d   163c   164c   165c   166c   167c   168c   169c   183c   185c
gbuffersize
  CRT(crt)                           220d   230d
  C_HOOK                              69d    78d
  DGL_RAS(dgl_raster)               17313d 17316d 17370d 17379d
  GCRT(crtb)                          142d   151d
gbyte
  C_HOOK                              72d    73d
  DGL_HPGL(dgl_hpgl)         ***    17189d
  DGL_RAS(dgl_raster)               17050d 17373d 17374d
  GCRT(crtb)                          145d   146d
  GEN(dgl_gen)               ***     3151d
  TYPES(dgl_types)           ***     1007d
```

```
gbytes
  LIBRARIAN                           125d   398c  1295c  1298c  1382c  1513c  1592c  1599c  1688c  1736c  1746c  1749c  1783c  1894c
                                     1896c  1902c  1920c  1922c  1929c  1945c  2022c  2032c  2074c  2382c  2385c  2410c  2479c  2522c
                                     2524c  2581c  2770c  2800c  2813c  2816c  2850c  3152c  3221c  3240c
gcb
  DGL_C_OUT(dgl_confg_out)          11007d 11059c 11153d 11174c 11218c 11231c 11250c 11265c 11351c 11399d 11414c 11444c 11486c 11502d
                                    11508c 11510c 11511c
  DGL_HPGL(dgl_hpgl)         ***    17261c
  DGL_HPGLI(dgl_hpgli)              18059c 18126c
  DGL_INQ(dgl_inq)                   6044c  6071c  6181c  6185c  6186c  6198c  6214c
  DGL_KNOB(dgl_knob)                18060c 18172c
  DGL_POLY(dgl_poly)                20062c 20114c 20134c 20150c 20176c 20208c 20592c 20885c 20942c 20987c 21016c 21059c 21104c 21135c
                                    21166c
  DGL_RAS(dgl_raster)               17175c 17207c 17258c 17286d 17291d 17297c 17592c
  DGL_VARS(dgl_vars)         ***     1246d
  GEN(dgl_gen)                       3319c  3347c  3413c  3473c  3610c
  GLE_GEN(gle_gen)                   2008d  2009d  2010d  2011d  2012d  2013d  2014d  2015d  2016d  2017d  2018d  2019d  2020d  2021d
                                     2022d  2023d  2024d  2025d  2026d  2027d  2028d  2029d  2030d  2031d  2032d  2033d  2034d  2035d
                                     2036d  2037d  2038d  2039d  2040d  2041d  2045d  2048c  2051d  2054c  2057d  2060c  2063d  2066c
                                     2069d  2072c  2075d  2078c  2081d  2084c  2087d  2090c  2093d  2096c  2099d  2102c  2105d  2108c
                                     2111d  2114c  2117d  2120c  2123d  2126c  2139d  2142c  2145d  2148c  2151d  2154c  2157d  2160c
                                     2163d  2166c  2167c  2170d  2173c  2177d  2180c  2183d  2186c  2189d  2192c  2195d  2198c  2201d
                                     2204c  2207d  2210c  2213d  2216c  2219d  2222c  2225d  2228c  2231d  2234c  2237d  2240c  2243d
                                     2246c  2249d  2252c  2255d  2258c
  GLE_HPGL(gle_hpgl_out)             7029d  7040d  7043c  7052d  7057d  7060c  7062c  7065d  7072c  7080d  7086c  7093d  7100c  7105c
                                     7110c  7111c  7118d  7121c  7122c  7125c  7126c  7127c  7137d  7144c  7162d  7170c  7171c  7177c
                                     7178c  7179c  7219d  7227c  7228c  7238c  7239c  7306c  7309c  7310c  7311c  7313c  7314c  7315c
                                     7316c  7317c  7318c  7319c  7320c  7321c  7332d  7335c  7340c  7342c  7343c  7344c  7345c  7348c
                                     7358d  7361c  7366c  7368c  7369c  7370c  7371c  7374c  7384d  7387c  7390c  7401d  7404c  7405c
                                     7408c  7409c  7410c  7411c  7430d  7433c  7434c  7437c  7442c  7448c  7452c  7454c  7455c  7456c
                                     7458c  7468d  7471c  7472c  7475c  7476c  7477c  7487d  7490c  7491c  7494c  7495c  7496c  7497c
                                     7498c  7500c  7510d  7517c  7571c  7572c  7579c  7580c  7614c  7615c  7678c  7684c
  GLE_RGL(gle_ras_out)               8008d  8009d  8146d  8181c  8208c  8215c  8223d  8228c  8231c  8244d  8258c  8275c  8280c  8290d
                                     8296c  8301d  8304c  8309d  8326c  8344c  8374d  8381c  8394c  8479c  8511c  8517c
  GLE_SCLIP(gle_sclip)               6008d  6012d  6015c
  GLE_SMARK(gle_smark)               5008d  5009d  5010d  5016d  5112c  5120c  5131c  5134c  5138c  5141c  5147c  5150c  5163c  5165c
                                     5171c  5174c  5178d  5181c  5185d  5188c
  GLE_STEXT(gle_stext)               4010d  4011d  4012d  4013d  4014d  4018d  4030c  4058d  4061c  4070d  4073c  4082d  4085c  4094d
                                     4097c
  GLE_TYPES(gle_types)               1029d  1030d  1031d  1032d  1033d  1034d  1035d  1036d  1037d  1038d  1039d  1040d  1041d  1042d
                                     1043d  1044d  1045d  1046d  1047d  1048d  1049d  1050d  1051d  1052d  1053d  1054d  1055d  1056d
                                     1057d  1058d  1059d  1060d  1062d  1172d  1180d  1181d  1204d  1205d  1206d  1207d  1208d  1209d
                                     1210d  1212d
  LIB(dgl_lib)                      20183c 20256c 20274c 20297c 20332c 20358c 20391c 20405c 20416c 20441c 20455c 20477c 20506c 20550c
                                    20717c 20764c 20782c 20801c 20828c 20862c 20898c 20943c 20971c 20996c 21000c 21032c 21049c 21076c
                                    21119c 21229c 21276c 21314c 21330c 21340c 21358c 21387c 21405c
gcb_space
  DGL_VARS(dgl_vars)         ***     1338d
  LIB(dgl_lib)               ***    21387c
gcbi
  DGL_C_IN(dgl_confg_in)            12007d 12039d 12047c 12056c 12084d 12093c 12115c 12131d 12134c 12135c
  GLE_GENI(gle_geni)                 3008d  3009d  3010d  3011d  3012d  3013d  3014d  3015d  3016d  3020d  3023c  3036d  3039c  3042d
                                     3045c  3049d  3052c  3055d  3058c  3061d  3064c  3067d  3070c  3073d  3076c  3079d  3082c
  GLE_HPGLI(gle_hpgl_in)            18021d 18028d 18031c 18038d 18045c 18053d 18059c 18066d 18069c 18071c 18072c 18076d 18083c 18093d
                                    18101c 18102c 18104c 18105c 18131d 18138c 18139c 18144c 18145c 18201c 18204c 18205c 18207c 18208c
                                    18209c 18210c 18211c 18212c 18213c 18214c 18215c 18220d 18248c 18249c 18251c 18252c 18261c 18262c
                                    18278d 18281c 18283c 18284c 18288d 18294c 18295c 18297c 18298c 18307c 18308c 18312d 18315c 18319c
                                    18320c 18325d 18332c 18344c 18345c 18352c 18353c 18384c
  GLE_KNOB(gle_knob_in)             18024d 18075d 18080d 18083c 18106d 18119c 18183d 18186c 18194d 18197c 18207d 18210c 18213d 18216c
                                    18263c
gchar_list
  DGL_INQ(dgl_inq)                   6012d  6088d
  TYPES dgl_types)           ***     1014d
```

```
gcp
  CTABLE(scanstuff)                  1065d  1138c
gdata
  MOUSE(mouse)                        64d    71c    83c    84c    89c    90c    91c    96c   101c   104c   107c   161c   164c   168c
                                     171c
gdotrow
  CRT(crt)                            224d   225d
general
  INITLOAD(loader)           ***      858d
  LIBRARIAN                           182c  1567c  1832c  2380c  2389c  2791c
general_0
  DC_DRV(init_dc)            ***      596d
  DC_DRV(intdc)              ***      260d
  DI_DRV(init_discint)       ***      202d
  DMA_DRV(init_dma)          ***      178d
  GLE_HPIB(gle_hpib_io)      ***    10005d
  G_DRV(init_gpio)           ***      220d
  H_DRV(init_hpib)           ***      218d
  IOLIB(general_1)           ***      252d
  IOLIB(hpib_1)              ***      420d
  IOLIB(hpib_3)              ***     2054d
  IOLIB(serial_0)            ***     2227d
  IOLIB(serial_3)            ***     2525d
  KERNEL                     ***     1323d
  KERNEL(general_0)          ***      726d
  RS_DRV(init_rs)            ***      214d
general_1
  IOLIB(general_1)           ***      220d
  IOLIB(general_2)           ***      828d
  IOLIB(hpib_3)              ***     2055d
general_2
  IOLIB(general_2)           ***      783d
general_3
  IOLIB(general_3)           ***     1092d
general_4
  IOLIB(general_4)           ***     1181d
generaltrans
  KEYS(keys)                          185d   260c
generalvalue
  INITLOAD(loader)                    871d   920d  1458c
  LIBRARIAN                           170c   172c  1178c  1543c  1544c  1592c  1598c  1638c  1639c  1922c  2186c  2318c  2556c  2557c
                                     2647c  2652c  2759c  2778c  2781c  2806c
german_kbd
  A804XDVR(a804xdvr)                  188d   193d
  NONUSKBD1(non_us_kbd1)              176c   181c
  NONUSKBD2                           179c   184c
  SYSDEVS(sysdevs)           ***      149d
getchar
  MOREFSYS(mfs)                        86d   116c   142c   168c   200c   234c   272c   311c   344c
get_amigo_letter_proc
  CTABLE(scanstuff)                  1076d  1124c  1161c
get_auto_delay
  SYSDEVS(sysdevs)           ***      141d
get_auto_repeat
  SYSDEVS(sysdevs)           ***      141d
get_bootdevice_msus
  CTABLE                     ***     1484c
  CTABLE(brstuff)                      937d   985d
get_color_map
  GLE_GEN(gle_gen)           ***     2252c
  GLE_HPGL(gle_hpgl_out)     ***     7557c
  GLE_TYPES(gle_types)       ***     1040d
get_cs80_letter_proc
  CTABLE(scanstuff)                  1077d  1123c  1162c
```

```
get_cs80_parms
  CTABLE                             1380c  1568c
  CTABLE(scanstuff)                  1041d  1133d
get_cs80_parms_proc
  CTABLE(scanstuff)                  1078d  1137c  1138c  1163c
get_cs80_parms_type
  CTABLE(scanstuff)                  1054d  1065d
get_digitize
  GLE_GENI(gle_geni)         ***     3082c
  GLE_HPGLI(gle_hpgl_in)     ***    18372c
  GLE_KNOB(gle_knob_in)      ***    18223c
  GLE_TYPES(gle_types)       ***     1204d
get_letter
  AMIGO(amigodvr)                     428d   439d
  CS80(cs80dvr)                       982d  1027d
get_letter_proc
  CTABLE(scanstuff)                  1099d  1103c  1105c
get_letter_type
  CTABLE(scanstuff)                  1053d  1064d
get_map
  AMIGO(csamigo)                      140d   189d   223c   344c   368c
  MINIT(hminit)                       288c   485c
get_message
  IOLIB(hpib_2)              ***     1991c
  KERNEL(iodeclarations)     ***      303d
get_parms
  CS80(cs80dvr)                       984d  1017d
get_polygon_info
  GLE_GEN(gle_gen)           ***     2142c
  GLE_HPGL(gle_hpgl_out)     ***     7542c
  GLE_RGL(gle_ras_out)       ***     8413c
  GLE_TYPES(gle_types)       ***     1042d
get_raster
  GLE_GEN(gle_gen)           ***     2246c
  GLE_HPGL(gle_hpgl_out)     ***     7556c
  GLE_RGL(gle_ras_out)       ***     8425c
  GLE_TYPES(gle_types)       ***     1041d
get_rpg_rate
  SYSDEVS(sysdevs)           ***      178d
getbounds
  LIBRARIAN                          1421d  1462c  1463c
getbuffer
  UCSD_AM(ucsd_am)                    158d   233c   244c
getbytes
  INIT(ldr)                          2375c  2377c
  INITLOAD(loader)                   1084d  1111d  1235c  1259c  1262c  1286c  1320c  1322c  1328c  1433c  1456c  1458c  1464c  1515c
                                     1542c  1602c  1604c
getc
  M68KSYS(ci)                         411d   444c   449c   477c   481c   488c
getchar
  M68KSYS(ci)                          70d   571d   576c   713c   902c  1100c
getcodeblocks
  LIBRARIAN                          1269d  1416c
getcommand
  LIBRARIAN                          3615d  3738c
getcommandchar
  LIBRARIAN                            98d  1452c  3356c  3407c  3621c
getcursor
  CRT(crt)                   ***      413c
  GCRT(crtb)                 ***      283c
  INITLOAD(sysglobals)       ***       67d
getdeftable
  INITLOAD(loader)                   1246d  1333c
```

```
getexitable
  INITLOAD(loader)                    1281d  1345c
getinfo
  EDRIVER(edriver)                     122d   296c
getinstrbytes
  LIBRARIAN                            478d   491c   528c   687c   780c   945c   974c
getinstruction
  LIBRARIAN                            488d  1195c  1211c  1239c
getioerrmsg
  ETU                          ***     104c
  INIT(ldr)                    ***    2414c
  INIT(misc)                           209d   300d   389c
  LIBRARIAN                    ***    3663c
  M68KSYS(ci)                  ***     682c
  MIUI                                 26c   144c
  TAPEBKUP                             990c  1074c
  TAPEBKUP(cs80tbdvr)          ***     746c
getpacc
  SRMDAM(srmdammodule)                 664d   727c   770c   778c   787c
getrealnumber
  MOREFSYS(mfs)                        396d   528c   548c
getrecsize
  ASCII(asciimodule)                   164d   195c   285c
getsdate
  LIFDAM(lifmodule)                    324d  1149c
getsize
  HEAPT(hpm)                            99d   106c   165c   171c   244c
getspace
  LIFDAM(lifmodule)                    676d   931c
gettirfo
  A804XDVR(a804xdvr)           ***     314c
  SYSDEVS(sysdevs)                     214d   480c
getuntildelim
  SRMDAM(srmdammodule)                 217d   250c   265c   275c
getvalue
  M68KSYS(ci)                          745d   768c   797c   800c   803c   846c
getvolumedate
  INITLOAD(sysglobals)         ***     133d
  LIFDAM(lifmodule)            ***    1149c
  M68KSYS(ci)                  ***     160c
  SRMDAM(srmdammodule)         ***    1791c
  VCSD_DAM(ucsdmodule)         ***     654c
getvolumename
  INIT(misc)                   ***     446c
  INITLOAD(bootdammodule)      ***     692c
  INITLOAD(sysglobals)         ***     133d
  LIFDAM(lifmodule)            ***    1134c
  M68KSYS(ci)                  ***     615c
  MIUI                                 249c   529c
  SRMDAM(srmdammodule)         ***    1789c
  TAPEBKUP                     ***    1136c
  VCSD_DAM(ucsdmodule)         ***     659c
getxy
  CRT(crt)                             293d   413c
  GCRT(crtb)                           202d   283c
```

```
gfiles
  CRT(crt)                             198c   256c   259c
  C_HOOK                               109c   113c
  DGL_RAS(dgl_raster)                17339c 17343c 17393c 17417c 17421c
  GCRT(crtb)                           161c   185c   189c
  INIT                                2523c  2524c  2525c  2526c  2545c
  INIT(ldr)                           2290c  2291c  2304c  2305c  2306c  2308c  2309c
  INITLOAD(sysglobals)         ***     257d
  M68KSYS(ci)                          117c   119c   547c   548c   567c   568c
  SWVOL                                 22c    24c
gheight
  CRT(crt)                             218d   225d   248c
gheightb
  CRT(crt)                             218d   226d   242c
gint_list
  DGL_HPGL(dgl_hpgl)                 17019d 17064d
  DGL_INQ(dgl_inq)                    6013d  6089d
  DGL_POLY(dgl_poly)                 20045d 20193d 20317d 20844d
  DGL_RAS(dgl_raster)                17350d 17429d
  DGL_VARS(dgl_vars)                  1124d  1129d
  LIB(dgl_lib)                       20059d 20066d 20753d 20771d
  TYPES(dgl_types)             ***     1012d
gl
  CTABLE(scanstuff)                    1064d  1105c
gle
  GLE_UTLS(gle_utls)           ***    21008d
gle_aras_out
  GLE_RGL(gle_ras_out)         ***     8004d
gle_asclip
  GLE_HPGL(gle_hpgl_out)       ***     7035d
  GLE_RGL(gle_ras_out)         ***     8089d
gle_astext
  GLE_HPGL(gle_hpgl_out)       ***     7034d
  GLE_RGL(gle_ras_out)         ***     8087d
gle_autl
  DGL_RAS(dgl_raster)          ***    17017d
  GLE_RGL(gle_ras_out)         ***     8091d
  GLE_SMARK(gle_smark)         ***     5014d
gle_await_blanking
  DGL_RAS(dgl_raster)          ***    17459c
  GLE_GEN(gle_gen)                     2041d  2255d
  LIB(dgl_lib)                 ***    20230c
gle_buffer_mode
  GLE_GEN(gle_gen)                     2033d  2219d
  LIB(dgl_lib)                 ***    20395c
gle_byte
  GLE_TYPES(gle_types)         ***     1006d
gle_char6
  GLE_TYPES(gle_types)                 1012d  1078d  1080d  1227d  1229d
gle_char_size
  GLE_GEN(gle_gen)                     2011d  2075d
  LIB(dgl_lib)                 ***    20836c
gle_clear
  DGL_HPGL(dgl_hpgl)           ***    17295c
  DGL_RAS(dgl_raster)                17528c 17614c
  GLE_GEN(gle_gen)                     2017d  2093d
  LIB(dgl_lib)                 ***    20551c
gle_clip_limits
  GEN(dgl_gen)                 ***     3545c
  GLE_GEN(gle_gen)                     2018d  2117d
gle_copy_to_string
  GLE_FILE(gle_file_io)        ***     9057c
  GLE_UTLS(gle_utls)                 21010d 21106d
```

```
gle_cursor
  DGL_HPGLI(dgl_hpgli)              18075c 18093c 18109c
  DGL_KNOB(dgl_knob)                18125c 18155c
  GLE_GEN(gle_gen)                   2026d  2177d
gle_define_color_map
  DGL_RAS(dgl_raster)               17219c 17282c
  GLE_GEN(gle_gen)                   2029d  2195d
gle_define_drawing_mode
  DGL_RAS(dgl_raster)          ***  17507c
  GLE_GEN(gle_gen)                   2030d  2201d
  LIB(dgl_lib)                      20993c 21012c
gle_draw
  DGL_POLY(dgl_poly)           ***  20237c
  GLE_GEN(gle_gen)                   2009d  2063d
gle_file_io
  DGL_C_OUT(dgl_confg_out)     ***  11013d
  GLE_FILE(gle_file_io)        ***   9002d
gle_fill_index_color
  DGL_POLY(dgl_poly)                20082c 20089c 20096c
  GLE_GEN(gle_gen)                   2028d  2189d
gle_flush_buffer
  DGL_HPGL(dgl_hpgl)           ***  17048c
  GLE_GEN(gle_gen)                   2024d  2163d
  LIB(dgl_lib)                      20383c 21050c
gle_gcb
  DGL_C_OUT(dgl_confg_out)          11054c 11377c
  DGL_HPGL(dgl_hpgl)                17032c 17047c 17048c 17050c 17099c 17107c 17115c 17120c 17179c 17194c 17201c 17208c 17209c 17268c
                                    17291c 17295c
  DGL_HPGLI(dgl_hpgli)              18070c 18075c 18088c 18093c 18106c 18109c
  DGL_INQ(dgl_inq)                   6181c  6316c
  DGL_KNOB(dgl_knob)           ***  18091c
  DGL_POLY(dgl_poly)                20062c 20082c 20089c 20096c 20114c 20208c 20229c 20237c 20238c 20601c 20824c 20825c 20892c 20895c
                                    20987c 21043c 21048c 21053c 21088c 21093c 21098c
  DGL_RAS(dgl_raster)               17055c 17061c 17108c 17175c 17193c 17207c 17219c 17258c 17282c 17329c 17338c 17441c 17459c 17482c
                                    17507c 17528c 17586c 17587c 17590c 17591c 17592c 17610c 17614c 17647c 17648c
  DGL_VARS(dgl_vars)           ***   1333d
  GEN(dgl_gen)                       3536c  3545c
  LIB(dgl_lib)                      20126c 20130c 20137c 20141c 20216d 20224c 20230c 20242c 20256c 20383c 20394c 20395c 20416c 20438c
                                    20443c 20455c 20509c 20547c 20551c 20574c 20578c 20604c 20608c 20631c 20661c 20723c 20728c 20742c
                                    20745c 20746c 20806c 20811c 20832c 20836c 20840c 20971c 20985c 20988c 20993c 21000c 21006c 21009c
                                    21012c 21050c 21051s 21053c 21121c 21128c 21179c 21185c 21188c 21210c 21221c 21261c 21270c 21283c
                                    21284c 21389c 21390c 21401c 21402c

gle_gcb_space
  DGL_VARS(dgl_vars)           ***   1339d
  LIB(dgl_lib)                 ***  21389c
gle_gcbi
  DGL_C_IN(dgl_confg_in)            12032c 12074c
  DGL_HPGLI(dgl_hpgli)              18026c 18028c 18032c 18059c 18064c 18079c 18098c 18115c
  DGL_INQ(dgl_inq)                   6234c  6247c  6539c  6546c
  DGL_KNOB(dgl_knob)                18032c 18033c 18060c 18104c 18110c 18111c 18130c 18161c
  DGL_VARS(dgl_vars)           ***   1334d
  LIB(dgl_lib)                      20274c 20903c 21086c 21314c 21322c 21330c 21338c 21340c 21392c 21393c
gle_gcbi_space
  DGL_VARS(dgl_vars)           ***   1340d
  LIB(dgl_lib)                 ***  21392c
gle_gen
  DGL_HPGL(dgl_hpgl)           ***  17012d
  DGL_HPGLI(dgl_hpgli)         ***  18013d
  DGL_KNOB(dgl_knob)           ***  18013d
  DGL_POLY(dgl_poly)           ***  20037d
  DGL_RAS(dgl_raster)          ***  17016d
  GEN(dgl_gen)                 ***   3055d
  GLE_GEN(gle_gen)             ***   2002d
  LIB(dgl_lib)                 ***  20105d
```

```
gle_geni
  DGL_HPGLI(dgl_hpgli)         ***  18014d
  DGL_KNOB(dgl_knob)           ***  18014d
  GLE_GENI(gle_geni)           ***   3002d
  LIB(dgl_lib)                 ***  20106d
gle_get_color_map
  GLE_GEN(gle_gen)                   2038d  2249d
gle_get_digitize
  DGL_HPGLI(dgl_hpgli)         ***  18098c
  DGL_KNOB(dgl_knob)           ***  18130c
  GLE_GENI(gle_geni)                 3016d  3079d
gle_get_input_p1p2
  GLE_GENI(gle_geni)                 3009d  3036d
  LIB(dgl_lib)                 ***  21338c
gle_get_p1p2
  GLE_GEN(gle_gen)                   2020d  2145d
  LIB(dgl_lib)                      21051s 21128c
gle_get_polygon_info
  DGL_POLY(dgl_poly)                21048c 21093c
  GLE_GEN(gle_gen)                   2021d  2139d
gle_get_raster
  DGL_RAS(dgl_raster)          ***  17338c
  GLE_GEN(gle_gen)                   2037d  2243d
gle_gload
  GLE_GEN(gle_gen)                   2035d  2231d
gle_graphics_on_off
  DGL_RAS(dgl_raster)          ***  17482c
  GLE_GEN(gle_gen)                   2034d  2225d
  LIB(dgl_lib)                 ***  21188c
gle_gstore
  GLE_GEN(gle_gen)                   2036d  2237d
gle_hpgl_in
  DGL_C_IN(dgl_confg_in)       ***  12011d
  GLE_HPGLI(gle_hpgl_in)       ***  18002d
gle_hpgl_out
  DGL_C_OUT(dgl_confg_out)     ***  11011d
  GLE_HPGL(gle_hpgl_out)       ***   7002d
gle_hpib_io
  DGL_C_IN(dgl_confg_in)       ***  12012d
  DGL_C_OUT(dgl_confg_out)     ***  11014d
  GLE_HPIB(gle_hpib_io)        ***  10002d
gle_index_color
  DGL_HPGL(dgl_hpgl)           ***  17209c
  DGL_RAS(dgl_raster)               17193c 17648c
  GLE_GEN(gle_gen)                   2016d  2111d
gle_init_gcb
  GLE_GEN(gle_gen)                   2019d  2123d
  LIB(dgl_lib)                      21390c 21397c
gle_init_hpgl_input
  DGL_C_IN(dgl_confg_in)       ***  12115c
  GLE_HPGLI(gle_hpgl_in)            18021d 18325d
gle_init_hpgl_output
  DGL_C_OUT(dgl_confg_out)          11444c 11486c
  GLE_HPGL(gle_hpgl_out)             7029d  7510d
gle_init_input_gcb
  GLE_GENI(gle_geni)                 3008d  3020d
  LIB(dgl_lib)                 ***  21393c
gle_init_knob_input
  DGL_C_IN(dgl_confg_in)       ***  12056c
  GLE_KNOB(gle_knob_in)             18024d 18213d
gle_init_raster_output
  DGL_C_OUT(dgl_confg_out)     ***  11351c
  GLE_RGL(gle_ras_out)               8008d  8374d
```

```
gle_input_echo
  DGL_HPGLI(dgl_hpgli)              18032c 18115c
  DGL_KNOB(dgl_knob)                18033c 18161c
  GLE_GENI(gle_geni)                 3011d  3049d
gle_input_escape1
  GLE_GENI(gle_geni)                 3012d  3055d
gle_input_escape2
  GLE_GENI(gle_geni)                 3013d  3061d
gle_input_term
  GLE_GENI(gle_geni)                 3010d  3042d
  LIB(dgl_lib)                  ***  21086c
gle_ishift
  GLE_SMARK(gle_smark)               5158c  5159c
gle_knob_echo_gcb
  DGL_KNOB(dgl_knob)                18068c 18072c 18077c
  DGL_VARS(dgl_vars)            ***   1335s
  LIB(dgl_lib)                      21395c 21397c
gle_knob_echo_gcb_space
  DGL_VARS(dgl_vars)            ***   1341d
  LIB(dgl_lib)                  ***  21395c
gle_knob_in
  DGL_C_IN(dgl_config_in)       ***  12013d
  GLE_KNOB(gle_knob_in)         ***  18002d
gle_linestyle
  DGL_HPGL(dgl_hpgl)            ***  17201c
  DGL_RAS(dgl_raster)           ***  17061c
  GLE_GEN(gle_gen)                   2031d  2207d
  LIB(dgl_lib)                      20985c 21006c
gle_linewidth
  GLE_GEN(gle_gen)                   2032d  2213d
  LIB(dgl_lib)                      20443c 20988c 21009c
gle_marker
  GLE_GEN(gle_gen)                   2015d  2105d
  LIB(dgl_lib)                  ***  20746c
gle_marker_size
  GLE_GEN(gle_gen)                   2023d  2157d
  LIB(dgl_lib)                  ***  21185c
gle_match
  DGL_HPGL(dgl_hpgl)                17035c 17036c 17037c 17038c 17039c 17040c 17123c 17137c 17138c 17146c 17147c 17148c 17149c 17150c
                                    17151c 17157c 17158c
  GLE_HPGL(gle_hpgl_out)             7232c  7233c  7234c  7253c  7254c  7271c  7272c  7280c  7281c  7289c  7297c  7616c  7619c  7620c
                                     7627c  7634c  7635c  7636c  7643c  7644c  7645c  7646c  7647c  7648c  7649c  7650c  7651c  7652c
                                     7660c  7661c  7662c  7663c  7666c  7667c
  GLE_HPGLI(gle_hpgl_in)            18140c 18141c 18142c 18158c 18159c 18167c 18168c 18176c 18184c 18192c
  GLE_UTLS(gle_utls)                21011d 21118d 21131c 21133c
gle_move
  DGL_POLY(dgl_poly)                20229c 20238c
  GLE_GEN(gle_gen)                   2008d  2057d
  LIB(dgl_lib)                  ***  20242c
gle_output_escape1
  DGL_HPGL(dgl_hpgl)            ***  17050c
  GLE_GEN(gle_gen)                   2039d  2051d
gle_output_escape2
  DGL_HPGL(dgl_hpgl)                17047c 17179c
  GLE_GEN(gle_gen)                   2040d  2045d
gle_polygon
  DGL_POLY(dgl_poly)                21053c 21098c
  GLE_GEN(gle_gen)                   2027d  2183d
gle_ras_out
  DGL_C_OUT(dgl_config_out)     ***  11012d
  DGL_INQ(dgl_inq)              ***   6030d
  DGL_RAS(dgl_raster)           ***  17018d
  GLE_RGL(gle_ras_out)          ***   8002d
```

```
gle_read_integer
  DGL_C_IN(dgl_config_in)           12049c 12097c
  DGL_C_OUT(dgl_config_out)         11279c 11419c
  GLE_HPGL(gle_hpgl_out)             7182c  7184c  7186c  7188c  7242c  7244c  7246c  7248c
  GLE_HPGLI(gle_hpgl_in)            18108c 18110c 18112c 18114c 18148c 18150c 18152c 18154c 18254c 18255c 18265c 18301c 18303c 18305c
  GLE_HPIB(gle_hpib_io)         ***  10493c
  GLE_UTLS(gle_utls)                21008d 21069d 21103c
gle_sample
  DGL_HPGLI(dgl_hpgli)              18028c 18079c
  DGL_KNOB(dgl_knob)            ***  18110c
  GLE_GENI(gle_geni)                 3014d  3067d
gle_sclip
  GLE_HPGL(gle_hpgl_out)        ***   7036d
  GLE_RGL(gle_ras_out)          ***   8088d
  GLE_SCLIP(gle_sclip)          ***   6002d
gle_set_marker
  GLE_GEN(gle_gen)                   2022d  2151d
  LIB(dgl_lib)                  ***  20745c
gle_sn
  GLE_UTLS(gle_utls)            ***  21069d
gle_shortint
  DGL_C_IN(dgl_config_in)           12043d 12089d
  DGL_C_OUT(dgl_config_out)         11158d 11406d
  DGL_POLY(dgl_poly)                20552d 20553d 20554d
  DGL_RAS(dgl_raster)               17203d 17250d
  GLE_FILE(gle_file_io)              9012d  9014d
  GLE_HPGL(gle_hpgl_out)             7066d  7069d  7081d  7083d  7140d  7165d  7166d  7222d  7223d  7514d
  GLE_HPGLI(gle_hpgl_in)            18039d 18042d 18054d 18056d 18079d 18096d 18097d 18134d 18135d 18226d 18227d 18230d 18291d
  GLE_HPIB(gle_hpib_io)             10013d 10486d
  GLE_KNOB(gle_knob_in)             18011d 18016d 18017d
  GLE_RGL(gle_ras_out)               8016d  8017d  8018d  8030d  8031d  8033d  8038d  8039d  8040d  8041d  8042d  8043d  8044d  8045d
                                     8047d  8048d  8049d  8050d  8051d  8052d  8053d  8054d  8060d  8061d  8065d  8066d  8094d  8095d
                                     8247d  8293d  8312d  8314d  8320d  8377d
  GLE_SMARK(gle_smark)               5096d  5097d  5098d  5099d  5100d  5101d
  GLE_STEXT(gle_stext)          ***  4008d
  GLE_TYPES(gle_types)               1007d  1074d  1075d  1076d  1079d  1081d  1091d  1092d  1093d  1094d  1095d  1096d  1097d  1098d
                                     1099d  1100d  1103d  1104d  1105d  1106s  1115d  1119d  1135d  1139d  1141d  1142d  1143d  1144d
                                     1146d  1147d  1148d  1149d  1151d  1152d  1153d  1154d  1166d  1167d  1168d  1169d  1171d  1187d
                                     1224d  1225d  1228d  1230d  1249d  1268d  1269d  1270s  1274d  1275d  1276d
  GLE_UTLS(gle_utls)                21008d 21009d 21010d 21011d 21012d 21013d 21014d 21019d 21028d 21037d 21038d 21045d 21046d 21047d
                                    21069d 21074d 21076d 21077d 21078d 21106d 21110d 21118d 21119d 21123d
gle_shortint_max
  GLE_KNOB(gle_knob_in)             18174c 18175c
  GLE_UTLS(gle_utls)                21014d 21028d 21033c 21034c
gle_shortint_min
  GLE_KNOB(gle_knob_in)             18174c 18175c
  GLE_UTLS(gle_utls)                21013d 21019d 21024c 21025c
gle_smark
  GLE_HPGL(gle_hpgl_out)        ***   7037d
  GLE_RGL(gle_ras_out)          ***   8090d
  GLE_SMARK(gle_smark)          ***   5002d
gle_soft_char_size
  GLE_HPGL(gle_hpgl_out)        ***   7531c
  GLE_RGL(gle_ras_out)          ***   8401c
  GLE_STEXT(gle_stext)               4010d  4058d
gle_soft_clip_draw
  GLE_HPGL(gle_hpgl_out)        ***   7528c
gle_soft_clip_limits
  GLE_HPGL(gle_hpgl_out)             7532c  7684c
  GLE_RGL(gle_ras_out)               8400c  8511c
  GLE_SCLIP(gle_sclip)               6008d  6012d
gle_soft_clip_move
  GLE_HPGL(gle_hpgl_out)        ***   7527c
```

```
gle_soft_marker
  GLE_HPGL(gle_hpgl_out)       ***   7537c
  GLE_RGL(gle_ras_out)         ***   8405c
  GLE_SMARK(gle_smark)               5008d  5016d
gle_soft_marker_size
  GLE_HPGL(gle_hpgl_out)       ***   7538c
  GLE_RGL(gle_ras_out)         ***   8406c
  GLE_SMARK(gle_smark)               5010d  5185d
gle_soft_set_marker
  GLE_HPGL(gle_hpgl_out)       ***   7539c
  GLE_RGL(gle_ras_out)         ***   8407c
  GLE_SMARK(gle_smark)               5009d  5178d
gle_soft_text
  GLE_HPGL(gle_hpgl_out)       ***   7530c
  GLE_RGL(gle_ras_out)         ***   8399c
gle_soft_text_dir
  GLE_HPGL(gle_hpgl_out)       ***   7535c
  GLE_RGL(gle_ras_out)         ***   8403c
  GLE_STEXT(gle_stext)               4012d  4082d
gle_soft_text_just
  GLE_HPGL(gle_hpgl_out)       ***   7536c
  GLE_RGL(gle_ras_out)         ***   8404c
  GLE_STEXT(gle_stext)               4013d  4094d
gle_soft_text_spacing
  GLE_HPGL(gle_hpgl_out)       ***   7533c
  GLE_RGL(gle_ras_out)         ***   8402c
  GLE_STEXT(gle_stext)               4011d  4070d
gle_start_digitize
  DGL_HPGLI(dgl_hpgli)         ***  18064c
  DGL_KNOB(dgl_knob)           ***  18104c
  GLE_GENI(gle_geni)                 3015d  3073d
gle_stext
  GLE_HPGL(gle_hpgl_out)       ***   7033d
  GLE_RGL(gle_ras_out)         ***   8086d
  GLE_STEXT(gle_stext)         ***   4002d
gle_term
  GLE_GEN(gle_gen)                   2025d  2170d
  LIB(dgl_lib)                 ***  21053c
gle_text
  GLE_GEN(gle_gen)                   2010d  2069d
  LIB(dgl_lib)                 ***  20728c
gle_text_dir
  GLE_GEN(gle_gen)                   2013d  2087d
  LIB(dgl_lib)                 ***  20811c
gle_text_just
  GLE_GEN(gle_gen)                   2014d  2099d
gle_text_spacing
  GLE_GEN(gle_gen)                   2012d  2081d
  LIB(dgl_lib)                 ***  20840c
gle_text_xform
  GLE_HPGL(gle_hpgl_out)       ***   7545c
  GLE_RGL(gle_ras_out)         ***   8414c
  GLE_STEXT(gle_stext)               4014d  4018d
```

```
gle_types
  DGL_C_IN(dgl_confg_in)       ***  12004d
  DGL_C_OUT(dgl_confg_out)     ***  11004d
  DGL_HPGLI(dgl_hpgli)         ***  18012d
  DGL_KNOB(dgl_knob)           ***  18012d
  DGL_POLY(dgl_poly)           ***  20036d
  DGL_RAS(dgl_raster)          ***  17015d
  DGL_TOOLS(dgl_tools)         ***  20011d
  DGL_VARS(dgl_vars)           ***     52d
  GEN(dgl_gen)                 ***   3056d
  GLE_FILE(gle_file_io)        ***   9004d
  GLE_GEN(gle_ger)             ***   2004d
  GLE_GENI(gle_geni)           ***   3004d
  GLE_HPGL(gle_hpgl_out)       ***   7004d
  GLE_HPGLI(gle_hpgl_in)       ***  18004d
  GLE_HPIB(gle_hpib_io)        ***  10004d
  GLE_KNOB(gle_knob_in)        ***  18004d
  GLE_RGL(gle_ras_out)         ***   8004d
  GLE_SCLIP(gle_sclip)         ***   6004d
  GLE_SMARK(gle_smark)         ***   5004d
  GLE_STEXT(gle_stext)         ***   4004d
  GLE_TYPES(gle_types)         ***   1002d
  GLE_UTLS(gle_utls)           ***  21004d
  LIB(dgl_lib)                 ***  20104d
gle_utls
  DGL_C_IN(dgl_confg_in)       ***  12014d
  DGL_C_OUT(dgl_confg_out)     ***  11015d
  DGL_HPGL(dgl_hpgl)           ***  17013d
  GLE_FILE(gle_file_io)        ***   9025d
  GLE_HPGL(gle_hpgl_out)       ***   7038d
  GLE_HPIB(gle_hpib_io)        ***  10036d
  GLE_KNOB(gle_knob_in)        ***  18028d
  GLE_UTLS(gle_utls)           ***  21002d
gle_write_integer
  GLE_HPGL(gle_hpgl_out)       ***   7088c
  GLE_HPGLI(gle_hpgl_in)       ***  18061c
  GLE_UTLS(gle_utls)                 21009d 21037d
gload
  GLE_GEN(gle_gen)             ***   2234c
  GLE_HPGL(gle_hpgl_out)       ***   7554c
  GLE_TYPES(gle_types)         ***   1043d
global
  INITLOAD(loader)             ***    858d
  LIBRARIAN                           181c  1566c  1617c
  SEGMENTER(asm)                       96d    99c
  SEGMENTER(segmenter)         ***     38d
globalbase
  INITLOAD(loader)                    953d  1412c
  LIBRARIAN                           370c  1685c  1971c
globaldelta
  INITLOAD(loader)                   1027d  1412c
  LIBRARIAN                          1495c  1496c  1685c
globalsize
  INITLOAD(loader)                    952d  1379c  1413c
  LIBRARIAN                           371c  1686c  1970c  2617c
globase
  INITLOAD(loader)                   1025d  1412c
  LIBRARIAN                          1685c  2617c
gmaxheight
  C_HOOK                       ***     68d
  DGL_RAS(dgl_raster)          ***  17369d
  GCRT(crtb)                   ***    141d
gmem
  CRT(crt)                            227d   255c
```

```
gmemt
  CRT(crt)                              228d   254c
gmemtype
  CRT(crt)                              226d   228d
gmemtype
  CRT(crt)                              225d   227d
go
  INIT(ldr)                            2280d  2321d
  INITLOAD                             1711d  1793c
  TAIL                                   75c    88c
go_prog
  M68KSYS(ci)                           869d   930c   938c  1136c
goff
  CRT(crt)                              206d   212c
  DGL_C_OUT(dgl_confg_out)            11214d 11225c
gon
  CRT(crt)                              205d   211c
  DGL_C_OUT(dgl_confg_out)            11213d 11224c
goodio
  ETU                                   115d   153c   437c   439c   470c   631c   878c   907c   929c  1046c  1054c
  LIFOAM(lifmodule)                      82d   301c   353c   445c   448c   528c  1111c  1145c
gotcode
  M68KSYS(ci)                            46d   641c   642c  1019c  1022c  1023c
gotexposign
  MOREFSYS(mfs)                         136d   179c
gotmantissasign
  MOREFSYS(mfs)                         305d   355c
gotnum
  M68KSYS(ci)                           211d   212c   216c   221c   224c
gotoxy
  CRT(crt)                              308d   412c   450c   696c
  GCRT(crtb)                            217d   282c   320c   510c
gotsym
  M68KSYS(ci)                            46d   635c   636c  1019c
gp
  F9885(f9885dvr)                ***     37d
gpio
  F9885(f9885dvr)                ***    107d
gpio_card
  KERNEL(general_0)              ***   1052c
  KERNEL(iodeclarations)         ***    327d
gpio_drivers
  G_DRV(init_gpio)                      209d   235c   236c   260c
gpio_enable
  F9885(f9885dvr)                        72d   178c
gpio_enable_type
  F9885(f9885dvr)                        66d    73d
gpio_initialize
  G_DRV                          ***    144d
gpio_r3_type
  F9885(f9885dvr)                        66d    74d    76d   108d
gpioclear
  F9885(f9885dvr)                        90c    97c
gpiodma_proc
  F9885(f9885dvr)                       107d   170c   175c   178c
gpiodmain
  F9885(f9885dvr)                ***    170c
gpiodmaout
  F9885(f9885dvr)                ***    175c
gpiotype
  F9885(f9885dvr)                        80d   107d
gpiowordin
  F9885(f9885dvr)                       129c   196c

gpiowordout
  F9885(f9885dvr)                       124c   127c   128c   146c   151c   163c
gptr
  F9885(f9885dvr)                        80d    87c    90c    97c   121c   122c   124c   127c   128c   129c   146c   151c   163c   178c
                                        183c   191c   196c   263c
  LIBWARIAN                            1480d  1592c  1593c  1594c  1607c  1628c
graph
  C_HOOK                         ***     45d
  DGL_TOOLS(dgl_tools)           ***  20023d
graphics_base
  C_HOOK                                 58d    86c   135c
  DGL_C_OUT(dgl_confg_out)            11156d 11236c
graphics_bd
  DGL_C_OUT(dgl_confg_out)            11164d 11195c 11203c 11287c 11289c 11300c 11305c 11336c 11347c
graphics_card
  KERNEL(general_0)                     1079c  1091c
  KERNEL(iodeclarations)         ***     329d
graphics_control_block
  DGL_VARS(dgl_vars)                    1339d  1341d
  GLE_TYPES(gle_types)                  1014d  1019d
graphics_control_block1
  DGL_VARS(dgl_vars)                    1045d  1246d  1338d
graphics_control_block_ptr
  DGL_C_OUT(dgl_confg_out)            11007d 11153d 11399d 11502d
  DGL_KNOB(dgl_knob)             ***  18047d
  DGL_RAS(dgl_raster)                 17286d 17291d
  DGL_VARS(dgl_vars)                    1333d  1335s
  GLE_GEN(gle_gen)                      2008d  2009d  2010d  2011d  2012d  2013d  2014d  2015d  2016d  2017d  2018d  2019d  2020d  2021d
                                        2022d  2023d  2024d  2025d  2026d  2027d  2028d  2029d  2030d  2031d  2032d  2033d  2034d  2035d
                                        2036d  2037d  2038d  2039d  2040d  2041d  2045d  2051d  2057d  2063d  2069d  2075d  2081d  2087d
                                        2093d  2099d  2105d  2111d  2117d  2123d  2139d  2145d  2151d  2157d  2163d  2170d  2177d  2183d
                                        2189d  2195d  2201d  2207d  2213d  2219d  2225d  2231d  2237d  2243d  2249d  2255d
  GLE_HPGL(gle_hpgl_out)                7029d  7040d  7052d  7057d  7065d  7080d  7093d  7118d  7137d  7162d  7219d  7332d  7358d  7384d
                                        7401d  7430d  7468d  7487d  7510d
  GLE_RGL(gle_ras_out)                  8008d  8009d  8146d  8223d  8244d  8290d  8301d  8309d  8374d
  GLE_SCLIP(gle_sclip)                  6008d  6012d
  GLE_SMARK(gle_smark)                  5008d  5009d  5010d  5016d  5178d  5185d
  GLE_STEXT(gle_stext)                  4010d  4011d  4012d  4013d  4014d  4018d  4058d  4070d  4082d  4094d
  GLE_TYPES(gle_types)                  1014d  1029d  1030d  1031d  1032d  1033d  1034d  1035d  1036d  1037d  1038d  1039d  1040d  1041d
                                        1042d  1043d  1044d  1045d  1046d  1047d  1048d  1049d  1050d  1051d  1052d  1053d  1054d  1055d
                                        1056d  1057d  1058d  1059d  1060d  1062d  1172d  1180d  1181d
  LIB(dgl_lib)                   ***  20216d
graphics_error
  DGL_VARS(dgl_vars)             ***   1244d
  GEN(dgl_gen)                   ***   3069c
  LIB(dgl_lib)                        20120c 21451c
graphics_error_number
  DGL_VARS(dgl_vars)             ***   1209d
  GEN(dgl_gen)                   ***   3070c
graphics_init
  LIB(dgl_lib)                        20017d 21372d
graphics_input_control_block
  DGL_VARS(dgl_vars)             ***   1340d
  GLE_TYPES(gle_types)                  1015d  1195d
graphics_input_control_block_ptr
  DGL_C_IN(dgl_confg_in)              12007d 12039d 12084d 12131d
  DGL_VARS(dgl_vars)             ***   1334d
  GLE_GENI(gle_geni)                    3008d  3009d  3010d  3011d  3012d  3013d  3014d  3015d  3016d  3020d  3036d  3042d  3049d  3055d
                                        3061d  3067d  3073d  3079d
  GLE_HPGLI(gle_hpgl_in)              18021d 18028d 18038d 18053d 18066d 18076d 18093d 18131d 18220d 18278d 18288d 18312d 18325d
  GLE_KNOB(gle_knob_in)               18024d 18075d 18080d 18106d 18183d 18194d 18207d 18213d
  GLE_TYPES(gle_types)                  1015d  1204d  1205d  1206d  1207d  1208d  1209d  1210d  1212d
```

```
graphics_on_off
  DGL_RAS(dgl_raster)              17291d 17587c 17591c
  GLE_GEN(gle_gen)          ***     2228c
  GLE_HPGL(gle_hpgl_out)    ***     7543c
  GLE_RGL(gle_ras_out)      ***     8416c
  GLE_TYPES(gle_types)      ***     1044d
graphics_rev
  DGL_VARS(dgl_vars)        ***     1003d
graphics_term
  LIB(dgl_lib)                     20031d 21090d 21378c
graphicsbase
  CRT(crt)                           229d   254c   255c
graphicserror
  LIB(dgl_lib)                     20041d 20115d 20120c
graphicstate
  CRT(crt)                           210c   211c
  C_HOOK                             120c   121c
  DGL_C_OUT(dgl_confg_out)         11165d 11196c 11219c 11224c 11233c
  DGL_RAS(dgl_raster)              17300c 17301c
  KEYS(keys)                         384c   390c
  SYSDEVS(sysdevs)          ***      118d
gread
  INITLOAD(bootdammodule)            581d   632c   655c   686c   710c   740c   741c   743c   750c   799c
greal_list
  DGL_HPGL(dgl_hpgl)               17020d 17065d
  DGL_INQ(dgl_inq)                  6014d  6090d
  DGL_POLY(dgl_poly)               20025d 20030d 20911d 21072d 21148d
  DGL_RAS(dgl_raster)              17247d 17351d 17430d
  DGL_VARS(dgl_vars)                1125d  1130d
  LIB(dgl_lib)                     20048d 20060d 20067d 20693d 20754d 20772d
  TYPES(dgl_types)          ***     1013d
green
  DGL_INQ(dgl_inq)                  6052c  6056c
  DGL_POLY(dgl_poly)               20079c 20087c
  DGL_RAS(dgl_raster)              17026d 17027d 17028d 17029d 17030d 17031d 17032d 17033d 17034d 17035d 17036d 17037d 17038d 17039d
                                   17040d 17041d 17111c 17181c 17187c 17191c 17224c 17235c 17240c 17274c 17629c
  DGL_VARS(dgl_vars)        ***     1025d
grn_intensity
  GLE_RGL(gle_ras_out)              8044d  8198c
growbyte
  CRT(crt)                           222d   255c
gs
  M68KSYS(ci)                        174d   213c   214c   222c   235c   236c   245c   259c   260c   270c   271c
gshortint
  DGL_HPGL(dgl_hpgl)               17246d 17247d
  DGL_INQ(dgl_inq)          ***     6164d
  DGL_RAS(dgl_raster)              17255d 17550d 17551d
  DGL_VARS(dgl_vars)                1084d  1085d  1086d  1087d  1089d  1090d  1091d  1092d  1093d  1094d  1095d  1100d  1102d  1253d
                                    1254d  1294d  1295d  1296d  1297d  1298d  1299d  1303d
  GEN(dgl_gen)                      3026d  3152d  3274d  3281d  3402d  3403d  3404d  3405d
  LIB(dgl_lib)                     20049d 20050d 20616d 20646d
  TYPES(dgl_types)                  1008d  1011d
gshortint_list
  DGL_POLY(dgl_poly)               20022d 20023d 20026d 20027d 20028d 20031d 20046d 20194d 20910d 20912d 21026d 21027d 21073d 21116d
                                   21117d 21149d
  LIB(dgl_lib)                     20046d 20677d
  TYPES(dgl_types)          ***     1011d
gsize
  HEAPT(hpm)                         136d   171c   172c
gspacing
  GLE_RGL(gle_ras_out)      ***     8039d
```

```
gstore
  GLE_GEN(gle_gen)          ***     2240c
  GLE_HPGL(gle_hpgl_out)    ***     7555c
  GLE_TYPES(gle_types)      ***     1045d
gstring255
  DGL_VARS(dgl_vars)        ***     1110s
  LIB(dgl_lib)                     20021d 20022d 20072d 20709d 21193d 21194d
  TYPES(dgl_types)          ***     1009d
gsxmax
  GEN(dgl_gen)                      3403d  3425c  3442c
gsxmin
  GEN(dgl_gen)                      3402d  3424c  3442c  3445c
gsymax
  GEN(dgl_gen)                      3405d  3427c  3446c
gsymin
  GEN(dgl_gen)                      3404d  3426c  3446c  3449c
gtext
  LIB(dgl_lib)                     20072d 20709d
gtl_message
  IOLIB(hpib_2)                     1867c  1880c
  KERNEL(iodeclarations)    ***      300d
gunitbusy
  GLE_KNOB(gle_knob_in)            18059d 18063c 18068c 18123c
gunitread
  GLE_KNOB(gle_knob_in)            18045d 18132c
gvalue
  INITLOAD(loader)          ***     1104d
gvaluestring
  LIBRARIAN                          41d   158c   159c   163c   181c   191c   195c   196c   209c   213c   216c   361c   540c  1064c
                                    1306c  1335c  2221c  2500c  2501c  2564c  2565c  2575c  2576c
gvp
  CTABLE(ctr)               ***      403c
  INITLOAD(loader)                  898d  1459c  1526c
  LIBRARIAN                         143d   166c   204c   360c   365c   408c   409c   411c   524c   535c  1280c  1293c  1294c  1334c
                                   1541c  1593c  1594c  1607c  1636c  1811c  1812c  1923c  1941c  2102c  2182c  2184c  2217c  2313c
                                   2374c  2440c  2451c  2475c  2486c  2488c  2498c  2531c  2545c  2551c  2621c  2643c  2657c  2715c
                                   2754c  2774c  2802c
  SEGMENTER(asm)            ***      135c
gvptr
  INITLOAD(loader)          ***     1104d
  LIBRARIAN                        1633d  1636c  1638c  1639c
gvrbase
  LIBRARIAN                        1165d  1174c  1175c  1178c
gvrequal
  LIBRARIAN                        1805d  1810c  1841c  1843c  1877c  2434c
gvrp
  LIBRARIAN                        2130d  2498c  2499c  2551c  2552c
gvrptr
  INITLOAD(loader)                  871d   898d  1272c
  LIBRARIAN                          87d   143d   153d   170c   172c   176c   199c   303d  1175c  1808d  2052c  2130d
gvrstring
  LIBRARIAN                         143d   360c   524c   535c   539c  1063c  1294c  1334c  2217c  2499c  2552c
gwidth
  CRT(crt)                          219d   224d   249c   255c
gwidthb
  CRT(crt)                          219d   220d   226d   243c   254c
  C_HOOK                             67d    69d   109c
  DGL_RAS(dgl_raster)              17368d 17370d 17417c
  GCRT(crtb)                        140d   142d   185c
gword
  CRT(crt)                          221d   224d
```

```
h
  DGL_POLY(dgl_poly)                    20059d 20108d
  DGL_RAS(dgl_raster)           ***     17172d
  GEN(dgl_gen)                          3169d  3173c  3174c  3175c
  MIU(                                    15d   176c   333c   349c   474c
haddress
  SRM_DRV(srm)                           261d  1503c
halfword
  SRM_DRV(srm)                  ***       38d
hallocate
  HEAPT(hpm)                             210d   298c
handle_bad_status
  CS80(cs80dsr)                          604d   635d   883c   889c   898c   904c   910c   945c   951c
  CS80(cs80dvr)                         1048c  1054c  1127c
  MINIT(qminit)                          963c   969c   979c  1130c  1151c  1159c  1168c  1174c  1180c  1188c  1198c  1210c  1218c  1229c
                                        1235c  1241c  1249c  1268c
  TAPEBKUP(cs80tb_vr)                    373c   378c   402c   420c   438c   443c   671c   725c   731c   920c   938c
handle_copydata_status
  TAPEBKUP(cs80tb_vr)                    695d   902c
handle_wee_pending
  TAPEBKUP(cs80tb_vr)                    706d   874c
handle_verify_status
  TAPEBKUP(cs80tb_vr)                    515d   680c
hard_xmax
  GLE_RGL(gle_rasterout)                8041d  8505c
hard_xmax
  DGL_C_OUT(dgl_config_out)     ***     11063c
  GLE_RGL(gle_rasterout)                8042d  8233c  8476c  8506c
harddisc_msus
  CTABLE                                1204d  1335c  1443c  1525c
hardvols
  CS80(cs80dvr)                          985d  1018d  1022c
hardwarestatus
  INITLOAD(sysglobals)          ***      249d
has_color
  DGL_C_OUT(dgl_config_out)            11027d 11079d 11268c
hasclock
  CRT(crt)                      ***      116d
  GCRT(crtb)                    ***       49d
  SYSDEVS(sysdevs)              ***       70d
hascolor
  CRT(crt)                              168d   477c   692c   694c
haslcrt
  CRT(crt)                      ***      115d
  GCRT(crtb)                    ***       48d
  SYSDEVS(sysdevs)              ***       70d
hasprompts
  HPHIL(hphil)                  ***      121c
  SYSDEVS(sysdevs)              ***      317d
hasxycrt
  CRT(crt)                      ***      114d
  GCRT(crtb)                    ***       47d
  SYSDEVS(sysdevs)              ***       69d
hatch
  DGL_POLY(dgl_poly)                   20425d 20595c 20604c 20832c 20838c 20877d 20890c 20897c 20903c
haveData
  A804XDVR(a804xdvr)                     66d    82c    83c    84c   128c   148c   236c   387c
head
  AMIGO(csamigo)                         91d   351c   356c   372c   373c
  MINIT(cs80ir)                         523d   564d   565d   728d   747d   753c   818d   839c
  MINIT(qminit)                        1105d  1192c  1197c
  MINIT(xminit)                        1405d  1556c  1562c  1586c  1628c  1699c  1722c  1733c  1757c  1782c  1798c
```

```
head_vector_compare_error
  AMIGO(amigodvr)               ***      808c
  AMIGO(csamigo)                ***       73d
  MINIT(hminit)                 ***      320c
header
  EPROMS(eproms)                          25d    86c
headersize
  EPROMS(eproms)                          19d    57c
headnum
  MINIT(asmr)                           1331d  1335d  1342d  1346d
headp
  ETU                                    838d   868d
headrec
  ETU                                    836d   841d   865d
heap
  M68KSYS(ci)                            395d   432c   532c
heapbase
  HEAPT(hpm)                              41d   128c   129c   138c   196c   219c   265c   282c   289c   294c   307c
  HEAPT(sysglobals)             ***       33d
  INITLOAD(sysglobals)          ***      245d
heapiruse
  ETU                                     76d   887c  1130c  1135c  1229c  1262c  1263c
heapmark
  LIFDAM(lifmodule)                      388d   397c   470c   473c
  UCSD_DAM(ucsdmodule)                   572d   575c   624c   625c
heapmax
  HEAPT(hpm)                              40d   140c   176c   276c   281c
  HEAPT(sysglobals)             ***       32d
  INITLOAD(sysglobals)          ***      244d
heapoff
  SEGMENTER(asm)                         141d   146c
height
  CRT(crt)                                87d   133d   188c   670c   672c   673c
  GCRT(crtb)                              66d   504c
  LIB(dgl_lib)                         20071d 20816d 20831c 20835c 20839c
  SYSDEVS(sysdevs)              ***       88d
here
  LIFDAM(lifmodule)                       73d   686d   689c   691c   706c   713c   722c   723c   724c   728c   730c   731c   769c   775c
                                         791c   797c   831c   833c   844c   846c   873c   932c
hestablish
  HEAPT(hpm)                              45d   119d
hexout
  LIBRARIAN                              423d   474c   482c
hi_rom
  EDRIVER(edriver)                        82d   174c   187c
  EPROMS(eproms)                          16d   100c   102c
high
  C_HOOK                        ***       40d
  DGL_TOOLS(dgl_tools)          ***    20018d
  M68KSYS(ci)                            745d   762c   765c
highbyte
  HEAPT(hpm)                              65d    89c
highcode
  SEGMENTER(asm)                          72d    79c    80c    82c    92c    98c   294c
highcode
  SEGMENTER(asm)                          76d    79c    81c
  SEGMENTER(segmenter)          ***       24d
highglob
  SEGMENTER(asm)                          71d    85c    86c    88c    93c    99c   147c   162c   205c   206c   207c   208c
highglobal
  SEGMENTER(asm)                          76d    85c    87c
  SEGMENTER(segmenter)          ***       25d
```

```
highheap
  INIT(ldr)                          2430c  2440c  2460c  2468c  2469c  2471c  2472c
  INITLOAD(loader)                   1050d  1115c  1142c  1143c  1144c  1148c  1151c  1152c  1153c  1158c  1159c  1161c  1598c  1625c
                                     1631c  1632c  1634c  1635c
  LIBRARIAN                           128c  2513c  2838c  2840c  2841c  2843c  2849c  2961c  2967c  3093c  3095c  3096c  3722c  3723c
  SEGMENTER(asm)                      189c   190c   201c   235c   236c   237c   241c   242c   243c
highheap0
  LIBRARIAN                            61d  2967c  3723c
  SEGMENTER(asm)                      178d   190c   201c
highid
  HPHIL(hphil)                   ***  257c
  MOUSE(mouse)                   ***  187c
  SYSDEVS(sysdevs)               ***  338d
highlight
  CRT(crt)                            167d   416c   478c   479c   480c   481c   482c   485c   683c
  GCRT(crtb)                          114d   286c   348c   377c   409c   455c   457c   509c
highlightbyte
  SYSDEVS(sysdevs)               ***   48d
highlightsxorbiggraphics
  INITLOAD(sysglobals)           ***  285d
highlite
  C_HOOK                         ***   44d
  DGL_TOOLS(dgl_tools)           *** 20022d
highmark
  HEAPT(hpm)                          135d   140c   164c   174c
highrange
  LIBRARIAN                           307d  1200c  1201c  1251c  1252c  1429c  1430c  1434c
hintdata
  MINIT(hminit)                       251d   259d   266c
  MIUI                           ***  333c
hlimit
  M68KSYS(ci)                         209d   218c   224c
hminit
  MINIT(hminit)                  ***  245d
  MIUI                           ***   12d
hminitialize
  MINIT(hminit)                       254d   299d
  MIUI                           ***  349c
holdpathid
  SRMDAM(srmdammodule)               1628d  1749c  1756c  1769c  1776c
hole
  LIFDAM(lifmodule)                    74d   686d   690c   691c   693c   695c   698c   706c   708c   709c   710c   724c   732c   748c
                                      769c   776c   791c   798c   813c   814c   815c   816c   829c   831c   834c   842c   844c   847c
                                      872c
home
  CRT(crt)                       ***  125d
  GCRT(crtb)                     ***   58d
  SYSDEVS(sysdevs)               ***   78d
homechar
  CRT(crt)                       ***  460c
  ETU                            ***  112c
  GCRT(crtb)                     ***  331c
  INIT(misc)                     ***  160d
  M68KSYS(ci)                         556c   710c
  TAPEBKUP                       ***  1501c
homecursor
  M68KSYS(ci)                          66d   556c   563c
```

```
hour
  A804XDVR(a804xdvr)                  279c   307c   308c
  CLOCK(clock)                         79c    80c   120c
  INITLOAD(sysglobals)           ***  224d
  LIBRARIAN                      ***  235c
  LIFDAM(lifmodule)                   123c   555c   588c
  M68KSYS(ci)                         130c   165c   275c
  MIUI                                 78c    80c
  SRMDAM(srmdammodule)                373c   374c   388d
  UCSD_DAM(ucsdmodule)                523c   553c
hp
  MINIT(bminit)                        65d    69c    90c    92c
hp7905
  AMIGO(amigodvr)                     504d   958c
  AMIGO(csamigo)                       44d   168c   179c   185c
  CTABLE                            1399c  1546c  1558c  1689s  1695s  1702s
  CTABLE(brstuff)                    992d  1009c
  CTABLE(ctr)                         251d   550c   568c   745c   747c
hp7905_mp
  CTABLE(ctr)                         534d   550c
hp7905_pp
  CTABLE(ctr)                    ***  568c
  CTABLE(options)                ***  178d
hp7906
  AMIGO(amigodvr)                ***  504d
  AMIGO(csamigo)                       44d   169c   179c   185c
  CTABLE                            1399c  1546c  1707s  1713s  1718s
  CTABLE(brstuff)                    992d  1009c
  CTABLE(ctr)                         252d   551c   569c   745c
hp7906_mp
  CTABLE(ctr)                         535d   551c
hp7906_pp
  CTABLE(ctr)                    ***  569c
  CTABLE(options)                ***  180d
hp7920
  AMIGO(amigodvr)                ***  504d
  AMIGO(csamigo)                       44d   170c   179c
  CTABLE                            1256c  1399c  1533c  1723s  1729s
  CTABLE(brstuff)                ***  992d
  CTABLE(ctr)                         253d   552c   570c   727c
hp7920_mp
  CTABLE(ctr)                         536d   552c
hp7920_pp
  CTABLE(ctr)                    ***  570c
  CTABLE(options)                ***  182d
hp7925
  AMIGO(amigodvr)                     504d   958c
  AMIGO(csamigo)                       44d   171c   179c   191d
  CTABLE                            1256c  1399c  1533c  1734s  1740s
  CTABLE(brstuff)                ***  992d
  CTABLE(ctr)                         254d   553c   571c   727c
hp7925_mp
  CTABLE(ctr)                         537d   553c
hp7925_pp
  CTABLE(ctr)                    ***  571c
  CTABLE(options)                ***  184d
hp8290x
  AMIGO(amigodvr)                     520c   685c   752c
  AMIGO(csamigo)                       44d   164c   191d   320c
  CTABLE                            1256c  1396c
  CTABLE(brstuff)                ***  991d
  CTABLE(ctr)                         245d   544c   716c   795c
  MINIT(hminit)                  ***  490c
```

```
hp8280x_mp
  CTABLE(ctr)                          528d   544c
hp913x_a
  AMIGO(amigodvr)                      689c   958c
  AMIGO(csamigo)                        44d   165c
  CTABLE                               1256c  1399c  1533c  1658s  1663s
  CTABLE(brstuff)              ***     991d
  CTABLE(ctr)                          248d   547c   565c   727c
  MINIT(hminit)                ***     491c
hp913x_a_mp
  CTABLE(ctr)                          531d   547c
hp913x_a_pp
  CTABLE(ctr)                  ***     565c
  CTABLE(options)              ***     172d
hp913x_b
  AMIGO(csamigo)                        44d   166c
  CTABLE                               1256c  1399c  1533c  1668s  1673s
  CTABLE(brstuff)              ***     992d
  CTABLE(ctr)                          249d   548c   566c   727c
hp913x_b_mp
  CTABLE(ctr)                          532d   548c
hp913x_b_pp
  CTABLE(ctr)                  ***     566c
  CTABLE(options)              ***     174d
hp913x_c
  AMIGO(amigodvr)                      492d   689c   958c
  AMIGO(csamigo)                        44d   167c
  CTABLE                               1256c  1399c  1533c  1678s  1683s
  CTABLE(brstuff)              ***     992d
  CTABLE(ctr)                          250d   549c   567c   727c
  MINIT(hminit)                ***     491c
hp913x_c_mp
  CTABLE(ctr)                          533d   549c
hp913x_c_pp
  CTABLE(ctr)                  ***     567c
  CTABLE(options)              ***     176d
hp98253
  KERNEL(general_0)            ***    1084c
  KERNEL(iodeclarations)       ***     357d
hp98259
  KERNEL(general_0)            ***    1105c
  KERNEL(iodeclarations)       ***     359d
hp98620
  DMA_DRV(init_dma)            ***     236c
  KERNEL(iodeclarations)       ***     343d
hp98622
  F9885(f9885dvr)              ***     262c
  G_DRV(init_gpio)                     257c   293c
  KERNEL(general_0)            ***    1053c
  KERNEL(iodeclarations)       ***     352d
hp98623
  KERNEL(iodeclarations)       ***     353d
hp98624
  H_DRV(init_hpib)                     258c   304c
  KERNEL(general_0)            ***    1039c
  KERNEL(iodeclarations)       ***     350d
hp98625
  DISCHPIB(dischpib)                   233c   240c   280c   339c   343c   483c   510c
  DI_DRV(init_disc;nt)                 242c   281c
  KERNEL(general_0)            ***    1060c
  KERNEL(iodeclarations)       ***     354d
```

```
hp98626
  IOLIB(serial_0)                     2260c  2323c  2417c
  IOLIB(serial_3)                     2571c  2636c  2711c  2765c  2817c  2852c
  KERNEL(general_0)            ***    1045c
  KERNEL(iodeclarations)       ***     351d
  PRINTER(prtdvr)              ***     224c
  RS_DRV(init_rs)                      247c   278c
hp98627
  KERNEL(general_0)            ***    1090c
  KERNEL(iodeclarations)       ***     358d
hp98627a_address
  DGL_C_OUT(dgl_confg_out)            11024d 11323c 11324c
hp98628_async
  DC_DRV(init_dc)                      631c   679c
  DC_DRV(intdc)                        276c   391c
  IOLIB(serial_0)                     2240c  2303c  2369c
  IOLIB(serial_3)                     2560c  2607c  2703c  2745c  2810c  2845c
  KERNEL(iodeclarations)       ***     355d
hp98628_dsndl
  DC_DRV(init_dc)                      632c   680c
  DC_DRV(intdc)                ***     277c
  KERNEL(iodeclarations)       ***     340d
hp98629
  DC_DRV(init_dc)                      633c   686c
  DC_DRV(intdc)                ***     293c
  KERNEL(iodeclarations)       ***     341d
  SRMAM(srmammodule)           ***     150c
hp98644
  IOLIB(serial_0)                     2260c  2323c  2417c
  IOLIB(serial_3)                     2572c  2637c  2712c  2766c  2817c  2852c
  KERNEL(general_0)            ***    1140c
  KERNEL(iodeclarations)       ***     360d
  RS_DRV(init_rs)                      247c   278c
hp9885
  CTABLE                              1195d  1254c  1396c
  CTABLE(brstuff)              ***     991d
  CTABLE(ctr)                          246d   545c   693c   694c   797c
hp9885_default_dav
  CTABLE                       ***    1195d
  CTABLE(options)              ***     117d
hp9885_default_msus
  CTABLE                              1194d  1458c
hp9885_mp
  CTABLE(ctr)                          529d   545c
hp9895
  AMIGO(amigodvr)                      492d   659c
  AMIGO(csamigo)                        44d   163c   208c   316c
  CTABLE                              1256c  1347c  1402c  1528c
  CTABLE(brstuff)              ***     991d
  CTABLE(ctr)                          247d   546c   706c   707c   798c
  MINIT(hminit)                ***     490c
hp9895_mp
  CTABLE(ctr)                          530d   546c
hp_datacomm
  DC_DRV(init_dc)                      635c   639c   671c
  DC_DRV(intdc)                        275c   392c
  KERNEL(general_0)            ***    1130c
  KERNEL(iodeclarations)       ***     342d
hpgator
  KERNEL(general_0)            ***    1078c
  KERNEL(iodeclarations)       ***     356d
hpgl_buffer_mode
  GLE_HPGL(gle_hpgl_out)              7384d  7546c
```

```
hpgl_clear
  GLE_HPGL(gle_hpgl_out)            7468d  7529c
hpgl_color
  DGL_HPGL(dgl_hpgl)               17205d 17275c
hpgl_color_table
  DGL_HPGL(dgl_hpgl)               17212d 17276c
hpgl_cursor
  GLE_HPGL(gle_hpgl_out)            7487d  7544c
hpgl_device_rec
  DGL_C_OUT(dgl_confg_out)    ***  11033d
  GLE_HPGL(gle_hpgl_out)            7023d  7024d
hpgl_device_rec_ptr
  DGL_C_OUT(dgl_confg_out)         11373d 11405d
  GLE_HPGL(gle_hpgl_out)            7023d  7045c  7100c  7121c  7172c  7229c  7335c  7361c  7404c  7433c  7471c  7490c  7518c
hpgl_device_rec_space
  DGL_C_OUT(dgl_confg_out)         11033d 11426c
hpgl_draw
  GLE_HPGL(gle_hpgl_out)            7358d  7526c
hpgl_flush_buffer
  GLE_HPGL(gle_hpgl_out)            7040d  7062c  7179c  7239c  7310c  7321c  7390c  7500c  7559c  7572c  7580c  7615c
  GLE_HPGLI(gle_hpgl_in)           18028d 18072c 18105c 18145c 18215c 18252c 18262c 18284c 18298c 18308c 18320c 18345c 18353c
hpgl_get_digitize
  GLE_HPGLI(gle_hpgl_in)           18288d 18372c
hpgl_get_hard_clip
  GLE_HPGL(gle_hpgl_out)            7219d  7678c
hpgl_get_input_hard_clip
  GLE_HPGLI(gle_hpgl_in)           18131d 18384c
hpgl_get_input_p1p2
  GLE_HPGLI(gle_hpgl_in)           18093d 18201c 18205c 18373c
hpgl_get_p1p2
  GLE_HPGL(gle_hpgl_out)            7162d  7306c  7311c  7541c
hpgl_input_echo
  GLE_HPGLI(gle_hpgl_in)           18312d 18374c
hpgl_input_esc
  DGL_HPGL(dgl_hpgl)               17016d 17273c
hpgl_input_escapei
  GLE_HPGLI(gle_hpgl_in)           18076d 18375c
hpgl_input_escapeo
  GLE_HPGLI(gle_hpgl_in)           18066d 18376c
hpgl_linestyle
  DGL_HPGL(dgl_hpgl)               17184d 17274c
  GLE_HPGL(gle_hpgl_out)            7430d  7534c
hpgl_move
  GLE_HPGL(gle_hpgl_out)            7332d  7525c
hpgl_output_esc
  DGL_HPGL(dgl_hpgl)               17061d 17272c
hpgl_output_escapei
  GLE_HPGL(gle_hpgl_out)            7137d  7548c
hpgl_output_escapeo
  GLE_HPGL(gle_hpgl_out)            7118d  7547c
hpgl_sample
  GLE_HPGLI(gle_hpgl_in)           18220d 18370c
hpgl_set_color
  GLE_HPGL(gle_hpgl_out)            7401d  7540c
hpgl_start_digitize
  GLE_HPGLI(gle_hpgl_in)           18278d 18371c
hpgli_await_locator
  DGL_HPGLI(dgl_hpgli)             18036d 18128c
hpgli_sample_locator
  DGL_HPGLI(dgl_hpgli)             18017d 18129c
hphil
  HPHIL                       ***    386d
  HPHIL(hphil)                ***     28d
```

```
hphilcmdhook
  HPHIL(hphil)                ***    371c
  MOUSE(mouse)                ***    195c
  SYSDEVS(sysdevs)                   361d   653c
hphilcmdproc
  SYSDEVS(sysdevs)                   308d   361d
hphilop
  HPHIL(hphil)                ***    291d
  SYSDEVS(sysdevs)                   306d   308d   593d
hphydata
  MINIT(hminit)                      253d   283d   295c
  MIUI                        ***    474c
hpib
  CTABLE                      ***   1451c
  CTABLE(options)                     79d    81d
  C_HOOK                      ***     39d
  DGL_TOOLS(dgl_tools)        ***  20017d
hpib_0
  IOLIB(hpib_0)               ***   1692d
  IOLIB(hpib_2)               ***   1806d
  IOLIB(hpib_3)               ***   2056d
hpib_1
  IOLIB(general_2)            ***    827d
  IOLIB(general_4)            ***   1255d
  IOLIB(hpib_1)               ***    377d
  IOLIB(hpib_2)               ***   1807d
  IOLIB(hpib_3)               ***   2057d
hpib_2
  IOLIB(hpib_2)               ***   1761d
hpib_3
  IOLIB(hpib_3)               ***   2011d
hpib_card
  DISCHPIB(dischpib)          ***    248c
  GLE_HPIB(gle_hpib_io)              10146c 10210c 10234c 10298c 10322c
  IOLIB(hpib_1)                       470c   485c   532c   556c   620c   644c   713c   763c
  IOLIB(hpib_3)               ***   2120c
  KERNEL(general_0)                  1018c  1038c  1059c
  KERNEL(iodeclarations)      ***    326d
  PRINTER(prtdvr)                     121c   215c   235c
hpib_drivers
  H_DRV(init_hpib)                    208d   233c   234c   262c
hpib_init
  DGL_C_IN(dgl_confg_in)             12113c 12119c
  DGL_C_OUT(dgl_confg_out)           11483c 11489c
  GLE_HPIB(gle_hpib_io)              10023d 10483d
hpib_initialize
  H_DRV                       ***    144d
hpib_inq_timeout
  DGL_C_IN(dgl_confg_in)      ***  12107c
  DGL_C_OUT(dgl_confg_out)    ***  11476c
  GLE_HPIB(gle_hpib_io)              10024d 10467d
hpib_iocb
  DGL_C_IN(dgl_confg_in)      ***  12024d
  DGL_C_OUT(dgl_confg_out)    ***  11036d
  GLE_HPIB(gle_hpib_io)              10010d 10011d
hpib_iocb_ptr
  DGL_C_IN(dgl_confg_in)             12070d 12087d
  DGL_C_OUT(dgl_confg_out)           11371d 11403d
  GLE_HPIB(gle_hpib_io)              10010d 10470c 10479c 10489c 10514c 10535c
hpib_iocb_space
  DGL_C_IN(dgl_confg_in)             12024d 12101c
  DGL_C_OUT(dgl_confg_out)           11036d 11471c
```

```
hpib_line
  IOLIB(hpib_0)                    1713d  1744d  1754c
  IOLIB(hpib_3)            ***     2068c
hpib_read
  DGL_C_IN(dgl_confg_in)   ***    12105c
  DGL_C_OUT(dgl_confg_out) ***    11474c
  GLE_HPIB(gle_hpib_io)           10027d 10528d
hpib_set_timeout
  DGL_C_IN(dgl_confg_in)   ***    12108c
  DGL_C_OUT(dgl_confg_out) ***    11477c
  GLE_HPIB(gle_hpib_io)           10025d 10476d
hpib_term
  DGL_C_IN(dgl_confg_in)   ***    12076c
  DGL_C_OUT(dgl_confg_out) ***    11389c
  GLE_HPIB(gle_hpib_io)           10028d 10552d
hpib_write
  DGL_C_IN(dgl_confg_in)   ***    12104c
  DGL_C_OUT(dgl_confg_out) ***    11473c
  GLE_HPIB(gle_hpib_io)           10026d 10507d
hpibaddr
  MINIT(xminit)                    1365d  1585c  1628c  1722c  1733c  1757c  1798c
hpibamigo_clear
  AMIGO(amigodvr)                   474c   524c
  DISCHPIB(dischpib)                197d   310d
hpibamigo_identify
  AMIGO(amigodvr)                   512c   614c
  CS80(cs80dvr)                    1004c  1103c
  DISCHPIB(dischpib)                199d   383d   406c   412c
  MINIT(qminit)            ***      957c
  TAPEBKUP(cs80tbdvr)      ***      350c
hpibcheck_sc
  AMIGO(amigodvr)                   581c   613c
  CS80(cs80dvr)            ***     1003c
  DISCHPIB(dischpib)                194d   244d
hpibcheck_sc_proc
  CTABLE(scanstuff)                1073d  1119c  1158c  1168c
hpibget_amigo_ident
  DISCHPIB(dischpib)                200d   410d
hpibget_amigo_ident_proc
  CTABLE(scanstuff)                1074d  1120c  1159c  1169c
hpibget_amigo_ident_type
  CTABLE(scanstuff)                1052d  1063d
hpibrec
  GLE_HPIB(gle_hpib_io)           10192d 10248c 10249c 10251c 10280d 10336c 10337c 10339c
  IOLIB(hpib_1)                     514d   570c   571c   573c   602d   658c   659c   661c
hpibsdc
  PRINTER(prtdvr)                   109d   123c   128c
hpibshort_msge_in
  AMIGO(csamigo)                    235c   312c   335c   414c
  CS80(cs80)                        376c   466c   494c
  DISCHPIB(dischpib)                198d   357d
  MINIT(cs80ir)                     757c   804c
  TAPEBKUP(cs80tbr)        ***      281c
hpibshort_msge_out
  AMIGO(csamigo)                    269c   292c
  CS80(cs80)                        395c   412c   429c   516c   532c   550c   589c
  DISCHPIB(dischpib)                196d   287d
  MINIT(cs80ir)                     616c   619c   643c   670c   695c   722c   754c   800c   841c   866c
  MINIT(iramigo)           ***      218c
  TAPEBKUP(cs80tbr)                  67c    94c   127c   150c   173c   203c   249c   278c
hpibupon_dxfr_comp
  AMIGO(amigodvr)          ***      768c
  CS80(cs80dvr)            ***     1116c
  DISCHPIB(dischpib)                202d   440d
```

```
hpibupon_ppol_resp
  AMIGO(amigodvr)                   693c   755c   778c   961c
  DISCHPIB(dischpib)                201d   416d
hpibwait_for_ppol
  AMIGO(amigodvr)                   475c   525c   682c   825c
  AMIGO(csamigo)                    311c   413c
  CS80(cs80)                        396c   440c   452c   465c   468c   480c   493c   495c   517c   533c   551c
  CS80(cs80dvr)                    1114c  1124c
  DISCHPIB(dischpib)                195d   256d
  MINIT(cs80ir)                     617c   620c   644c   671c   696c   723c   755c   765c   803c   813c   842c   867c
  MINIT(hminit)                     360c   365c   367c   392c   394c   399c   402c   404c   425c   438c   440c   448c
  MINIT(iramigo)           ***      216c
  TAPEBKUP(cs80tbr)                  68c    95c   128c   151c   174c   204c   250c   279c   291c
hpm
  HEAPT(hpm)               ***       37d
hptr
  EPROMS(eproms)                     30d    34d
hr
  CLOCK(clock)                      115d   120c   121c
hrecord
  EPROMS(eproms)                     24d    30d
hreturn
  HEAPT(hpm)                        193d   237c   255c   313c
hs_successfully_initiated
  PRINTER(prtdvr)                   148d   244c   258c   260c
htoc
  LIBRARIAN                         300d   474c   576c
htoctyp
  LIBRARIAN                         297d   300d
hue
  GEN(dgl_gen)                     3048d  3049d  3166d  3173c  3190d  3216c  3233c  3234c  3239c  3240c  3244c  3245c  3247c
hz50
  GLE_TYPES(gle_types)     ***     1246d
```

```
i
  A804XDVR(a804xdvr)                152d  154c  230d  235c  236c
  ASCII(asciimodule)                177d  181c  190c  194c  195c  202c  203c  204c  213d  215c  219c  222c  223c  224c
                                    225c  226c  234d  244c  245c  248c  251c  252c
  CONVERT(convert_text)              46d   59c   60c   64c   68c   73c   74c   78c   82c   90d   94c   96c   99c  100c
                                    103c  105c  106c  107c  110c  117c  120c  121c
  CRT(crt)                          231d  251c  254c  255c  329d  334c  335c  339d  344c  504d  513c  519c  524c  529c
                                    530c  531c  532c  551d  583c  587c  588c  602c  614c  615c  622c  624c  627c  628c
                                    632c  633c  641c  642c  643c  644c  645c  664d  675c  713d  718c
  CTABLE                           1212d 1529c 1531c 1538c 1541c 1543c 1552c 1555c 1557c 1572c 1575c 1577c 1653s 1654s
                                   1660s 1661s 1662s 1664s 1670s 1671s 1672s 1674s 1680s 1681s 1682s 1684s 1692s 1693s
                                   1694s 1696s 1699s 1700s 1701s 1703s 1710s 1711s 1712s 1714s 1715s 1716s 1717s 1719s
                                   1726s 1727s 1728s 1730s 1737s 1738s 1739s 1741s 1753s 1754s 1755s 1757s 1763s 1764s
  C_HOOK                             79d   97c  100c  107c  128d  133c  134c
  DGL_C_OUT(dgl_confg_out)         11086d 11116c 11117c 11118c 11129c
  DGL_HPGL(dgl_hpgl)               17258d 17284c 17285c
  DGL_POLY(dgl_poly)               20202d 20223c 20237c 20411d 20646c 20647c 20662c 20712c 20714c 20759c 20760c 20773c 20775c 20778c
                                   20781c 20783c 20786c 20787c 20789c 20792c 20793c 20801c 20806c 20807c 20820c 20822c 20823c 20923d
                                   20954c 20960c 20962c 20969c 20973c 20976c 20978c 20980c
  DGL_RAS(dgl_raster)              17085d 17108c 17109c 17161c 17254d 17260c 17262c 17271c 17380d 17405c 17408c 17415c 17581d 17627c
                                   17628c 17629c 17635c 17637c 17638c 17642c 17643c 17654c 17655c
  EDRIVER(edriver)                   101d  212c  215c  217c  220c  222c  226c  228c  230c  232c  235c  237c  242c  243c
                                     251c  283c  285c  286c  288c  290c  291c  328d  331c  336d  339c  340c  345d  348c
  ETU                                149d  162c  163c  164c  176d  182c  195d  204c  205c  536d  541c  542c  543c  655d
                                     677c  679c  680c  714d  741d  746c  748c  749c  750c  751c  800c  801c  802c  803c
                                     806c  851d  980c  988c 1001c 1003c 1006c 1013c 1015c 1075c 1093c 1098c 1099c 1126c
                                    1127c 1128c 1165d 1181c 1182c 1190c 1218c
  GCRT(crtb)                         152d  173c  176c  183c  372d  400c  401c  402c  403c  419d  438c  439c  446c  447c
                                     455c  457c  463c  464c  498d  530d
  GEN(dgl_gen)                     3152d 3155c 3157c
  GLE_HPGL(gle_hpgl_out)           7069d 7074c 7075c 7140d 7150c 7151c 7514d 7583c 7584c
  GLE_HPGLI(gle_hpgl_in)           18042d 18047c 18048c 18079d 18088c 18329d 18356c 18357c
  GLE_HPIB(gle_hpib_io)            10510d 10519c 10520c 10531d 10540c 10542c 10543c 10544c 10545c 10546c 10547c
  GLE_RGL(gle_ras_out)              8152d 8201c 8202c 8203c 8206c 8213c 8214c 8225d 8233c 8236c 8248d 8268c 8293d 8320d
                                    8329c 8330c 8332c 8336c 8345c 8346c 8349c 8350c 8352c 8353c 8356c 8357c 8359c 8360c
                                    8362c 8363c 8378d
  GLE_SMARK(gle_smark)              5094d 5153c 5154c
  GLE_STEXT(gle_stext)              4026d 4033c 4035c 4037c 4038c 4039c 4041c 4042c 4043c
  GLE_UTLS(gle_utls)               21045d 21051c 21053c 21055c 21060c 21061c 21064c 21065c 21066c 21077d 21081c 21082c 21083c 21085c
                                   21087c 21093c 21094c 21096c 21098c 21100c 21101c 21110d 21114c 21115c 21123d 21128c 21130c 21131c
  HEAPT(hpm)                         126d  136d  144c  145c  147c  148c  182c  183c  185c  186c
  HPHIL(hphil)                       185d  196c  228d  241c  243c  249c  251c
  INIT                        ***   2519d
  INIT(fs)                           748d  803d  804d  805d  806d  811d  812d  820d  822d  833d  835d
  INIT(initunits)                   2172d 2174c 2176c 2177c 2236d 2239c 2241c 2255d
  INIT(ldr)                         2314d 2316c 2317c 2386d 2392c 2393c 2395c
  INIT(misc)                         247d  250c  252c  253c  517d  526c  538d  555d  567c  597c  598c  599c  683c  684c
  INITLOAD                          1728d 1730c 1731c 1737d 1741c 1742c 1743c 1746c 1749c 1750c 1752c
  INITLOAD(bootdammodule)            614d  652d  661c  662c  696c  700c  783d  790c
  INITLOAD(loader)                   887d 1224d 1230c 1235c 1236c 1238c 1239c 1251d 1267c 1269c 1401d 1443c 1465c 1531c
                                    1591d 1615c 1616c 1617c
  INITLOAD(mini)                     373d  378c  379c
  IOLIB(general_2)                   835d  863c  867c  868c  869c  870c  874c  875c  876c  877c  878c  879c  882c  883c
                                     915d  926c  927c  938d  947c  949c  950c  951c  952c  953c  954c  955c  968d  978c
                                     980c  981c  982c  983c  984c  994d 1002c 1003c 1072d 1081c
  IOLIB(general_4)                  1615d 1629c 1631c 1642d 1650c 1652c
  KEYS(keys)                         253d  280c  282c  284c  285c  289c  291c  301c  303c  304c  318c  320c  321c  322c
                                     324c  325c  329c  330c  335c  343c  377d  464c
  LIB(dgl_lib)                     20679d 20688c 20689c 20695d 20704c 20705c 21114d 21302d 21317c
  LIBRARIAN                          109d  118c  154d  216c  242d  285d  288c  290c  389d  399c  400c  459d  461c 1345d
                                    1354c 1356c 1369d 1389c 1391c 1394c 1656d 1689c 1723d 1747c 1761c 1768c 1770c 1771c
                                    1772c 1774c 1785c 1786c 1792c 1796c 2004d 2033c 2046c 2056c 2058c 2059c 2060c 2062c
                                    2079c 2080c 2081c 2103c 2107c 3119d 3247d 3254c 3255c 3256c 3259c 3260c 3271d 3278c
                                    3279c 3280c 3283c
  LIFDAM(lifmodule)                  91d   93c   94c  100d  105c  162d  181c  229d  233c  235c  237c  244d  265c  341d

                                    349c  484d  516c  522c  529c
  M68KSYS(ci)                       176d  211d  212c  217c  218c  224c  225c  233d  234c  238c  241c  244c  248c  249c
                                    250c  261c  264c  265c  266c  272c  275c  276c  278c  293d  313d  366c  367c  368c
                                    369c  371c  374c  375c  376c  378c  379c  394d  659d  671d  679c  702c  748d  752c
                                    754c  755c  757c  788d  813c  842d  848c  983d 1033d 1085c
  MINIT(bminit)                      66d   83c   84c   86c
  MINIT(qminit)                    1037d 1046c 1047c
  MINIT(xminit)                    1411d 1668c 1669c 1708c 1712c 1717c 1720c 1736c
  MIUI                               54d   77c   79c   80c   82c   83c   86c   87c  291d  309c  310c  458d  466c  467c
  MOREFSYS(mfs)                      61d   66c   67c   68c   77c   78c   79c  400d  415c  564d  631c  632c  645c  646c
  MOUSE(mouse)                       68d   70c   71c   72c  132d  134c  135c  138c  148c
  NONUSKBD1(non_us_kbd1)            113d  136c  138c  139c  147c  149c  150c  171c
  NONUSKBD2                         125d  144c  145c  147c  148c  153c
  SRMDAM(srmdammodule)              100d  102c  103c  104c  106c  185d  188c  189c  466d  471d  481c  482c  485c  487c
                                    508d  513d  533c  536c  539c  543c  545c  550c  574d  594c  597c  600c  629c  634c
                                    655d  910d  944c  960d  991d 1016d 1019d 1033d 1045c 1049c 1070c 1072c 1418c 1526d
                                   1600c 1601c 1602c 1603c 1608c 1610c
  SRM_DRV                          1772c 1817c 1855c 1874c 1894c 1953c 2009c 2034c 2036c 2056c 2095c 2102c 2103c
  SRM_DRV(srm)                      206d  214d 1499c 1693c 1694c 1767c
  SYSDEVS(sysdevs)                  494d  499c  500c  598d  644c  648c
  UCSD_AM(ucsd_am)                   49d   63c   71d   84c
  UCSD_DAM(ucsdmodule)              138d  141c  142c  144c  145c  149c  153c  184d  195c  199c  204c  220c  222c  224c
                                    232c  254c  256c  257c  265d  272c  273c  275c  279c  284d  287c  288c  302d  306c
                                    307c  317c  380d  387c  389c  390c  429d  432c  433c  438d  444c  445c  448c  449c
                                    453c  474d  501c  531d  535c  536c  542c  560c  563c  569d  581c  587c  589c  590c
                                    613c  617c  621c
i1
  UCSD_DAM(ucsdmodule)              570d  598c  604c  609c
i2
  UCSD_DAM(ucsdmodule)              570d  600c  601c  606c  610c  612c
iand
  A804XDVR(a804xdvr)          ***   165c
iarray
  CRT(crt)                          546d  547d
  GCRT(crtb)                        416d  417d
ibadclose
  INIT(misc)                  ***   333c
  INITLOAD(sysglobals)        ***   190d
ibadfiletype
  INIT(misc)                  ***   340c
  INITLOAD(sysglobals)        ***   192d
  SRM_DRV(srm)                ***  1184c
  UCSD_DAM(ucsdmodule)        ***   213c
ibadformat
  ASCII(asciimodule)          ***   250c
  INIT(misc)                  ***   318c
  INITLOAD(sysglobals)        ***   186d
  IOLIB(general_2)            ***   893c
  M68KSYS(ci)                 ***   766c
  MOREFSYS(mfs)                     386c  508c  511c
ibadpass
  INIT(misc)                  ***   350c
  INITLOAD(sysglobals)        ***   194d
  SRMDAM(srmdammodule)              190c  685c  692c  699c 1107c 1545c 1594c
  SRM_DRV(srm)                ***  1160c
```

```
ibadrequest
  AMIGO(amigodvr)                 ***    639c
  ASCII(asciimodule)              ***    309c
  BUBBLES(bubble)                 ***    118c
  CRT(crt)                        ***    498c
  CS80(cs80dvr)                   ***   1399c
  ETU                             ***    629c
  F9885(f9885dvr)                 ***    290c
  GCRT(crtb)                      ***    366c
  INIT(misc)                             343c   476c   506c   687c
  INITLOAD(bootdammodule)         ***    726c
  INITLOAD(mini)                  ***    518c
  INITLOAD(sysglobals)            ***    193d
  KEYS(keys)                      ***    138c
  LIFDAM(lifmodule)                      488c  1065c  1086c  1168c
  M68KSYS(ci)                     ***    741c
  SRMAM(srmammodule)              ***    212c
  SRMDAM(srmdammodule)            ***   1798c
  UCSD_AM(ucsd_am)                ***    263c
  UCSD_DAM(ucsdmodule)                   476c   673c
ibadtitle
  ETU                             ***    329c
  INIT(misc)                             311c   439c
  INITLOAD(bootdammodule)         ***    621c
  INITLOAD(sysglobals)            ***    185d
  LIFDAM(lifmodule)                      103c   171c   177c   189c   222c   394c  1071c  1162c
  SRMDAM(srmdammodule)                   124c   145c   694c   753c   764c   783c   998c  1197c  1273c  1287c  1581c  1592c  1604c
  SRM_DRV(srm)                    ***   1142c
  UCSD_DAM(ucsdmodule)                   175c   441c   462c   486c   577c   653c   666c
ibadunit
  GLE_KNOB(gle_knob_in)           ***  18035c
  INIT(misc)                      ***    306c
  INITLOAD(sysglobals)            ***    184d
  SRM_DRV(srm)                    ***   1136c
  UNITIO(uio)                     ***     42c
ibadvalue
  ETU                                    163c   168c
  INIT(misc)                      ***    341c
  INITLOAD(sysglobals)            ***    192d
  SRMDAM(srmdammodule)                   233c   272c   297c   324c
  SRM_DRV(srm)                    ***   1144c
  UCSD_DAM(ucsdmodule)            ***    491c
icantstretch
  INIT(misc)                      ***    342c
  INITLOAD(sysglobals)            ***    192d
icc
  CS80(cs80)                             401d   439c   464c   492c
icode
  INIT(ldr)                             2409d  2489c  2504c
icuc
  CS80(cs80)                             416d   451c   479c
icuval
  CS80(cs80)                             355d   556d
  CS80(cs80dvr)                   ***   1110c
```

```
id
  BUBBLES(bubble)                        140d   157c
  CTABLE(ctr)                            336d   756d   768c
  EDRIVER(edriver)                        63d   139c   155c
  GLE_RGL(gle_ras_out)            ***   8378d
  HPHIL(hphil)                    ***    257c
  MINIT(cs80ir)                          631d   639c   640c   641c   642c   643c
  MIUI                                   395d   430d
  SRM_DRV                         ***   1931c
  SRM_DRV(srm)                    ***    311d
  SYSDEVS(sysdevs)                ***    313d
idc_ident
  AMIGO(amigodvr)                        502d   515c
idc_init
  DC_DRV(init_dc)                 ***    614c
  DC_DRV(intdc)                          244d   263d
idc_isr
  DC_DRV(init_dc)                 ***    615c
  DC_DRV(intdc)                          245d   305d
idc_rdb
  DC_DRV(init_dc)                 ***    616c
  DC_DRV(intdc)                          246d   312d
idc_rds
  DC_DRV(init_dc)                 ***    620c
  DC_DRV(intdc)                          250d   348d
idc_rdw
  DC_DRV(init_dc)                 ***    618c
  DC_DRV(intdc)                          248d   330d
idc_tff
  DC_DRV(init_dc)                 ***    622c
  DC_DRV(intdc)                          254d   408d
idc_wtb
  DC_DRV(init_dc)                 ***    617c
  DC_DRV(intdc)                          247d   322d
idc_wtc
  DC_DRV(init_dc)                 ***    621c
  DC_DRV(intdc)                          252d   356d
idc_wtw
  DC_DRV(init_dc)                 ***    619c
  DC_DRV(intdc)                          249d   340d
idchar
  EPROMS(eproms)                  ***     26d
idcmd
  HPHIL(hphil)                           181d   197c
ident
  AMIGO(amigodvr)                        428d   439d   449d   457d   458d   459d   460d   461d   462d   463d   468c   507d   512c   515c
                                         519c   558d   614c
  CS80(cs80dvr)                          982d  1027d
  CTABLE(scanstuff)                     1052d  1053d  1092d  1105c  1120c  1121c  1122c  1125c
  DISCHPIB(dischpib)                     200d   386d   398c   401c   406c   410d   412c
ident_table
  AMIGO(amigodvr)                        456d   468c   469c
ident_table_entries
  AMIGO(amigodvr)                        441d   452d   467c
ident_table_type
  AMIGO(amigodvr)                        452d   456d
identify
  M68KSYS(ci)                            181d   257c   288c
idirfull
  INIT(misc)                      ***    331c
  INITLOAD(sysglobals)            ***    190d
  LIFDAM(lifmodule)               ***    882c
  UCSD_DAM(ucsdmodule)            ***    216c
```

```
idirnotempty
  INIT(misc)                    ***    347c
  INITLOAD(sysglobals)          ***    194d
  SRM_DRV(srm)                  ***   1183c
idle
  CRT(crt)                             685c   690c
  GCRT(crtb)                    ***    501c
  INIT                          ***   2541c
  INIT(misc)                    ***    207d
  KEYS(keys)                           108c   575c
  M68KSYS(ci)                   ***    116c
idupfile
  INIT(misc)                    ***    315c
  INITLOAD(sysglobals)          ***    186d
  LIFDAM(lifmodule)                    975c  1073c
  SRMDAM(srmdammodule)                1139c  1152c
  SRM_DRV(srm)                  ***   1175c
  UCSD_DAM(ucsdmodule)          ***    445c
ieof
  AMIGO(amigodvr)               ***    611c
  ASCII(asciimodule)                    27c    82c
  BUBBLES(bubble)                       85c   104c
  CONVERT(convert_text)         ***     53c
  CS80(cs80dvr)                 ***   1393c
  ETU                                  462c   464c
  F9885(f9885dvr)               ***    285c
  INIT(misc)                           334c   493c   499c   613c   618c
  INITLOAD(mini)                ***    513c
  INITLOAD(sysglobals)          ***    190d
  MOREFSYS(mfs)                 ***    524c
  SRM_DRV(srm)                  ***   1178c
  UCSD_AM(ucsd_am)                      58c   164c
ierr
  DGL_HPGL(dgl_hpgl)                 17021d 17034c 17043c 17056c 17066d 17126c 17129c 17132c 17134c 17141c 17143c 17154c 17160c 17163c
                                     17166c 17169c 17171c 17174c 17175c
  DGL_INQ(dgl_inq)                    6015d  6091d  6171c  6176c  6180c  6444c  6465c  6478c  6492c  6507c  6519c  6535c  6551c
  DGL_RAS(dgl_raster)                17350d 17358c 17431d 17444c 17449c 17454c 17459c 17465c 17473c 17479c 17489c 17500c 17514c 17519c
                                     17524c 17534c 17536c 17541c
  LIB(dgl_lib)                       20024d 20027d 20039d 20054d 20061d 20068d 20084d 20489d 20530c 20532c 20535c 20755d 20763c 20764c
                                     20773d 20781c 20782c 20881d 20921c 20924c 20927c 20930c 21111d 21196d 21222c 21225c 21227c 21233c
                                     21238c 2124lc 21247d 21271c 21274c 21281c 21288c 21291c 21296d 21323c 21325c 21367c
ifc_line
  IOLIB(hpib_2)                        1818c  1820c
  KERNEL(iodeclarations)        ***    389d
ifilelocked
  INIT(misc)                    ***    345c
  INITLOAD(sysglobals)          ***    193d
  LOCKMOD(lockmodule)                   45c    56c    86c
  SRMDAM(srmdammodule)          ***   1480c
  SRM_DRV(srm)                  ***   1169c
ifilenotdir
  INIT(misc)                    ***    351c
  INITLOAD(sysglobals)          ***    194d
  SRM_DRV(srm)                  ***   1182c
ifileunlocked
  INIT(misc)                    ***    346c
  INITLOAD(sysglobals)          ***    193d
  LOCKMOD(lockmodule)           ***     70c
  SRMAM(srmammodule)                   194c   201c
ignored_key
  KEYS(keys)                           199c   204c   295c   296c   338c   339c   340c   533c
  NONUSKBD1(non_us_kbd1)                44c    49c    52c    56c    65c   152c
  SYSDEVS(sysdevs)              ***    258d
```

```
il
  INITLOAD                           1759d  1765c  1766c  1771c  1772c
ilist
  DGL_HPGL(dgl_hpgl)                 17019d 17052c 17064d 17077c 17084c 17091c 17098c 17099c 17100c 17106c 17107c 17108c 17114c 17115c
                                     17116c
  DGL_INQ(dgl_inq)                    6013d  6089d  6305c  6306c  6311c  6312c  6318c  6319c  6324c  6325c  6330c  6331c  6336c  6337c
                                      6342c  6343c  6348c  6349c  6354c  6359c  6364c  6369c  6374c  6375c  6380c  6385c  6390c  6391c
                                      6396c  6397c  6402c  6403c  6408c  6409c  6414c  6415c  6420c  6421c  6426c  6430c  6433c  6442c
                                      6451c  6462c  6476c  6489c  6504c  6505c  6515c  6516c  6531c  6532c  6533c  6544c  6545c  6547c
                                      6548c
  DGL_RAS(dgl_raster)                17350d 17429d 17481c 17491c 17502c 17514c 17526c
  DGL_VARS(dgl_vars)                  1124d  1129d
  LIB(dgl_lib)                       20059d 20066d 20753d 20764c 20771d 20782c
  LIBRARIAN                           1715d  1746c  1747c  1754c  1761c  1768c  1771c  1772c  1774c  1781c  1786c  1998d  2032c  2033c
                                      2039c  2046c  2056c  2059c  2060c  2062c  2070c  2080c
illegal_access_to_spare_track
  AMIGO(amigodvr)               ***    813c
  AMIGO(csamigo)                ***     77d
illegal_drive_type
  AMIGO(amigodvr)               ***    878c
  AMIGO(csamigo)                ***     70d
illegal_opcode
  AMIGO(csamigo)                ***     69d
  CS80(cs80)                    ***    136d
  CS80(cs80dsr)                 ***    672c
  TAPEBKUP(cs80tbdvr)                   552c   784c
illegal_parallel_operation
  CS80(cs80)                    ***    165d
illegal_parameter
  CS80(cs80)                    ***    140d
  CS80(cs80dsr)                 ***    674c
  TAPEBKUP(cs80tbdvr)                   554c   786c
illegalchar
  CRT(crt)                              352d   369c   389c
ilostfile
  INIT(misc)                    ***    310c
  INITLOAD(sysglobals)          ***    185d
  LIFDAM(lifmodule)                    1108c  1110c
  SRMDAM(srmdammodule)                 1280c  1348c
  SRM_DRV(srm)                  ***   1139c
  UCSD_DAM(ucsdmodule)          ***    348c
ilostunit
  INIT(misc)                    ***    309c
  INITLOAD(sysglobals)          ***    185d
  LIFDAM(lifmodule)             ***    309c
  SRMDAM(srmdammodule)          ***    401c
  UCSD_DAM(ucsdmodule)          ***    132c
ilp
  INITLOAD(loader)              ***    914d
  LIBRARIAN                           1747c  1754c  1761c  1768c  1771c  1772c  1774c  1781c  1786c  2033c  2039c  2046c  2056c  2059c
                                      2060c  2062c  2070c  2080c
im
  LIBRARIAN                           3306d  3372c  3374c  3375c  3381c  3382c  3384c  3385c  3386c  3387c  3396c  3397c  3400c  3426c
                                      3428c  3429c  3447c  3448c  3459c  3460c  3485c  3486c  3493c  3494c  3497c
  MINIT(cs80ir)                        658d   666c   667c   668c   669c   670c
  TAPEBKUP(cs80tbr)                     82d    90c    91c    92c    93c    94c
immediate
  LIBRARIAN                            619d   642c   679c   698c   703c   935c   940c   955c   968c
immop
  LIBRARIAN                            654d   683c
immoptype
  LIBRARIAN                            652d   654d
implem
  MIUI                                 416d   450d
```

```
implicit_unlock
  SRM_DRV                          1855c  1953c  2009c  2034c  2095c
  SRM_DRV(srm)                      571d   677d   711d   733d   752d   801d
in_buffer
  DC_DRV(intdc)                     428c   532c
  DISCHPIB(dischpib)          ***   251c
  IOLIB(general_4)                 1391c  1402c  1671c
  KERNEL(general_0)           ***  1211c
  KERNEL(iodeclarations)            497d   511d   525d
  PRINTER(prtdvr)             ***   291c
  SRM_DRV(srm)                     1327c  1331c
inbuf
  LIBRARIAN                         314d   481c  1295c  1296c  1310c
inc
  CRT(crt)                          553d   576c   580c   584c
  DGL_POLY(dgl_poly)               20547d 20566c 20569c 20574c 20575c 20578c 20579c 20585c
increment
  LIFDRM(lifmodule)                 686d   702c   705c   708c   710c
increment_and_test_fun
  CTABLE                           1222d  1225c  1530c  1539c  1553c  1573c
indeftable
  MINIT(xminit)                    1439d  1446c  1448c
index
  AMIGO(amigodvr)                   443d   467c   468c   469c
  CS80(cs80dsr)                     789d   918c   919c   926c   927c   930c   932c
  CS80(cs80dvr)                    1036d  1063c  1064c  1068c  1069c
  CTABLE                           1212d  1268d  1270c  1292d  1295c  1297c  1298c  1299c  1301c  1303c  1305c  1306c  1440c  1442c
                                   1443c  1444c  1464c  1467c  1513c  1514c  1524c  1525c  1834c  1836c  1837c  1838c
  C_HOOK                             80d   100c   102c   105c
  DGL_HPGL(dgl_hpgl                17184d 17196c 17198c 17205d 17208c 17212d
  DGL_INQ(dgl_inq)                  6017d  6022d  6034d  6047c  6048c  6060d  6073c  6074c  6164d  6174c  6175c  6182c  6449c  6450c
                                    6460c  6461c  6487c  6488c  6502c  6503c  6513c  6514c  6529c  6530c  6542c  6543c
  DGL_POLY(dgl_poly)               20013d 20014d 20015d 20020d 20102d 20116c 20117c 20118c 20119c 20126d 20136c 20137c 20142d 20152c
                                   20153c 20160c 20165d 20178c 20182c 20187c 20554c 20566c 20569c 20571c 20574c 20575c 20577c 20578c
                                   20579c 20585c
  DGL_RAS(dgl_raster)              17045d 17057c 17060c 17168d 17177c 17189c 17197d 17211c 17213c 17214c 17221c 17226c 17381d 17408c
                                   17410c 17413c
  DGL_VARS(dgl_vars)                1133d  1135d  1137d
  DISCHPIB(bkgnd)                     79d    90c   111c   112c   113c
  GCRT(crtb)                         153d   176c   178c   181c
  GLE_RGL(gle_ras_out)        ***   8062d
  LIB(dgl_lib)                     20035d 20036d 20037d 20086d 20408d 20418c 20424c 20431d 20440c 20441c 20442c 20447d 20457c 20458c
                                   20459c 20461c 20463c 20464c 20469d 20479c 20482c 20483c
  LIBRARIAN                         1857d  1902c  1903c  2337d  2339c  2343c
  MINIT(qminit)                      937d  1007c  1008c  1011c
  TAPEBKUP(cs80tbdvr)                456d   485c   486c   488c   704d   866c   869c   870c   871c   874c
  UCSD_AM(ucsd_am)                    45d    62c    63c    94c   116c   117c   118c   119c   139c   184c   202c   203c   204c   205c
                                     218c   221c   233c   234c   238c   240c   241c   244c   245c   246c   247c
index_color
  GLE_GEN(gle_gen)            ***   2114c
  GLE_HPGL(gle_hpgl_out)      ***   7540c
  GLE_RGL(gle_ras_out)        ***   8408c
  GLE_TYPES(gle_types)        ***   1046d
index_def
  GLE_SMARK(gle_smark)              5022d  5029d
indexlist
  INITLOAD(loader)                   914d  1043d
indx
  SRM_DRV(srm)                      1012d  1020d  1554d  1566c  1592d  1605c
ineedtempdir
  INIT(misc)                  ***    353c
  INITLOAD(sysglobals)        ***    195d
  SRMDAM(srmdammodule)        ***   1132c
```

```
infilename
  LIBRARIAN                          63d  3081c  3082c  3084c  3086c  3089c  3359c  3363c  3365c  3411c  3590c  3635c  3670c
info
  BUBBLES(bubble)                    57d   67c   68c   80c   99c  108c  109c  127d  130c  131c  145d  162c  163c  164c
                                    169c
  EDRIVER(edriver)                   43d  103d  112c  124c  142c  147c  207c  294c  326d  331c
info1
  DGL_C_OUT(dgl_config_out)        11218c 11234c 11238c 11255c 11275c 11321c 11416c
  DGL_HPGL(dgl_hpgl)               17046c 17051c 17178c 17196c 17208c 17293c
  DGL_HPGLI(dgl_hpgli)             18029c 18031c 18072c 18080c 18090c 18101c 18114c
  DGL_KNOB(dgl_knob)               18032c 18112c 18122c 18135c 18142c 18160c
  DGL_POLY(dgl_poly)               20073c 20094c 20999c 20990c
  DGL_RAS(dgl_raster)              17057c 17182c 17183c 17187c 17189c 17191c 17213c 17279c 17299c 17333c 17481c 17502c 17503c 17504c
                                   17506c 17526c 17612c 17647c
  GEN(dgl_gen)                ***   3541c
  GLE_HPGL(gle_hpgl_out)             7126c  7149c  7182c  7190c  7192c  7193c  7205c  7242c  7256c  7263c  7274c  7283c  7291c  7299c
                                     7307c  7391c  7410c  7412c  7438c  7443c  7447c  7452c  7454c  7496c  7679c
  GLE_HPGLI(gle_hpgl_in)           18071c 18087c 18108c 18116c 18118c 18119c 18148c 18161c 18170c 18178c 18186c 18194c 18202c 18254c
                                   18258c 18301c 18317c 18385c
  GLE_KNOB(gle_knob_in)            18086c 18092c 18098c 18127c 18177c 18200c 18210c 18257c 18259c 18264c
  GLE_RGL(gle_ras_out)              8183c  8211c  8277c  8329c  8332c  8345c  8383c  8429c  8509c  8516c
  GLE_SCLIP(gle_sclip)        ***   6017c
  GLE_SMARK(gle_smark)              5118c  5129c  5132c  5140c  5145c  5148c  5167c  5182c  5190c
  GLE_STEXT(gle_stext)              4062c  4065c  4074c  4077c  4086c  4089c  4098c  4101c
  GLE_TYPES(gle_types)              1021d  1197d
  LIB(dgl_lib)                     20226c 20394c 20442c 20549c 20726c 20744c 20808c 20834c 20838c 20981c 20987c 20992c 21002c 21008c
                                   21011c 21130c 21183c 21187c 21219c 21268c 21320c 21342c
info2
  DGL_C_OUT(dgl_config_out)        11240c 11256c 11276c 11322c 11323c
  DGL_HPGL(dgl_hpgl)               17197c 17294c
  DGL_HPGLI(dgl_hpgli)             18029c 18063c 18073c 18080c 18091c 18101c
  DGL_KNOB(dgl_knob)               18074c 18087c 18100c 18113c 18123c 18136c 18142c
  DGL_POLY(dgl_poly)               20079c 20086c 20095c 20998c 21001c 21004c 21005c
  DGL_RAS(dgl_raster)              17058c 17214c 17280c 17337c 17527c 17613c
  GEN(dgl_gen)                ***   3542c
  GLE_HPGL(gle_hpgl_out)             7186c  7190c  7193c  7194c  7206c  7246c  7257c  7264c  7275c  7284c  7292c  7300c  7307c  7439c
                                     7456c  7498c  7681c
  GLE_HPGLI(gle_hpgl_in)           18112c 18116c 18119c 18120c 18152c 18162c 18171c 18179c 18187c 18195c 18202c 18255c 18259c 18303c
                                   18386c
  GLE_KNOB(gle_knob_in)            18087c 18093c 18099c 18128c 18178c 18189c 18201c 18258c 18260c 18265c
  GLE_RGL(gle_ras_out)              8185c  8214c  8228c  8278c  8305c  8329c  8345c  8387c  8388c  8389c  8390c  8463c  8509c
  GLE_SCLIP(gle_sclip)        ***   6018c
  GLE_SMARK(gle_smark)              5118c  5130c  5133c  5146c  5149c  5168c  5191c
  GLE_STEXT(gle_stext)              4062c  4066c  4074c  4078c  4086c  4090c  4098c  4102c
  GLE_TYPES(gle_types)              1022d  1198d
  LIB(dgl_lib)                     20227c 20550c 20809c 20835c 20839c 20982c 21003c 21131c 21184c 21220c 21269c 21321c 21343c
info3
  DGL_C_OUT(dgl_config_out)        11176c 11180c 11241c 11257c 11318c 11319c
  DGL_HPGL(dgl_hpgl)               17198c 17199c
  DGL_HPGLI(dgl_hpgli)             18074c 18081c 18092c 18099c 18108c
  DGL_KNOB(dgl_knob)               18111c 18124c 18131c 18154c
  DGL_POLY(dgl_poly)               20080c 20087c 20994c
  DGL_RAS(dgl_raster)         ***  17059c
  GEN(dgl_gen)                ***   3543c
  GLE_HPGL(gle_hpgl_out)             7184c  7196c  7198c  7199c  7207c  7244c  7258c  7265c  7276c  7285c  7293c  7301c  7308c  7440c
                                     7451c  7499c  7680c
  GLE_HPGLI(gle_hpgl_in)           18110c 18122c 18124c 18125c 18150c 18163c 18172c 18180c 18188c 18196c 18203c 13269c 18272c 18274c
                                   18305c 18387c
  GLE_KNOB(gle_knob_in)            18088c 18094c 18100c 18121c 18122c 18158c 18164c 18202c 18266c
  GLE_RGL(gle_ras_out)              8186c  8473c  8510c
  GLE_SCLIP(gle_sclip)        ***   6019c
  GLE_SMARK(gle_smark)              5119c  5169c
  GLE_TYPES(gle_types)              1023d  1199d
  LIB(dgl_lib)                     20233c 20236c 20993c 21004c 21132c 21344c
```

```
info4
  DGL_HPGL(dgl_hpgl)          ***  17200c
  DGL_POLY(dgl_poly)               20081c 20088c 20995c
  DGL_RAS(dgl_raster)         ***  17060c
  GEN(dgl_gen)                ***   3544c
  GLE_HPGL(gle_hpgl_out)            7188c  7196c  7199c  7200c  7208c  7248c  7259c  7266c  7277c  7286c  7294c  7302c  7308c  7441c
                                    7682c
  GLE_HPGLI(gle_hpgl_in)           18114c 18122c 18125c 18126c 18154c 18164c 18173c 18181c 18189c 18197c 18203c 18388c
  GLE_KNOB(gle_knob_in)            18089c 18095c 18101c 18267c
  GLE_RGL(gle_ras_out)              8187c  8510c
  GLE_SCLIP(gle_sclip)        ***   6020c
  GLE_SMARK(gle_smark)              5119c  5170c
  GLE_TYPES(gle_types)              1024d  1200d
  LIB(dgl_lib)                     20984c 21005c 21133c 21345c
info_ptr1
  DGL_C_OUT(dgl_confg_out)         11236c 11258c 11324c
  DGL_HPGL(dgl_hpgl)               17045c 17049c 17177c
  DGL_POLY(dgl_poly)               21051c 21096c
  DGL_RAS(dgl_raster)              17218c 17281c 17332c
  GLE_HPGL(gle_hpgl_out)            7126c  7148c
  GLE_HPGLI(gle_hpgl_in)           18071c 18086c
  GLE_RGL(gle_ras_out)              8279c  8328c  8385c
  GLE_SMARK(gle_smark)        ***   5139c
  GLE_TYPES(gle_types)              1026d  1202d
  LIB(dgl_lib)                ***  20725c
info_ptr2
  DGL_C_OUT(dgl_confg_out)         11237c 11259c 11325c
  DGL_POLY(dgl_poly)               21052c 21097c
  GLE_TYPES(gle_types)        ***   1027d
infoptr
  BUBBLES(bubble)                    57d   127c   145d
  EDRIVER(edriver)                   71d   103d
inforec
  M68KSYS(ci)                        44d    50d
inforecptr
  M68KSYS(ci)                        50d    59d
inform_operator
  PRINTER(prtdvr)                   165d   249c
infost
  LIBRARIAN                          56d   393c  1340c  1362c  1409c  1418c
infostart
  LIBRARIAN                          83d  2737d  2747c  2825c  2953c  2975c  3008c  3124c
ininfo
  ETU                                78d   335c   388c   389c   391c   399c   885c   890c   918c  1023c  1228c
init
  GLE_HPGLI(gle_hpgl_in)      ***  18240d
init_913x_abc
  MINIT(hminit)                     409d   491c
init_9895_913x_chinook
  MINIT(hminit)                     332d   490c
init_aspect
  DGL_VARS(dgl_vars)          ***   1182d
  LIB(dgl_lib)                ***  21413c
init_bkgnd
  DISCHPIB                    ***    26d
init_c
  C_HOOK                             125d   145c
init_char_height
  DGL_VARS(dgl_vars)          ***   1160d
  LIB(dgl_lib)                ***  21435c
init_char_height_factor
  DGL_VARS(dgl_vars)          ***   1162d
  LIB(dgl_lib)                ***  21150c
```

```
init_char_rot_h
  DGL_VARS(dgl_vars)          ***   1164d
  LIB(dgl_lib)                     21155c 21438c
init_char_rot_w
  DGL_VARS(dgl_vars)          ***   1163d
  LIB(dgl_lib)                     21154c 21437c
init_char_width
  DGL_VARS(dgl_vars)          ***   1159d
  LIB(dgl_lib)                ***  21434c
init_char_width_factor
  DGL_VARS(dgl_vars)          ***   1161d
  LIB(dgl_lib)                ***  21148c
init_cmd_buf
  MINIT(iramigc)                     210d   215c
init_color
  DGL_VARS(dgl_vars)          ***   1156d
  LIB(dgl_lib)                     21165c 21171c 21444c 21448c
init_color_table
  DGL_RAS(dgl_raster)              17025d 17187c 17240c 17628c
init_color_table_def
  DGL_RAS(dgl_raster)              17022d 17025d
init_cpx
  DGL_VARS(dgl_vars)          ***   1167d
  LIB(dgl_lib)                     21176c 21426c 21429c
init_cpy
  DGL_VARS(dgl_vars)          ***   1168d
  LIB(dgl_lib)                     21177c 21427c 21430c
init_d_cmd
  AMIGO(csamigo)              ***     56d
  MINIT(iramigo)              ***    215c
init_dc
  DC_DRV                      ***    718d
  DC_DRV(init_dc)             ***    569d
init_dev_adr
  DGL_VARS(dgl_vars)          ***   1170d
  LIB(dgl_lib)                     21409c 21411c
init_discint
  DI_DRV                      ***    292d
  DI_DRV(init_discint)        ***    178d
init_display_lim
  DGL_VARS(dgl_vars)          ***   1187d
  LIB(dgl_lib)                     21040c 21419c
init_dma
  DMA_DRV                     ***    285d
  DMA_DRV(init_dma)           ***    152d
init_gpio
  G_DRV                       ***    304d
  G_DRV(init_gpio)            ***    192d
init_hpib
  H_DRV                       ***    315d
  H_DRV(init_hpib)            ***    193d
init_linestyle
  DGL_VARS(dgl_vars)          ***   1157d
  LIB(dgl_lib)                     21163c 21172c 21445c
init_linewidth
  DGL_VARS(dgl_vars)          ***   1158d
  LIB(dgl_lib)                     21173c 21446c
init_locator_lim
  DGL_VARS(dgl_vars)          ***   1192d
  LIB(dgl_lib)                     21083c 21421c
init_root_password
  SRM_DRV(srm)                ***    616d
```

```
init_rs
  RS_DRV                          ***    291d
  RS_DRV(init_rs)                 ***    189d
init_scanstuff
  CTABLE                          ***   1438c
  CTABLE(scanstuff)                     1039d  1149d
init_segmenter
  SEGMENTER(asm)                  ***     75d
  SEGMENTER(segmenter)            ***     24d
init_srm
  SRMDAM                          ***     26d
init_tapebuf
  CS80                            ***   1414c
  CS80(tapebuf)                          49d    53d
init_timming_mode
  DGL_VARS(dgl_vars)              ***   1166d
  LIB(dgl_lib)                    ***  21447c
init_ucsd_am
  UCSD_AM                         ***    281c
  UCSD_AM(ucsd_am)                       32d   268d
init_ucsd_dam
  UCSD_DAM                        ***    687c
  UCSD_DAM(ucsdmodule)                   34d    58d
init_viewport
  DGL_VARS(dgl_vars)              ***   1177d
  LIB(dgl_lib)                    ***  21415c
init_vir_lim
  DGL_VARS(dgl_vars)              ***   1184d
  LIB(dgl_lib)                    ***  21414c
init_window
  DGL_VARS(dgl_vars)              ***   1172d
  LIB(dgl_lib)                    ***  21412c
init_with_d_bits
  MINIT(hminit)                          393c   447c
  MINIT(iramigo)                         191d   208d
inita804x
  A804XDVR                        ***    416c
  A804XDVR(a804xdvr)                      44d   377d   381c   407c
initbat
  BAT                             ***     30d
initbootdam
  INIT(initunits)                 ***   2162d
  INITLOAD                        ***   1674c
  INITLOAD(bootdammodule)                542d   730d
initclock
  CLOCK                           ***    185c
  CLOCK(clock)                           34d   173d
initcrt
  CRT                             ***     30d
initcrtb
  GCRT                            ***     29d
initdate
  M68KSYS(ci)                            148d   649c  1182c
initdisg
  MINIT(cs80ir)                          634d   640c
initdumbuf
  SRM_DRV(srm)                          1191d  1221c  1377c  1413c
initfilekinds
  INIT                            ***   2550c
  INIT(misc)                             214d   694d
initfiles
  INIT                                  2521d  2570c
initfkinds
  INITLOAD                              1702d  1781c
```

```
initfnames
  M68KSYS(ci)                            579d   650c  1179c
initheap
  M68KSYS(ci)                            941d  1167c
initial_pattern
  DGL_VARS(dgl_vars)              ***   1218d
initial_seek
  AMIGO(amigodvr)                        543d   627c   654d
initialize_bkgnd
  DISCHPIB                        ***    556c
  DISCHPIB(bkgnd)                         61d    82d
  MIUI                            ***    544c
initialize_media
  MINIT(cs80ir)                          561d   649d   672c
  MINIT(qminit)                   ***   1158c
  TAPEBKUP(cs80tbdvr)             ***    419c
  TAPEBKUP(cs80tbr)                       25d    73d    96c
initialize_medium
  MIUI                                   283d   549c
initialize_options_byte
  MINIT(qminit)                          895d  1064c  1073c  1085c  1158c
  TAPEBKUP                              1246c  1477c  1478c
  TAPEBKUP(cs80tbdvr)                    328d   419c
initiate_data_transfer
  AMIGO(amigodvr)                        546d   755c   756c   762d
initiate_diagnostic
  MINIT(cs80ir)                          560d   625d   645c
  MINIT(qminit)                   ***   1129c
initiate_transfer
  DISCHPIB(dischpib)                     437d   500c   504d
initisrib
  MINIT(xminit)                         1375d  1520c  1830c
initkeys
  KEYS                            ***    598c
  KEYS(keys)                             35d   577d
initlang
  KEYS(keys)                             231d   586c
  NONUSKBD1                       ***    212c
  NONUSKBD1(non_us_kbd1)                  33d    74d   122c   204c
initlink
  LIBRARIAN                             2972d  3641c
initloaj
  INITLOAD                        ***      2d
initloader
  INITLOAD                        ***   1782c
  INITLOAD(loader)                      1094d  1653d
initloop
  HPHIL                           ***    389c
  HPHIL(hphil)                            33d   353d   360c   373c   380c
initmedia
  MINIT(cs80ir)                          661d   667c
  TAPEBKUP(cs80tbr)                       85d    91c
initmouse
  MOUSE                           ***    206c
  MOUSE(mouse)                            33d   179d   196c   198c
initpage
  UCSD_AM(ucsd_am)                        48d    97c   108c   109c   116c   144c   202c
initsysisr
  INIT(initunits)                       2171d  2258c
initsysunit
  CTABLE(ctr)                     ***    862c
  INIT                            ***   2565c
  INIT(sdr)                             2276d  2313d
```

```
inittime
  CLOCK(clock)                      126d   175c
initunitable
  INIT(initunits)                  2235d  2261c
initunitentry
  INIT(initunits)                  2181d  2240c  2243c  2245c  2247c  2249c
initunits
  INIT                       ***   2514d
  INIT(initunits)            ***   2133d
  M68KSYS(ci)                       951d  1126c
  MINIT(xminit)              ***   1359d
initutil
  MINIT(cs80ir)                     603d   613c   683d   693c   713d   719c   745d   751c   781d   792c   793c
  TAPEBKUP(cs80tbr)                 267d   275c
initvects
  INIT                       ***   2536c
initworkfile
  M68KSYS(ci)                       624d   651c  1178c
inmount
  ETU                               386d   899c
inoaccess
  INIT(misc)                 ***    349c
  INITLOAD(sysglobals)       ***    194d
  SRM_DRV(srm)               ***   1162c
inodirectory
  ETU                        ***    252c
  INIT(misc)                 ***    339c
  INITLOAD(sysglobals)       ***    192d
  LIFDAM(lifmodule)                 273c   494c
  MIUI                       ***    121d
  SRMDAM(srmdammodule)       ***    835c
  TAPEBKUP                   ***   1052d
  UCSD_DAM(ucsdmodule)              117c   479c
```

```
inoerror
  AMIGO(amigodvr)                   571c   579c   595c   634c   648c   777c
  BUBBLES(bubble)                    40c    60c   110c
  CONVERT(convert_text)              51c    53c
  CRT(crt)                          199c   257c   408c
  CS80(cs80dsr)                     646c   653c   751c   765c
  CS80(cs80dvr)                    1119c  1135c  1232c  1333c  1342c  1356c  1368c  1382c  1387c
  CTABLE                     ***   1235c
  C_HOOK                     ***    110c
  DGL_RAS(dgl_raster)             17340c 17418c
  DISCHPIB(bkgnd)            ***    118c
  EPROMS(eproms)             ***     45c
  ETU                               102c   116c   126c   133c   139c   203c   236c   268c   314c   358c   371c   467c   552c   599c
                                    823c   905c   927c  1004c  1230c  1268c
  F9885(f9885dvr)                   180c   186c   187c   293c
  GCRT(crtb)                        186c   278c
  GLE_KNOB(gle_knob_in)           18051c 18065c
  INIT(misc)                        304c   522c   549c   565c   573c
  INITLOAD(bootdammodule)           690c   754c
  INITLOAD(loader)           ***   1121c
  INITLOAD(mini)                    377c   400c   455c   521c
  INITLOAD(sysglobals)       ***    184d
  KEYS(keys)                 ***     79c
  LIBRARIAN                         134c   140c   289c   291c
  LIFDAM(lifmodule)                  83c   253c   273c   493c   495c  1123c
  LOCKMOD(lockmodule)                39c    51c    54c    73c    80c    92c
  M68KSYS(ci)                       161c   340c   422c   538c   600c   635c   641c   680c   753c   761c   829c   850c  1181c
  MINIT(bminit)                      80c    87c
  MINIT(cs80ir)                     763c   810c
  MINIT(hminit)                     310c   501c
  MINIT(mminit)              ***    168c
  MINIT(qminit)                     952c   988c   997c  1001c  1140c  1277c
  MIUI                              120d   381c
  MOREFSYS(mfs)                     105c   385c   522c   527c   540c   545c   567c   674c
  PRINTER(prtdvr)            ***    270c
  SRMAM(srmammodule)                 73c    82c    88c    91c   120c   129c   135c   138c   156c   158c   183c
  SRMDAM(srmdammodule)               81c   245c   283c   479c   491c   522c   527c   531c   590c   733c   756c   759c   772c   790c
                                    798c   803c   809c   813c   828c   841c   854c   883c   923c   977c  1042c  1062c  1077c  1084c
                                   1089c  1099c  1111c  1117c  1120c  1128c  1131c  1141c  1144c  1153c  1186c  1188c  1199c  1225c
                                   1227c  1229c  1231c  1254c  1320c  1354c  1361c  1370c  1371c  1386c  1390c  1393c  1423c  1425c
                                   1460c  1463c  1478c  1484c  1501c  1505c  1509c  1511c  1546c  1548c  1597c  1638c  1691c  1704c
                                   1713c  1724c  1727c  1739c  1742c  1751c  1771c  1801c
  SRM_DRV(srm)                      1133c  1147c  1540c  1542c
  TAPEBKUP                   ***   1051d
  TAPEBKUP(cs80tbdvr)               348c   358c   477c   480c   526c   533c   632c   655c   736c   738c   748c   754c   761c   863c
                                    888c
  TAPEBKUP(cs80tbr)          ***    289c
  UCSD_AM(ucsd_am)                   93c   166c
  UCSD_DAM(ucsdmodule)               91c   117c   605c   607c   630c   675c
  UNITIO(uio)                        60c    81c    97c   107c   122c
inofile
  INIT(misc)                 ***    314c
  INITLOAD(bootdammodule)    ***    716c
  INITLOAD(sysglobals)       ***    186d
  LIFDAM(lifmodule)                 672c   985c  1068c
  SRMDAM(srmdammodule)              1082c  1126c  1369c
  SRM_DRV(srm)               ***   1180c
  UCSD_DAM(ucsdmodule)       ***    279c
inoroom
  INIT(misc)                 ***    312c
  INITLOAD(sysglobals)       ***    185d
  LIFDAM(lifmodule)                 508c   509c   896c
  SRM_DRV(srm)               ***   1174c
  UCSD_DAM(ucsdmodule)       ***    234c
```

```
inotclosed
  INIT(misc)                    ***    316c
  INITLOAD(sysglobals)          ***    186d
  SRM_DRV(srm)                  ***   1172c
inotdirect
  INIT(misc)                    ***    330c
  INITLOAD(sysglobals)          ***    189d
inotlockable
  INIT(misc)                    ***    344c
  INITLOAD(sysglobals)          ***    193d
  LOCKMOD(lockmodule)                   42c    67c    83c
  SRMDAM(srmdammodule)          ***    920c
inotondir
  INIT(misc)                    ***    352c
  INITLOAD(sysglobals)          ***    195d
  SRMDAM(srmdammodule)                 926c  1110c
  SRM_DRV(srm)                  ***   1165c
inotopen
  INIT(misc)                    ***    317c
  INITLOAD(sysglobals)          ***    186d
  SRM_DRV(srm)                  ***   1154c
inotreadable
  INIT(misc)                    ***    328c
  INITLOAD(sysglobals)          ***    189d
  MOREFSYS(mfs)                 ***    520c
inotvalidsize
  INIT(misc)                    ***    327c
  INITLOAD(sysglobals)          ***    189d
  SRM_DRV(srm)                  ***   1138c
inotwriteable
  INIT(misc)                    ***    329c
  INITLOAD(sysglobals)          ***    189d
inounit
  INIT(misc)                    ***    313c
  INITLOAD(bootdammodule)       ***    719c
  INITLOAD(sysglobals)          ***    185d
  MIU1                          ***    144c
  SRMDAM(srmdammodule)          ***    886c
  TAPEBKUP                      ***   1074c
inp
  SYSDEVS(sysdevs)                      240d   536c   543c   563c   640c
inpass
  SRMDAM(srmdammodule)                 1529d  1537c  1553c  1557c  1564c  1572c
inpdtable
  MINIT(xminit)                        1412d  1663c  1669c  1670c
input_cpx
  DGL_KNOB(dgl_knob)                  18066c 18098c
  GLE_HPGLI(gle_hpgl_in)              18258c 18390c
  GLE_KNOB(gle_knob_in)               18127c 18159c 18174c 18177c 18257c
  GLE_TYPES(gle_types)          ***    1239d
input_cpy
  DGL_KNOB(dgl_knob)                  18067c 18099c
  GLE_HPGLI(gle_hpgl_in)              18259c 18391c
  GLE_KNOB(gle_knob_in)               18128c 18160c 18175c 18178c 18258c
  GLE_TYPES(gle_types)          ***    1240d
input_echo
  GLE_GENI(gle_geni)            ***    3052c
  GLE_HPGLI(gle_hpgl_in)        ***   18374c
  GLE_KNOB(gle_knob_in)         ***   18224c
  GLE_TYPES(gle_types)          ***    1206d
input_esc
  LIB(dgl_lib)                        20056d 20750d
```

```
input_escapei
  GLE_GENI(gle_geni)            ***    3058c
  GLE_HPGLI(gle_hpgl_in)        ***   18375c
  GLE_KNOB(gle_knob_in)         ***   18225c
  GLE_TYPES(gle_types)          ***    1207d
input_escapeo
  GLE_GENI(gle_geni)            ***    3064c
  GLE_HPGLI(gle_hpgl_in)        ***   18376c
  GLE_KNOB(gle_knob_in)         ***   18226c
  GLE_TYPES(gle_types)          ***    1208d
input_handler_char_count
  GLE_GENI(gle_geni)            ***    3026c
  GLE_HPGLI(gle_hpgl_in)        ***   18382c
  GLE_KNOB(gle_knob_in)         ***   18219c
  GLE_TYPES(gle_types)          ***    1230d
input_handler_name
  DGL_INQ(dgl_inq)              ***    6546c
  GLE_HPGLI(gle_hpgl_in)        ***   18381c
  GLE_KNOB(gle_knob_in)         ***   18218c
  GLE_TYPES(gle_types)          ***    1229d
input_max_x
  DGL_KNOB(dgl_knob)                  18082c 18095c
  GLE_HPGLI(gle_hpgl_in)        ***   18386c
  GLE_KNOB(gle_knob_in)               18087c 18174c 18265c
  GLE_TYPES(gle_types)          ***    1235d
  LIB(dgl_lib)                  ***   21333c
input_max_y
  DGL_KNOB(dgl_knob)                  18084c 18097c
  GLE_HPGLI(gle_hpgl_in)        ***   18388c
  GLE_KNOB(gle_knob_in)               18089c 18175c 18267c
  GLE_TYPES(gle_types)          ***    1237d
  LIB(dgl_lib)                  ***   21335c
input_min_x
  DGL_KNOB(dgl_knob)                  18081c 18094c
  GLE_HPGLI(gle_hpgl_in)              18385c 18390c
  GLE_KNOB(gle_knob_in)               18086c 18174c 18264c
  GLE_TYPES(gle_types)          ***    1234d
  LIB(dgl_lib)                  ***   21332c
input_min_y
  DGL_KNOB(dgl_knob)                  18083c 18096c
  GLE_HPGLI(gle_hpgl_in)              18387c 18391c
  GLE_KNOB(gle_knob_in)               18088c 18175c 18266c
  GLE_TYPES(gle_types)          ***    1236d
  LIB(dgl_lib)                  ***   21334c
input_name
  DGL_INQ(dgl_inq)              ***    6543c
  GLE_HPGLI(gle_hpgl_in)              18140c 18141c 18142c 18158c 18159c 18167c 18168c 18176c 18184c 18192c 18356c 18357c 18363c
  GLE_KNOB(gle_knob_in)               18231c 18237c 18243c 18249c
  GLE_TYPES(gle_types)          ***    1227d
input_name_char_count
  DGL_INQ(dgl_inq)                      6540c  6542c  6544c
  GLE_GENI(gle_geni)            ***    3025c
  GLE_HPGLI(gle_hpgl_in)              18358c 18364c
  GLE_KNOB(gle_knob_in)               18232c 18238c 18244c 18250c
  GLE_TYPES(gle_types)          ***    1228d
input_res_x
  DGL_INQ(dgl_inq)                      6236c  6249c
  GLE_HPGLI(gle_hpgl_in)        ***   18378c
  GLE_KNOB(gle_knob_in)               18233c 18239c 18245c 18251c
  GLE_TYPES(gle_types)          ***    1232d
  LIB(dgl_lib)                        20281c 20907c 20908c
```

```
input_res_y
  DGL_INQ(dgl_inq)                    6237c  6250c
  GLE_HPGLI(gle_hpgl_in)      ***    18379c
  GLE_KNOB(gle_knob_in)              18234c 18240c 18246c 18252c
  GLE_TYPES(gle_types)        ***     1233d
  LIB(dgl_lib)                       20282c 20909c 20910c
inputreal
  MOREFSYS(mfs)                        44d   526c   544c
inq_1050
  DGL_INQ(dgl_inq)                    6103d  6134d  6303c
inq_1051
  DGL_INQ(dgl_inq)                    6103d  6135d  6309c
inq_1052
  DGL_INQ(dgl_inq)                    6103d  6136d  6315c
inq_1053
  DGL_INQ(dgl_inq)                    6103d  6137d  6322c
inq_1054
  DGL_INQ(dgl_inq)                    6103d  6138d  6328c
inq_1056
  DGL_INQ(dgl_inq)                    6104d  6139d  6334c
inq_1057
  DGL_INQ(dgl_inq)                    6104d  6140d  6340c
inq_1059
  DGL_INQ(dgl_inq)                    6104d  6141d  6346c
inq_1060
  DGL_INQ(dgl_inq)                    6105d  6142d  6352c
inq_1062
  DGL_INQ(dgl_inq)                    6105d  6143d  6357c
inq_1063
  DGL_INQ(dgl_inq)                    6105d  6144d  6362c
inq_1064
  DGL_INQ(dgl_inq)                    6105d  6145d  6367c
inq_1065
  DGL_INQ(dgl_inq)                    6106d  6146d  6372c
inq_1066
  DGL_INQ(dgl_inq)                    6106d  6147d  6378c
inq_1067
  DGL_INQ(dgl_inq)                    6106d  6148d  6383c
inq_1068
  DGL_INQ(dgl_inq)                    6106d  6149d  6388c
inq_1069
  DGL_INQ(dgl_inq)                    6106d  6150d  6394c
inq_1070
  DGL_INQ(dgl_inq)                    6107d  6151d  6400c
inq_1071
  DGL_INQ(dgl_inq)                    6107d  6152d  6406c
inq_1072
  DGL_INQ(dgl_inq)                    6107d  6153d  6412c
inq_1073
  DGL_INQ(dgl_inq)                    6107d  6154d  6418c
inq_1074
  DGL_INQ(dgl_inq)                    6107d  6155d  6424c
inq_1075
  DGL_INQ(dgl_inq)                    6108d  6156d  6429c
inq_1076
  DGL_INQ(dgl_inq)                    6108d  6157d  6432c
inq_11050
  DGL_INQ(dgl_inq)                    6109d  6158d  6435c
inq_11052
  DGL_INQ(dgl_inq)                    6109d  6159d  6469c
inq_12050
  DGL_INQ(dgl_inq)                    6110d  6160d  6496c
inq_13052
  DGL_INQ(dgl_inq)                    6111d  6161d  6523c
```

```
inq_250
  DGL_INQ(dgl_inq)                    6100d  6122d  6183c
inq_251
  DGL_INQ(dgl_inq)                    6100d  6123d  6189c
inq_252
  DGL_INQ(dgl_inq)                    6100d  6124d  6195c
inq_253
  DGL_INQ(dgl_inq)                    6100d  6125d  6210c
inq_254
  DGL_INQ(dgl_inq)                    6100d  6126d  6226c
inq_255
  DGL_INQ(dgl_inq)                    6101d  6127d  6232c
inq_256
  DGL_INQ(dgl_inq)                    6101d  6128d  6245c
inq_257
  DGL_INQ(dgl_inq)                    6101d  6129d  6258c
inq_258
  DGL_INQ(dgl_inq)                    6101d  6130d  6264c
inq_259
  DGL_INQ(dgl_inq)                    6101d  6131d  6271c
inq_450
  DGL_INQ(dgl_inq)                    6102d  6132d  6285c
inq_451
  DGL_INQ(dgl_inq)                    6102d  6133d  6294c
inq_buffer
  GLE_TYPES(gle_types)        ***     1270s
inq_color_table
  DGL_INQ(dgl_inq)                    6017d  6034d
inq_pip2
  GLE_GEN(gle_gen)            ***     2148c
  GLE_GENI(gle_geni)          ***     3039c
  GLE_HPGL(gle_hpgl_out)      ***     7541c
  GLE_HPGLI(gle_hpgl_in)      ***    18373c
  GLE_KNOB(gle_knob_in)       ***    18227c
  GLE_RGL(gle_ras_out)        ***     8412c
  GLE_TYPES(gle_types)                1047d  1205d
inq_pgn_table
  DGL_INQ(dgl_inq)                    6022d  6060d
inq_ws
  DGL_INQ(dgl_inq)                    6008d  6084d
inrefindex
  LIBRARIAN                           2142d  2283c  2289c  2290c  2297c  2298c  2299c  2327c  2637c  2639c  2643c  2656c
inst_eprom
  EPROMS                      ***       3d
install_ucsd_dam
  UCSD_DAM                    ***      25d
installascii
  ASCII                       ***       2d
installlifdam
  LIFDAM                      ***    1187c
  LIFDAM(lifmodule)                    29d  1180d
instate
  ETU                                 854d  1018c  1027c  1036c  1118c
instlifdam
  LIFDAM                      ***      24d
instr
  LIBRARIAN                           428d   491c   497c   566c   604c   628c   657c   712c   811c   839c   848c   992c  1004c  1024c
                                     1196c  1197c  1204c  1205c  1214c  1215c  1240c  1241c
  M68KSYS(ci)                         408d   415c   416c   417c   442c   460c   461c   491c   492c   503c   504c
```

```
internal_mous
  CTABLE                    ***   1479c
  CTABLE(brstuff)                  932d   997c
interruptlevel
  GLE_KNOB(gle_knob_in)     ***  18115d
  KEYS(keys)                        74d    85c    86c    88c   109c   114c   132c
interrupttable
  INIT(initunits)           ***   2176c
  INIT(isr)                         73c    75c    89c    90c   118c   119c
  INITLOAD(sysglobals)      ***    252d
  M68KSYS(ci)               ***    296c
intfld
  MINIT(asmr)               ***   1326d
  MINIT(xminit)                    1427c  1572c  1587c  1630c  1681c  1702c  1724c  1734c  1759c  1800c
intlevel
  A804XDVR(a804xdvr)        ***    117c
  HPHIL(hphil)                     138c   165c   187c   214c
  INIT(isr)                 ***    142c
  KEYS(keys)                        85c   109c
  PRINTER(prtdvr)                  177c   194c
intlve
  MINIT(hminit)                    254d   299d   398c   424c
  MINIT(iramigo)                   192d   221d   236c
intr_tfr
  KERNEL(iodeclarations)    ***    560d
intra_block_offset
  CS80(cs80dvr)                   1220d  1245c  1249c  1257c  1265c  1268c  1291c  1295c
intreg
  BUBBLES(bubble)                  136d   167c
intregaddr
  DC_DRV(init_dc)           ***    648c
  DI_DRV(init_discint)      ***    253c
  DMA_DRV(init_dma)                252c   256c
  G_DRV(init_gpio)          ***    268c
  H_DRV(init_hpib)          ***    276c
  INIT(isr)                         69c   100c
  INITLOAD(sysglobals)      ***    205d
  KERNEL(general_0)                1216c  1220c  1221c
  RS_DRV(init_rs)           ***    256c
intregmask
  INIT(isr)                         70c   101c
  INITLOAD(sysglobals)      ***    206d
intregvalue
  INIT(isr)                         71c   102c
  INITLOAD(sysglobals)      ***    207d
inttabletype
  INITLOAD(sysglobals)             213d   252d   275d
invalid
  LIBRARIAN                        417d   567c   621c   858c   861c   865c   869c   875c   889c   902c   905c   915c   921c   926c
                                  1005c  1029c  1076c
invalidate_stateinfo
  CS80(cs80dsr)                    602d   611d   701c
  MINIT(qminit)             ***   1154c
io_bit
  DI_DRV(extdi)                    168d   169d   170d
  G_DRV(extg)                      183d   184d   185d
  H_DRV(exth)                      183d   184d   185d
  KERNEL(general_0)                839d   844d
  KERNEL(iodeclarations)           370d   434d   436d
```

```
io_byte
  DC_DRV(init_dc)           ***    604d
  DI_DRV(init_discint)      ***    210d
  G_DRV(init_gpio)          ***    228d
  H_DRV(init_hpib)          ***    226d
  KERNEL(general_0)                755d   758d   892d   911d
  KERNEL(iodeclarations)           371d   500d   514d   528d   572d   596d   598d   657d   672d
  RS_DRV(init_rs)           ***    221d
io_carriage_rtn
  GLE_HPIB(gle_hpib_io)           10521c 10546c
  IOLIB(general_2)                 954c  1031c  1044c
  KERNEL(general_0)         ***    786c
  KERNEL(iodeclarations)    ***    247d
io_drv_ptr
  DC_DRV(init_dc)                  642c   655c
  DISCHPIB(dischpib)               224c   261c   274c   294c   317c   333c   352c   365c   396c   426c   518c   526c
  DI_DRV(init_discint)             245c   247c   283c
  GLE_HPIB(gle_hpib_io)           10067c 10076c 10112c 10517c 10538c
  G_DRV(init_gpio)                 260c   262c   295c
  H_DRV(init_hpib)                 262c   264c   306c
  IOLIB(general_1)                 277c   286c   297c   309c   323c
  IOLIB(general_2)                 857c   922c   943c   974c   999c  1057c  1078c
  IOLIB(general_4)                1337c  1450c  1475c  1521c  1549c
  IOLIB(hpib_0)                   1726c  1737c  1750c
  IOLIB(hpib_1)                    436c   501c
  IOLIB(hpib_3)             ***   2085c
  IOLIB(serial_0)                 2280c  2343c  2465c
  KERNEL(general_0)                935c   951c  1202c  1239c  1240c  1281c  1283c
  KERNEL(iodeclarations)    ***    539d
  PRINTER(prtdvr)                   96c   111c   137c   208c
  RS_DRV(init_rs)                  250c   263c   281c
  SRM_DRV(srm)                    1231c  1351c
io_error_link
  IOLIB(general_3)          ***   1173c
  KERNEL(general_0)         ***    977c
  KERNEL(iodeclarations)    ***    641d
io_error_number
  DGL_VARS(dgl_vars)        ***   1208d
io_escape
  DC_DRV(intdc)                    284c   398c   429c
  GLE_HPIB(gle_hpib_io)           10087c 10092c 10131c 10136c 10212c 10219c 10262c 10300c 10307c 10350c
  IOLIB(general_1)                 336c   341c
  IOLIB(general_4)                1307c  1364c  1374c  1381c  1394c  1399c  1408c  1413c  1499c  1514c  1564c  1573c  1591c  1600c
                                  1620c  1626c  1647c
  IOLIB(hpib_1)                    455c   460c   534c   541c   584c   622c   629c   672c   718c   768c
  IOLIB(hpib_2)                   1924c  1958c  1965c
  IOLIB(hpib_3)                   2072c  2124c  2128c
  IOLIB(serial_0)                 2238c  2250c  2270c  2301c  2313c  2333c  2367c  2408c  2456c
  IOLIB(serial_3)                 2558c  2564c  2576c  2584c  2605c  2625c  2647c  2656c  2669c  2689c  2698c  2722c  2743c  2756c
                                  2777c  2788c  2808c  2825c  2843c  2860c
  KERNEL(general_0)                811c   861c   880c   900c   919c
  KERNEL(iodeclarations)           655d   663d
  SRM_DRV(srm)                    1276c  1315c  1317c  1319c  1328c  1330c  1336c  1338c
io_find_isc
  DC_DRV(intdc)                    269c   390c   398c
  KERNEL(general_0)         ***    811c
  KERNEL(iodeclarations)           657d   672d   677c   681c
io_funny_proc
  KERNEL(iodeclarations)           463d   494d   508d   522d   584d   595d   597d
io_init_dc
  DC_DRV                    ***    723c
  DC_DRV(init_dc)                  587d   600d
```

```
io_init_discint
  DI_DRV                        ***    297c
  DI_DRV(init_discint)                 195d   206d
io_init_dma
  DMA_DRV                       ***    290c
  DMA_DRV(init_dma)                    171d   214d
io_init_gpio
  G_DRV                         ***    309c
  G_DRV(init_gpio)                     211d   224d
io_init_hpib
  H_DRV                         ***    320c
  H_DRV(init_hpib)                     211d   222d
io_init_rs
  RS_DRV                        ***    296c
  RS_DRV(init_rs)                      205d   217d
io_inq_timeout
  DGL_C_IN(dgl_confg_in)        ***  12107c
  DGL_C_OUT(dgl_confg_out)           11435c 11476c
  GLE_HPGL(gle_hpgl_out)        ***   7566c
  GLE_HPGLI(gle_hpgl_in)        ***  18339c
  GLE_TYPES(gle_types)                1064d  1214d
io_isc
  DC_DRV(init_dc)                      602d   628c   629c   647c   669c   670c   683c   697c   707c
  DC_DRV(intdc)                        410d   418c   429c   434c   473c   505c
  DI_DRV(init_discint)                 208d   239c   240c   248c   279c   280c
  GLE_HPIB(gle_hpib_io)              10190d 10198c 10200c 10212c 10213c 10214c 10215c 10219c 10227c 10229c 10236c 10243c 10252c 10259c
                                     10261c 10262c 10269c 10278d 10286c 10288c 10300c 10301c 10302c 10303c 10307c 10315c 10317c 10324c
                                     10331c 10340c 10347c 10349c 10350c 10357c 10511d 10516c 10517c 10521c 10522c 10532d 10537c 10538c
  G_DRV(init_gpio)                     226d   254c   255c   267c   291c   292c
  H_DRV(init_hpib)                     224d   255c   256c   265c   270c   302c   303c
  IOLIB(general_2)                     837d   853c   857c   858c   917d   920c   922c   923c   939d   941c   943c   944c   969d   972c
                                       974c   975c   995d   997c   999c  1000c  1014d  1016c  1017c  1019c  1027d  1029c  1030c  1031c
                                      1032c  1040d  1042c  1043c  1044c  1045c  1053d  1055c  1057c  1058c  1073d  1076c  1078c  1079c
  IOLIB(general_4)                    1357d  1360c  1361c  1363c  1364c  1391c  1394c  1399c  1401c  1402c  1405c  1408c  1413c  1415c
                                      1416c  1419c  1430c  1441d  1445c  1449c  1451c  1465d  1469c  1474c  1476c  1490d  1494c  1499c
                                      1514c  1520c  1522c  1533d  1543c  1548c  1550c
  IOLIB(hpib_1)                        512d   520c   522c   534c   535c   536c   537c   541c   549c   551c   558c   565c   574c   581c
                                       583c   584c   591c   600d   608c   610c   622c   623c   624c   625c   629c   637c   639c   646c
                                       653c   662c   669c   671c   672c   679c   698d   703c   706c   708c   715c   718c   725c   746d
                                       752c   756c   758c   765c   768c   775c
  IOLIB(hpib_2)                       1839d  1841c  1844c  1847c  1862d  1866c  1867c  1870c  1871c  1874c  1875c  1876c  1880c  1898d
                                      1902c  1903c  1904c  1908c  1910c  1918d  1920c  1921c  1924c  1927c  1928c  1936d  1938c  1941c
                                      1942c  1945c  1952d  1956c  1957c  1958c  1959c  1960c  1963c  1964c  1965c  1966c
  IOLIB(hpib_3)                       2096d  2099c  2100c  2101c  2102c  2103c
  KERNEL(general_0)                    965d   984c   987c   991c  1015c  1030c  1032c  1124c  1127c  1199c  1202c  1204c  1205c  1206c
                                      1208c  1253d  1258c  1259c  1266c  1278c  1279c
  RS_DRV(init_rs)                      219d   244c   245c   255c   276c   277c
io_line_feed
  GLE_HPIB(gle_hpib_io)              10522c 10544c 10545c
  IOLIB(general_2)                     952c   953c  1018c  1019c  1032c  1045c
  KERNEL(iodeclarations)        ***    246d
io_lvl
  DC_DRV(init_dc)                      604d   647c   651c   659c
  DI_DRV(init_discint)                 210d   248c   256c   264c
  G_DRV(init_gpio)                     228d   267c   271c   279c
  H_DRV(init_hpib)                     226d   267c   270c   279c   287c
  RS_DRV(init_rs)                      221d   255c   259c   267c
io_proc
  DISCHPIB(bkgnd)               ***     55d
  DISCHPIB(dischpib)                   201d   202d   416d   440d
  KERNEL(iodeclarations)               418d   444d   466d
io_proc_c
  KERNEL(iodeclarations)               419d   447d   454d
```

```
io_proc_1
  KERNEL(iodeclarations)               433d   456d   457d
io_proc_ptr
  KERNEL(general_0)             ***   1188c
  KERNEL(iodeclarations)               440d   453d
io_proc_s
  KERNEL(iodeclarations)               427d   451d
io_proc_type
  KERNEL(iodeclarations)        ***    374d
io_proc_vb
  KERNEL(general_0)             ***   1187c
  KERNEL(iodeclarations)               438d   452d
io_proc_vc
  KERNEL(general_0)                   1181c  1190c
  KERNEL(iodeclarations)               421d   446d   455d
io_proc_vl
  KERNEL(iodeclarations)               435d   458d
io_proc_vs
  KERNEL(general_0)             ***   1185c
  KERNEL(iodeclarations)               430d   450d
io_proc_vw
  KERNEL(general_0)             ***   1183c
  KERNEL(iodeclarations)               425d   448d
io_proc_w
  KERNEL(iodeclarations)               423d   449d
io_program_error
  AMIGO(csamigo)                ***     74d
io_read
  DGL_C_IN(dgl_confg_in)        ***  12105c
  DGL_C_OUT(dgl_confg_out)      ***  11474c
  GLE_HPGL(gle_hpgl_out)              7147c  7180c  7240c  7573c  7581c
  GLE_HPGLI(gle_hpgl_in)             18085c 18106c 18146c 18253c 18263c 18299c 18346c 18354c
  GLE_TYPES(gle_types)                1065d  1215d
io_revid
  DC_DRV(init_dc)               ***    608c
  DI_DRV(init_discint)          ***    213c
  DMA_DRV(init_dma)             ***    218c
  G_DRV(init_gpio)              ***    231c
  H_DRV(init_hpib)              ***    229c
  KERNEL(general_0)             ***    975c
  KERNEL(iodeclarations)        ***    619s
  RS_DRV(init_rs)               ***    224c
io_set_timeout
  DGL_C_IN(dgl_confg_in)        ***  12108c
  DGL_C_OUT(dgl_confg_out)           11436c 11477c
  GLE_HPGL(gle_hpgl_out)              7567c  7610c
  GLE_HPGLI(gle_hpgl_in)             18340c 18398c
  GLE_TYPES(gle_types)                1066d  1216d
io_string
  IOLIB(general_2)                     809d   818d   993d  1039d
  IOLIB(general_3)                    1107d  1111d  1112d
  IOLIB(general_4)                    1245d  1641d
  KERNEL(general_0)             ***    959d
  KERNEL(iodeclarations)               373d   377d
io_system_reset
  IOLIB(general_1)                     259c   268c
  KERNEL(general_0)                    766d  1244c  1252d
io_temp_type
  KERNEL(iodeclarations)               492d   535d
io_temp_type2
  KERNEL(general_0)             ***    969d
  KERNEL(iodeclarations)        ***    506d
```

```
io_temp_type3
  KERNEL(general_0)          ***    970d
  KERNEL(iodeclarations)     ***    520d
io_term
  DGL_C_IN(dgl_confg_in)          12052c 12106c
  DGL_C_OUT(dgl_confg_out)        11278c 11434c 11475c
  GLE_GEN(gle_gen)           ***   2174c
  GLE_GENI(gle_geni)         ***   3046c
  GLE_TYPES(gle_types)           1067d  1217d
io_tmp_ptr
  BUBBLES(bubble)                  64c    67c   161c   162c
  DC_DRV(init_dc)                 648c   652c   660c   676c   678c
  DISCHPIB(dischpib)              224c   250c   263c   276c   277c   278c   279c   282c   283c   298c   302c   303c   304c   319c
                                  320c   321c   322c   335c   336c   337c   338c   341c   342c   352c   369c   370c   375c   376c
                                  398c   399c   401c   402c   428c   466c   518c   531c   541c
  DI_DRV(init_discint)            247c   283c
  DMA_DRV(init_dma)               192c   209c
  EDRIVER(edriver)           ***   142c
  F9885(f9885dvr)            ***   264c
  GLE_HPIB(gle_hpib_io)         10068c 10077c 10100c 10113c 10122c 10124c 10202c 10206c 10208c 10232c 10290c 10294c 10296c 10320c
                                10520c 10543c
  G_DRV(init_gpio)                262c   295c
  H_DRV(init_hpib)                264c   306c
  IOLIB(general_1)                278c   287c   298c   310c   324c   349c
  IOLIB(general_2)                867c   869c   875c   877c   927c   950c   981c  1003c  1061c  1082c
  IOLIB(general_4)               1338c  1363c  1391c  1402c  1405c  1416c  1419c  1451c  1476c  1522c  1550c  1670c
  IOLIB(hpib_0)                  1727c  1738c  1751c
  IOLIB(hpib_1)                   437c   446c   448c   502c   524c   528c   530c   554c   612c   616c   618c   642c   711c   761c
  IOLIB(hpib_3)              ***  2086c
  IOLIB(serial_0)                2281c  2344c  2466c
  KERNEL(general_0)               936c   952c   990c  1017c  1040c  1048c  1054c  1061c  1065c  1080c  1086c  1093c  1100c  1108c
                                 1114c  1135c  1143c  1150c  1204c  1205c  1263c  1264c  1284c  1296c  1301c
  KERNEL(iodeclarations)     ***  540d
  PRINTER(prtdvr)                 96c   111c   113c   115c   116c   137c   208c   217c   220c   221c   225c   231c   236c   288c
  RS_DRV(init_rs)                256c   260c   268c   281c
  SRM_DRV(srm)                  1231c  1259c  1260c  1314c  1324c  1342c  1351c
io_word
  DC_DRV(extdc)                   205d   206d   207d   208d   209d   210d
  DC_DRV(init_dc)            ***  603d
  DC_DRV(intdc)                   238d   248d   249d   250d   251d   252d   253d   264d   330d   340d   348d   349d   356d   357d
                                  358d
  DI_DRV(extdi)                   159d   160d   161d   162d   163d   164d
  DMA_DRV(init_dma)               183d   200d
  G_DRV(extg)                     176d   177d   178d   179d   180d   181d
  H_DRV(exth)                     174d   175d   176d   177d   178d   179d
  IOLIB(general_1)                306d   319d
  KERNEL(general_0)               748d   749d   751d   752d   754d   757d   760d   761d   763d   764d   824d   829d   834d   853d
                                  854d   855d   872d   873d   874d   891d   910d   930d   931d   932d   945d   946d   947d
  KERNEL(iodeclarations)          372d   381d   382d   424d   426d   428d   429d   431d   432d   502d   516d   530d
  RS_DRV(init_rs)            ***  220d
  RS_DRV(rs)                      176d   177d   178d   179d   180d   181d
io_work_char
  IOLIB(general_2)                883c   970d  1018c  1052d  1061c  1062c  1074d  1082c
  IOLIB(hpib_3)                  2097d  2101c  2104c
  KERNEL(iodeclarations)     ***  605d
io_work_str
  IOLIB(general_2)                834d   862c   867c   868c   869c   870c   875c   876c   877c   878c   882c   883c   887c   918d
                                  925c   927c
io_work_word
  DMA_DRV(init_dma)               183d   188c   200d   205c
```

```
io_write
  DGL_C_IN(dgl_confg_in)     ***  12104c
  DGL_C_OUT(dgl_confg_out)        11277c 11433c 11473c
  GLE_HPGL(gle_hpgl_out)     ***   7047c
  GLE_HPGLI(gle_hpgl_in)     ***  18034c
  GLE_TYPES(gle_types)           1068d  1218d
iobuffer
  IOLIB(general_4)               1228d  1257d
iocb
  DGL_C_IN(dgl_confg_in)          12077c 12102c
  DGL_C_OUT(dgl_confg_out)        11384c 11390c 11432c 11472c
  GLE_GEN(gle_gen)                2130c  2174c
  GLE_GENI(gle_geni)              3028c  3046c
  GLE_HPGL(gle_hpgl_out)          7047c  7147c  7180c  7240c  7566c  7567c  7573c  7581c  7610c
  GLE_HPGLI(gle_hpgl_in)         18034c 18085c 18106c 18146c 18253c 18263c 18299c 18339c 18340c 18346c 18354c 18398c
  GLE_TYPES(gle_types)            1070d  1220d
iocb_ptr
  DGL_C_IN(dgl_confg_in)          12026d 12067d 12076c
  DGL_C_OUT(dgl_confg_out)        11047d 11167d 11367d 11382c 11389c
  GLE_FILE(gle_file_io)           9017d  9018d  9019d  9020d  9021d  9037d  9043d  9049d  9055c  9062d  9065c  9072d  9075c
  GLE_HPIB(gle_hpib_io)          10023d 10024d 10025d 10026d 10027d 10028d 10467d 10470c 10476d 10479c 10483d 10489c 10507d 10514c
                                 10528d 10535c 10552d
  GLE_TYPES(gle_types)            1064d  1065d  1066d  1067d  1068d  1214d  1215d  1216d  1217d  1218d
iocb_ptr_file
  DGL_C_OUT(dgl_confg_out)        11370d 11384c 11402d 11431c 11432c 11437c 11443c 11453c
iocb_ptr_hpib
  DGL_C_IN(dgl_confg_in)          12070d 12077c 12087d 12101c 12102c 12109c 12113c 12119c
  DGL_C_OUT(dgl_confg_out)        11371d 11390c 11403d 11471c 11472c 11478c 11483c 11484c 11489c
iocheck
  CONVERT(convert_text)             49d    67c    72c    76c    79c
iocode
  ETU                              119d   120c
iocomasm
  F9885(f9885dvr)            ***    37d
  GLE_HPIB(gle_hpib_io)      ***  10006d
  IOLIB(hpib_1)              ***    419d
  IOLIB(hpib_3)              ***   2053d
  IOLIB(serial_0)            ***   2226d
  IOLIB(serial_3)            ***   2524d
  KERNEL(general_0)          ***    772d
  KERNEL(iocomasm)           ***    687d
iocontrol
  IOLIB(hpib_3)              ***   2123c
  IOLIB(serial_0)                 2255c  2276c  2318c  2339c
  IOLIB(serial_3)                 2565c  2566c  2577c  2632c  2665c  2706c  2717c  2760c  2783c  2813c  2820c  2848c  2855c
  KERNEL(general_0)                762d   944d
iod_clr
  DI_DRV(init_discint)       ***    232c
  G_DRV(init_gpio)           ***    247c
  H_DRV(init_hpib)           ***    248c
  IOLIB(hpib_0)              ***   1737c
  IOLIB(serial_0)            ***   2343c
  KERNEL(general_0)          ***   1192c
  KERNEL(iodeclarations)     ***    457d
iod_end
  DISCHPIB(dischpib)               370c   399c   531c
  DI_DRV(init_discint)       ***    227c
  H_DRV(init_hpib)           ***    243c
  IOLIB(hpib_1)              ***    501c
  KERNEL(general_0)          ***   1187c
  KERNEL(iodeclarations)     ***    452d
```

```
iod_init
  DC_DRV(init_dc)                ***    614c
  DISCHPIB(dischpib)             ***    224c
  DI_DRV(init_disc.int)                 219c   283c
  GLE_HPIB(gle_hpib_io)          ***  10067c
  G_DRV(init_gpio)                      237c   295c
  H_DRV(init_hpib)                      235c   306c
  IOLIB(general_1)               ***    277c
  IOLIB(general_4)               ***   1337c
  KERNEL(general_0)                    1179c  1235c  1281c  1283c
  KERNEL(iodeclarations)         ***    444d
  PRINTER(prtdvr)                       96c   137c
  RS_DRV(init_rs)                       230c   281c
  SRM_DRV(srm)                   ***   1231c
iod_isr
  DC_DRV(init_dc)                       615c   655c
  DI_DRV(init_disc.int)                 220c   260c
  G_DRV(init_gpio)                      238c   275c
  H_DRV(init_hpib)                      236c   283c
  KERNEL(general_0)              ***   1180c
  KERNEL(iodeclarations)         ***    445d
  RS_DRV(init_rs)                       231c   263c
iod_poll
  DISCHPIB(dischpib)                    263c   283c   342c   428c
  DI_DRV(init_disc.int)          ***    230c
  H_DRV(init_hpib)               ***    246c
  IOLIB(hpib_3)                  ***   2085c
  KERNEL(general_0)              ***   1190c
  KERNEL(iodeclarations)         ***    455d
iod_rdb
  DC_DRV(init_dc)                ***    616c
  DISCHPIB(dischpib)                    369c   375c   398c   401c
  DI_DRV(init_disc.int)          ***    221c
  GLE_HPIB(gle_hpib_io)          ***  10543c
  G_DRV(init_gpio)               ***    239c
  H_DRV(init_hpib)               ***    237c
  IOLIB(general_1)               ***    286c
  IOLIB(general_2)                      867c   869c   875c   877c   950c   981c  1061c  1082c
  KERNEL(general_0)                    1181c  1233c
  KERNEL(iodeclarations)         ***    446d
  RS_DRV(init_rs)                ***    232c
iod_rds
  DC_DRV(init_dc)                ***    620c
  DI_DRV(init_disc.int)          ***    225c
  G_DRV(init_gpio)               ***    243c
  H_DRV(init_hpib)               ***    241c
  KERNEL(general_0)                     935c  1185c
  KERNEL(iodeclarations)         ***    450d
  RS_DRV(init_rs)                ***    236c
iod_rdw
  DC_DRV(init_dc)                ***    618c
  DI_DRV(init_disc.int)          ***    223c
  G_DRV(init_gpio)               ***    241c
  H_DRV(init_hpib)               ***    239c
  IOLIB(general_1)               ***    309c
  KERNEL(general_0)              ***   1183c
  KERNEL(iodeclarations)         ***    448d
  RS_DRV(init_rs)                ***    234c
```

```
iod_send
  DISCHPIB(dischpib)                    276c   278c   279c   282c   304c   321c   322c   335c   337c   338c   341c   352c   376c   402c
                                        541c
  DI_DRV(init_disc.int)          ***    229c
  GLE_HPIB(gle_hpib_io)          ***  10112c
  H_DRV(init_hpib)               ***    245c
  IOLIB(hpib_1)                  ***    436c
  KERNEL(general_0               ***   1189c
  KERNEL(iodeclarations)         ***    454d
  PRINTER(prtdvr)                       113c   115c   116c   217c   220c   221c   236c
iod_set
  DISCHPIB(dischpib)                    302c   319c
  DI_DRV(init_disc.int)          ***    231c
  G_DRV(init_gpio)               ***    246c
  H_DRV(init_hpib)               ***    247c
  IOLIB(hpib_0)                  ***   1726c
  IOLIB(serial_0)                ***   2280c
  KERNEL(general_0               ***   1191c
  KERNEL(iodeclarations)         ***    456d
iod_temp
  KERNEL(general_0                      780d   795d
  KERNEL(iodeclarations)                657d   672d   675c   680c
iod_test
  DI_DRV(init_disc.int)          ***    233c
  G_DRV(init_gpio)               ***    248c
  H_DRV(init_hpib)               ***    249c
  IOLIB(hpib_0)                  ***   1750c
  IOLIB(serial_0)                ***   2465c
  KERNEL(general_0               ***   1193c
  KERNEL(iodeclarations)         ***    458d
iod_tfr
  DC_DRV(init_dc)                ***    622c
  DISCHPIB(dischpib)             ***    518c
  DI_DRV(init_disc.int)          ***    228c
  G_DRV(init_gpio)               ***    245c
  H_DRV(init_hpib)               ***    244c
  IOLIB(general_4)                      1450c  1475c  1521c  1549c
  KERNEL(general_0               ***   1188c
  KERNEL(iodeclarations)         ***    453d
  RS_DRV(init_rs)                ***    238c
  SRM_DRV(srm)                   ***   1351c
iod_wtb
  DC_DRV(init_dc)                ***    617c
  DISCHPIB(dischpib)                    298c   303c   320c
  DI_DRV(init_disc.int)          ***    222c
  GLE_HPIB(gle_hpib_io)                10076c 10520c
  G_DRV(init_gpio)               ***    240c
  H_DRV(init_hpib)               ***    238c
  IOLIB(general_1)               ***    297c
  IOLIB(general_2)                      927c  1003c
  KERNEL(general_0                     1182c  1234c
  KERNEL(iodeclarations)         ***    447d
  PRINTER(prtdvr)                ***    231c
  RS_DRV(init_rs)                ***    233c
iod_wtc
  DC_DRV(init_dc)                ***    621c
  DI_DRV(init_disc.int)          ***    226c
  G_DRV(init_gpio)               ***    244c
  H_DRV(init_hpib)               ***    242c
  KERNEL(general_0                      951c  1186c
  KERNEL(iodeclarations)         ***    451d
  PRINTER(prtdvr)                ***    225c
  RS_DRV(init_rs)                ***    237c
```

```
iod_wtw
  DC_DRV(init_dc)              ***    619c
  DI_DRV(init_discint)         ***    224c
  G_DRV(init_gpio)             ***    242c
  H_DRV(init_hpib)             ***    240c
  IOLIB(general_1)             ***    323c
  KERNEL(general_0)            ***   1184c
  KERNEL(iodeclarations)       ***    449d
  RS_DRV(init_rs)              ***    235c
iodeclarations
  BUBBLES(bubble)              ***     26d
  DC_DRV(extdc)                ***    194d
  DC_DRV(init_dc)              ***    579d
  DC_DRV(intdc)                ***    233d
  DGL_C_OUT(dgl_confg_out)     ***  11018d
  DISCHPIB(bkgnd)              ***     31d
  DISCHPIB(dischpib)           ***    189d
  DI_DRV(extdi)                ***    151d
  DI_DRV(init_discint)         ***    187d
  DMA_DRV(init_dma)            ***    164d
  EDRIVER(edriver)             ***     24d
  F9885(f9885dvr)              ***     37d
  GLE_HPIB(gle_hpib_io)        ***  10033d
  G_DRV(extg)                  ***    168d
  G_DRV(init_gpio)             ***    203d
  H_DRV(exth)                  ***    166d
  H_DRV(init_hpib)             ***    203d
  IOLIB(general_1)             ***    231d
  IOLIB(general_2)             ***    793d
  IOLIB(general_3)             ***   1103d
  IOLIB(general_4)             ***   1192d
  IOLIB(hpib_0)                ***   1703d
  IOLIB(hpib_1)                ***    388d
  IOLIB(hpib_2)                ***   1773d
  IOLIB(hpib_3)                ***   2023d
  IOLIB(serial_0)              ***   2208d
  IOLIB(serial_3)              ***   2497d
  KERNEL(general_0)            ***    736d
  KERNEL(iocomasm)             ***    697d
  KERNEL(iodeclarations)       ***    222d
  LIB(dgl_lib)                 ***  20111d
  MIUI                         ***     11d
  PRINTER(prtdvr)              ***     36d
  RS_DRV(init_rs)              ***    197d
  RS_DRV(rs)                   ***    167d
  SRMAM(srmammodule)           ***     34d
  SRMDAM(srmdammodule)         ***     36d
  SRM_DRV(srm)                 ***     29d
ioe_bad_cnt
  IOLIB(general_3)             ***   1131c
  IOLIB(general_4)             ***   1374c
  KERNEL(iodeclarations)       ***    265d
  SRM_DRV(srm)                 ***   1317c
ioe_bad_sct
  IOLIB(general_3)             ***   1141c
  IOLIB(serial_0)                    2238c  2250c  2270c  2301c  2313c  2333c  2367c  2408c  2456c
  KERNEL(iodeclarations)       ***    275d
ioe_bad_tfr
  DC_DRV(intdc)                ***    429c
  IOLIB(general_3)             ***   1128c
  IOLIB(general_4)                   1499c  1514c
  KERNEL(iodeclarations)       ***    262d
```

```
ioe_bad_tmo
  GLE_HPIB(gle_hpib_io)              10087c 10092c
  IOLIB(general_1)                     336c   341c
  IOLIB(general_3)             ***   1132c
  KERNEL(iodeclarations)       ***    266d
ioe_buf_busy
  IOLIB(general_3)             ***   1130c
  IOLIB(general_4)                   1307c  1381c  1573c  1600c
  KERNEL(iodeclarations)       ***    264d
  SRM_DRV(srm)                       1276c  1319c
ioe_crd_dwn
  IOLIB(general_3)             ***   1142c
  KERNEL(iodeclarations)       ***    276d
ioe_dc_act
  IOLIB(general_3)             ***   1161c
  KERNEL(iodeclarations)       ***    288d
ioe_dc_car
  IOLIB(general_3)             ***   1160c
  KERNEL(iodeclarations)       ***    287d
ioe_dc_clk
  IOLIB(general_3)             ***   1158c
  KERNEL(iodeclarations)       ***    285d
ioe_dc_conf
  IOLIB(general_3)             ***   1163c
  KERNEL(iodeclarations)       ***    290d
ioe_dc_conn
  IOLIB(general_3)             ***   1162c
  KERNEL(iodeclarations)       ***    289d
ioe_dc_cts
  IOLIB(general_3)             ***   1159c
  KERNEL(iodeclarations)       ***    286d
ioe_dc_fail
  IOLIB(general_3)                   1148c  1155c
  KERNEL(iodeclarations)       ***    282d
ioe_dc_ovfl
  IOLIB(general_3)             ***   1157c
  KERNEL(iodeclarations)       ***    284d
ioe_dc_reg
  IOLIB(general_3)             ***   1164c
  KERNEL(iodeclarations)       ***    291d
ioe_dc_rval
  IOLIB(general_3)                   1149c  1165c
  KERNEL(iodeclarations)       ***    292d
ioe_dc_usart
  IOLIB(general_3)             ***   1156c
  KERNEL(iodeclarations)       ***    283d
ioe_eod_seen
  DC_DRV(intdc)                       472c   504c
  IOLIB(general_3)             ***   1143c
  KERNEL(iodeclarations)       ***    277d
ioe_isc
  DC_DRV(init_dc)              ***    697c
  DC_DRV(intdc)                       473c   505c
  DISCHPIB(dischpib)           ***    221c
  KERNEL(iodeclarations)              611s   666c
  PRINTER(prtdvr)                      92c    98c   100c   102c   139c
ioe_isc_busy
  IOLIB(general_3)             ***   1129c
  IOLIB(general_4)                   1394c  1408c
  KERNEL(iodeclarations)       ***    263d
  SRM_DRV(srm)                       1328c  1336c
```

```
ioe_misc
  DC_DRV(intdc)                 ***    398c
  GLE_HPIB(gle_hpit_io)              10212c 10300c
  IOLIB(general_3)                    1116c  1144c
  IOLIB(general_4)              ***   1620c
  IOLIB(hpib_1)                       534c   622c
  IOLIB(hpib_3)                 ***   2124c
  IOLIB(serial_3)                     2558c  2564c  2576c  2584c  2605c  2625c  2647c  2656c  2669c  2689c  2698c  2722c  2743c  2756c
                                      2777c  2788c  2808c  2825c  2843c  2860c
  KERNEL(iodeclarations)        ***    278d
ioe_no_card
  IOLIB(general_3)              ***   1122c
  KERNEL(general_0)                   861c   880c   900c   919c
  KERNEL(iodeclarations)        ***    256d
ioe_no_data
  IOLIB(general_3)              ***   1127c
  IOLIB(general_4)                    1413c  1564c  1626c
  KERNEL(iodeclarations)        ***    261d
  SRM_DRV(srm)                  ***   1338c
ioe_no_dma
  IOLIB(general_3)              ***   1134c
  KERNEL(iodeclarations)        ***    268d
ioe_no_driver
  IOLIB(general_3)              ***   1133c
  IOLIB(general_4)              ***   1364c
  KERNEL(general_0)             ***    811c
  KERNEL(iodeclarations)        ***    267d
  SRM_DRV(srm)                  ***   1315c
ioe_no_error
  IOLIB(general_3)                    1117c  1121c
  KERNEL(iodeclarations)        ***    255d
ioe_no_space
  IOLIB(general_3)              ***   1126c
  IOLIB(general_4)                    1389c  1591c  1647c
  KERNEL(iodeclarations)        ***    260d
  SRM_DRV(srm)                  ***   1330c
ioe_no_word
  IOLIB(general_3)              ***   1135c
  KERNEL(iodeclarations)        ***    269d
ioe_not_act
  IOLIB(general_3)              ***   1124c
  IOLIB(hpib_1)                       718c   768c
  IOLIB(hpib_3)                 ***   2072c
  KERNEL(iodeclarations)        ***    258d
ioe_not_dvc
  IOLIB(general_3)              ***   1125c
  IOLIB(hpib_2)                 ***   1924c
  KERNEL(iodeclarations)        ***    259d
ioe_not_hpib
  GLE_HPIB(gle_hpib_io)              10131c 10136c 10219c 10307c
  IOLIB(general_3)              ***   1123c
  IOLIB(hpib_1)                       455c   460c   541c   629c
  IOLIB(hpib_3)                 ***   2128c
  KERNEL(iodeclarations)        ***    257d
ioe_not_lstn
  IOLIB(general_3)              ***   1137c
  KERNEL(iodeclarations)        ***    271d
ioe_not_sctl
  IOLIB(general_3)              ***   1139c
  IOLIB(hpib_2)                       1958c  1965c
  KERNEL(iodeclarations)        ***    273d
ioe_not_talk
  IOLIB(general_3)              ***   1136c
  KERNEL(iodeclarations)        ***    270d
```

```
ioe_rds_wtc
  IOLIB(general_3)              ***   1140c
  KERNEL(iodeclarations)        ***    274d
ioe_result
  DC_DRV(init_dc)               ***    707c
  DC_DRV(intdc)                        472c   504c
  DISCHPIB(dischpib)            ***    222c
  IOLIB(general_3)              ***   1169c
  KERNEL(iodeclarations)               610d   667c
  LIB(dgl_lib)                  ***  21056c
  PRINTER(prtdvr)                       93c   101c   104c
  SRM_DRV(srm)                  ***   1365c
ioe_sr_fail
  IOLIB(general_3)              ***   1169c
  KERNEL(iodeclarations)        ***    293d
ioe_sr_toomany
  IOLIB(general_3)              ***   1153c
  KERNEL(iodeclarations)        ***    281d
ioe_timeout
  DISCHPIB(dischpib)            ***    222c
  GLE_HPIB(gle_hpib_io)              10262c 10350c
  IOLIB(general_3)              ***   1138c
  IOLIB(hpib_1)                        584c   672c
  KERNEL(iodeclarations)        ***    272d
  PRINTER(prtdvr)               ***    104c
  SRM_DRV(srm)                  ***   1365c
ioerror
  IOLIB(general_3)                    1107d  1111d  1116c  1117c  1119c  1148c  1149c  1151c  1173c
  LIBRARIAN                             95d  3090c  3172c
ioerror_message
  IOLIB(general_3)                    1107d  1111d  1175c
ioescapecode
  DC_DRV(init_dc)               ***    696c
  DC_DRV(intdc)                        471c   503c
  DISCHPIB(dischpib)            ***    221c
  KERNEL(iodeclarations)               253d   668c
  PRINTER(prtdvr)                       98c   102c   139c
  SRMAM(srmammodule)            ***    219c
  SRMDAM(srmdammodule)          ***   1818c
  SRM_DRV(srm)                  ***   1359c
ioinitialize
  IOLIB(general_1)                     235d   256d
iolibrary_kernel
  KERNEL                        ***    202d
iomaxisc
  BUBBLES(bubble)               ***    149c
  DC_DRV(init_dc)                      628c   669c
  DI_DRV(init_discint)                 239c   279c
  EDRIVER(edriver)              ***    131c
  GLE_HPIB(gle_hpib_io)              10196c 10284c
  G_DRV(init_gpio)                     254c   291c
  H_DRV(init_hpib)                     255c   302c
  IOLIB(general_4)              ***   1360c
  IOLIB(hpib_1)                        518c   606c   701c   750c
  IOLIB(hpib_2)                       1842c  1864c  1900c  1939c  1954c
  KERNEL(general_0)                    984c  1199c  1258c  1278c
  KERNEL(iodeclarations)               242d   368d   369d
  RS_DRV(init_rs)                      244c   276c
```

```
iominisc
  BUBBLES(bubble)              ***    149c
  DC_DRV(init_dc)                     628c   669c
  DI_DRV(init_discint)                239c   279c
  EDRIVER(edriver)             ***    131c
  G_DRV(init_gpio)                    254c   291c
  H_DRV(init_hpib)                    255c   302c
  KERNEL(general_0)                   984c  1199c  1258c  1278c
  KERNEL(iodeclarations)              241d   368d   369d
  RS_DRV(init_rs)                     244c   276c
ior
  A804XDVR(a804xdvr)           ***    165c
  DISCHPIB(bkgnd)                      65d   155d   158c
  EDRIVER(edriver)             ***    285c
  INITLOAD(loader)                   1591d  1646c  1649c
  INITLOAD(mini)                      414d   454c   455c   464c   468c   472c
ioread_byte
  DC_DRV(init_dc)                     647c   683c
  DC_DRV(intdc)                ***    290c
  DI_DRV(init_discint)         ***    248c
  G_DRV(init_gpio)             ***    267c
  H_DRV(init_hpib)             ***    270c
  KERNEL(general_0)                   753d   890d   903c  1032c  1124c  1127c
  RS_DRV(init_rs)              ***    255c
ioread_word
  DMA_DRV(init_dma)                   188c   205c   234c
  KERNEL(general_0)                   747d   852d   865c
iores
  AMIGO(amigodvr)              ***    777c
  CS80(cs80dvr)                ***   1119c
  DISCHPIB(bkgnd)                      39d   118c   133c   158c
  MINIT(cs80ir)                       761c   763c   808c   810c
  MINIT(hminit)                       309c   310c
  MINIT(qminit)                       986c   988c  1134c  1140c
  MIUI                                 23d    26c
  TAPEBKUP                            987d   990c
  TAPEBKUP(cs80tbdvr)                 736c   739c   746c   748c
  TAPEBKUP(cs80tbr)                   286c   289c
iores_value
  F9885(f9885dvr)                      94d   100c
ioresc
  CS80(cs80dvr)                      1238c  1242c  1249c
  F9885(f9885dvr)                      89c   119c   138c   156c   180c   217c   220c   224c   228c   235c   236c   237c   242c   262c
                                      280c   283c   285c   290c   293c
  INITLOAD(mini)                      311d   322d   401c   468c   472c   490c   493c   508c   511c   513c   518c   521c
  MINIT(mminit)                ***    168c
  PRINTER(prtdvr)                     105c   132c   140c   199c   290c   312c
ioresc_bkgnd
  AMIGO(amigodvr)                     470c   477c   478c   483c   516c   519c   528c   529c   533c   535c   605c   607c   609c   611c
                                      668c   671c   802c   828c   873c   874c   880c   882c   889c   891c   892c   893c   897c   899c
                                      903c   906c   910c   914c   922c   923c   926c   931c   943c
  AMIGO(csamigo)                      172c   212c
  CS80(cs80dsr)                       651c   756c   766c   923c
  CS80(cs80dvr)                      1005c  1104c  1129c
  DISCHPIB(bkgnd)                      65d   155d
  DISCHPIB(dischpib)                  222c   225c   249c   353c
  MINIT(hminit)                       325c   327c   361c   379c   395c   400c   405c   426c   449c   460c   474c   475c   480c   481c
                                      482c   492c
  MINIT(qminit)                       958c  1137c  1254c
  TAPEBKUP(cs80tbdvr)                 351c   531c   646c   656c   684c   759c   879c   889c
  TAPEBKUP(cs80tbr)            ***    294c
ioreset
  GLE_HPIB(gle_hpib_io)              10064d 10498c
  IOLIB(general_1)                     237d   274d
```

```
ioresult_value
  INITLOAD(mini)                       311d   322d   324c
iorsltwd
  CS80(cs80dsr)                        638d   639d
  DISCHPIB(bkgnd)                       39d    65d   155d
  ETU                          ***     119d
  INITLOAD(bootdammodule)      ***     594d
  INITLOAD(mini)                       311d   313d   322d   329d   335d   414d
  INITLOAD(sysglobals)         ***     178d
  LIFDAM(lifmodule)            ***      86d
  TAPEBKUP(cs80tbdvr)                  518d   519d   698d   699d
iorval
  INITLOAD(mini)                       313d   329d   377c   380c   400c   401c   454c
  MINIT(mminit)                ***     168c
iorval_to_report
  CS80(cs80dsr)                        638d   646c   751c   752c   765c   766c
  TAPEBKUP(cs80tbdvr)                  518d   526c   632c   633c   655c   656c   698d   738c   739c   754c   863c   864c   888c   889c
ios_access_to_file_not_allowed
  SRM_DRV(srm)                         959d  1162c
ios_attach_table_full
  SRM_DRV(srm)                         943d  1137c
ios_bad_file_name
  SRM_DRV(srm)                         962d  1141c
ios_bad_select_code
  SRM_DRV(srm)                         937d  1136c
ios_conflicting_share_modes
  SRM_DRV(srm)                         961d  1168c
ios_corrupt_directory
  SRM_DRV(srm)                 ***     952d
ios_deadlock_detected
  SRM_DRV(srm)                         977d  1167c
ios_device_drivers_dont_match
  SRM_DRV(srm)                 ***     941d
ios_directory_formats_dont_match
  SRM_DRV(srm)                 ***     945d
ios_directory_not_empty
  SRM_DRV(srm)                         971d  1183c
ios_ds_rom_missing
  SRM_DRV(srm)                 ***     939d
ios_duplicate_filenames
  SRM_DRV(srm)                         965d  1175c
ios_duplicate_passwords
  SRM_DRV(srm)                         976d  1160c
ios_eof_encountered
  SRM_DRV(srm)                         981d  1178c
ios_error_base
  SRM_DRV(srm)                 ***     935d
ios_error_top
  SRM_DRV(srm)                 ***     987d
ios_file_in_use
  SRM_DRV(srm)                         963d  1171c
ios_file_locked_please_retry
  SRM_DRV(srm)                         983d  1169c
ios_file_not_directory
  SRM_DRV(srm)                         970d  1182c
ios_file_not_found
  SRM_DRV(srm)                         968d  1180c
ios_file_pathname_missing
  SRM_DRV(srm)                         950d  1142c
ios_file_unopened
  SRM_DRV(srm)                         955d  1154c
ios_illegal_byte_number
  SRM_DRV(srm)                         951d  1144c
```

```
ios_improper_mass_storage_device
  SRM_DRV(srm)                    ***    944d
ios_insufficient_disk_space
  SRM_DRV(srm)                           964d  1174c
ios_invalid_file_code
  SRM_DRV(srm)                           982d  1184c
ios_invalid_file_id
  SRM_DRV(srm)                           947d  1139c
ios_invalid_file_size
  SRM_DRV(srm)                           946d  1138c
ios_invalid_ios_request
  SRM_DRV(srm)                    ***    942d
ios_invalid_protect_code
  SRM_DRV(srm)                           973d  1158c
ios_link_to_directory_not_allowed
  SRM_DRV(srm)                           978d  1165c
ios_no_capability_for_file
  SRM_DRV(srm)                           967d  1157c
ios_no_reply
  SRM_DRV(srm)                           984d  1147c
ios_password_not_allowed
  SRM_DRV(srm)                           958d  1156c
ios_password_not_found
  SRM_DRV(srm)                           975d  1159c
ios_phys_eof_encountered
  SRM_DRV(srm)                           966d  1177c
ios_purge_on_open
  SRM_DRV(srm)                           985d  1172c
ios_rename_across_volumes
  SRM_DRV(srm)                    ***    979d
ios_software_bug
  SRM_DRV(srm)                    ***    936d
ios_successful_completion
  SRM_DRV(srm)                           953d  1146c
ios_system_down
  SRM_DRV(srm)                           954d  1149c
ios_unallocated_extent
  SRM_DRV(srm)                    ***    938d
ios_unsupported_dam
  SRM_DRV(srm)                    ***    940d
ios_unsupported_directory_operation
  SRM_DRV(srm)                           960d  1164c
ios_volume_down
  SRM_DRV(srm)                           980d  1152c
ios_volume_in_use
  SRM_DRV(srm)                           969d  1181c
ios_volume_io_error
  SRM_DRV(srm)                    ***    949d
ios_volume_labels_dont_match
  SRM_DRV(srm)                    ***    957d
ios_volume_not_found
  SRM_DRV(srm)                           972d  1151c
ios_volume_offline
  SRM_DRV(srm)                           956d  1150c
ios_volume_recoverable_error
  SRM_DRV(srm)                    ***    948d
ios_volume_unrecoverable_error
  SRM_DRV(srm)                    ***    974d
```

```
iostatus
  GLE_HPIB(gle_hpib_io)          10148c 10243c 10252c 10259c 10261c 10331c 10340c 10347c 10349c
  IOLIB(hpib_1)                    472c   487c   565c   574c   581c   583c   653c   662c   669c   671c
  IOLIB(hpib_3)                   2138c  2148c  2157c  2167c
  IOLIB(serial_0)                 2253c  2274c  2316c  2337c  2411c  2459c
  IOLIB(serial_3)                 2646c  2663c  2715c  2781c
  KERNEL(general_0)                759d   929d   939c
iotemp
  SEGMENTER(asm)                   180d   213c   215c
iounintialize
  IOLIB(general_1)                 236d   265d
iounit
  MINIT(asmr)                     1331d  1335d  1342d  1346d
iowrite_byte
  KERNEL(general_0)                756d   909d
iowrite_word
  KERNEL(general_0)                750d   871d
ip
  M68KSYS(ci)                      787d   808c
  SRM_DRV(srm)                    1373d  1383c  1384c  1391c  1392c
ipart
  CLOCK(clock)                      40:    94c    95c   146c   150c   152c   156c
ipointer
  INITLOAD(sysglobals)     ***      31d
  M68KSYS(ci)                      395d   738c   740c   787d
iptr
  CRT(crt)                         546d   633c   643c   645c
  GCRT(crtb)                       416d   471c
  LIBRARIAN                       1487d  1498c  1499c  1501c  1502c  1506c  1515c  1521c  1528c  1529c  1531c  1532c  1533c
iramigo
  MINIT(hminit)            ***     248d
  MINIT(iramigo)           ***     184d
irate
  SYSDEVS(sysdevs)                 422d   424c
ires
  LIBRARIAN                         58d    93c    96c  3663c  3669c  3679c
isc
  IOLIB(general_4)                1249d  1667d  1670c
isc_busy
  IOLIB(general_4)                1249d  1667d  1672c  1673c
```

```
isc_table
  BUBBLES(bubble)                    63c    64c    67c   150c
  DC_DRV(init_dc)                   629c   670c
  DC_DRV(intdc)                     274c   293c   390c
  DISCHPIB(dischpib)                223c   232c   239c   246c   261c   292c   315c   351c   363c   394c   426c   446c   508c   526c
  DI_DRV(init_discint)              240c   280c
  DMA_DRV(init_dma)                 192c   209c   236c   264c
  EDRIVER(edriver)                  132c   147c
  F9885(f9885dvr)            ***    260c
  GLE_HPIB(gle_hpib_io)           10066c 10075c 10095c 10111c 10122c 10124c 10146c 10200c 10229c 10288c 10317c 10472c 10517c 10538c
  G_DRV(init_gpio)                  255c   292c
  H_DRV(init_hpib)                  256c   303c
  IOLIB(general_1)                  276c   285c   295c   308c   322c   344c
  IOLIB(general_2)                  857c   858c   922c   923c   943c   944c   974c   975c   999c  1000c  1057c  1058c  1078c  1079c
  IOLIB(general_4)                 1336c  1363c  1391c  1402c  1405c  1416c  1419c  1449c  1451c  1474c  1476c  1520c  1522c  1548c
                                   1550c  1670c
  IOLIB(hpib_0)                    1725c  1736c  1749c
  IOLIB(hpib_1)                     435c   446c   448c   470c   485c   500c   522c   551c   610c   639c   708c   758c
  IOLIB(hpib_3)                    2084c  2120c
  IOLIB(serial_0)                  2236c  2240c  2299c  2303c  2365c  2369c
  IOLIB(serial_3)                  2556c  2560c  2571c  2572c  2603c  2607c  2636c  2637c  2687c  2703c  2711c  2712c  2741c  2745c
                                   2765c  2766c  2806c  2810c  2841c  2845c
  KERNEL(general_0)                 857c   876c   896c   915c   934c   950c   984c  1202c  1204c  1205c  1206c  1239c  1240c  1259c
                                   1279c  1296c  1301c
  KERNEL(iodeclarations)     ***    616d
  MIUI                       ***    350c
  PRINTER(prtdvr)            ***    274c
  RS_DRV(init_rs)                   245c   277c
  SRMAM(srmammodule)         ***    149c
  SRM_DRV(srm)                     1230c  1256c  1304c
isc_table_type
  DISCHPIB(dischpib)                270d   289d   312d   329d   359d   385d   442d
  KERNEL(iodeclarations)            538d   617d
  PRINTER(prtdvr)            ***     67d
isc_te
  DISCHPIB(dischpib)                270d   274c   329d   333c
isc_te_ptr
  DISCHPIB(dischpib)                289d   292c   293c   294c   312d   315c   316c   317c   359d   363c   364c   365c   385d   394c
                                    395c   396c   442d   446c   452c   457c   464c
isize
  DGL_HPGL(dgl_hpgl)              17017d 17062d
  DGL_INQ(dgl_inq)                 6010d  6086d  6171c
  DGL_RAS(dgl_raster)             17348d 17427d
  DGL_VARS(dgl_vars)               1123d  1128d
  LIB(dgl_lib)                    20057d 20064d 20751d 20763c 20764c 20769d 20781c 20782c
isr
  A804XDVR(a804xdvr)         ***     31d
  BUBBLES(bubble)            ***     26d
  DC_DRV(init_dc)            ***    595d
  DC_DRV(intdc)              ***    259d
  DI_DRV(init_discint)       ***    201d
  DMA_DRV(init_dma)          ***    177d
  G_DRV(init_gpio)           ***    219d
  H_DRV(init_hpib)           ***    217d
  INIT(isr)                  ***     32d
  MINIT(xminit)              ***   1360d
  RS_DRV(init_rs)            ***    213d
isrchange
  INIT(isr)                         55d   136d
```

```
isrib
  A804XDVR(a804xdvr)         ***     64d
  BUBBLES(bubble)                    32d   124d   129c
  INITLOAD(sysglobals)              201d   203d
  KERNEL(iodeclarations)            493d   507d   521d   599d   600d
  MINIT(xminit)              ***   1375d
isribp
  INIT(isr)                          43d    50d    53d    56d    65d    67c    75c    83d    98c   107c   108c   112d   121c   137d
                                    144c
isribptr
  A804XDVR(a804xdvr)                 89d   115d   122c
  INIT(initunits)                   2142d  2167d
  INITLOAD(sysglobals)       ***    202d
isrinfo
  BUBBLES(bubble)                   144d   161c   170c
isrlink
  INIT(isr)                          38d    60d
  MINIT(xminit)              ***   1520c
isrmcatchall
  INIT(misc)                 ***    354c
  INITLOAD(sysglobals)       ***    195d
  SRMAM(srmammodule)                214c   220c
  SRMDAM(srmdammodule)              1812c  1819c
  SRM_DRV(srm)                      1186c  1358c
isrproctype
  INIT(isr)                          38d    45d    55d    60d    78d   136d
  INITLOAD(sysglobals)              202d   209d
  KERNEL(general_0)          ***   1180c
  KERNEL(iodeclarations)     ***    445d
isrrec
  A804XDVR(a804xdvr)                 64d   122c   386c
isrunlink
  DC_DRV(init_dc)            ***    651c
  DI_DRV(init_discint)       ***    256c
  DMA_DRV(init_dma)                 253c   257c
  G_DRV(init_gpio)           ***    271c
  H_DRV(init_hpib)           ***    279c
  INIT(isr)                          52d   111d
  MINIT(xminit)              ***   1830c
  RS_DRV(init_rs)            ***    259c
issue_cmd
  AMIGO(amigodvr)            ***    753c
  AMIGO(csamigo)                    144d   240d   300c   310c   334c   388c   401c   412c
  MINIT(iramigo)                    205c   215c   239c
issue_transfer_request
  AMIGO(amigodvr)                   545d   705c   710d
istrovfl
  INIT(misc)                 ***    332c
  INITLOAD(sysglobals)       ***    190d
  IOLIB(general_2)           ***    894c
  MOREFSYS(mfs)              ***    542c
italian_kbd
  A804XDVR(a804xdvr)         ***    187d
  NONUSKBD2                         155c   164c
  SYSDEVS(sysdevs)           ***    148d
itet
  AMIGO(amigodvr)                   447d   452d   457d   458d   459d   460d   461d   462d   463d
```

```
itfkbd
  A804XDVR(a804xdvr)                    214c   224c   341c   399c
  HPHIL(hphil)              ***         364c
  KEYS(keys)                ***         310c
  MOUSE(mouse)              ***         181c
  NONUSKBD1(non_us_kbd1)                 54c   154c
  NONUSKBD2                 ***         127c
  SYSDEVS(sysdevs)          ***         145d
itcomanyopen
  INIT(misc)                ***         348c
  INITLOAD(sysglobals)      ***         194d
  SRM_DRV(srm)              ***        1137c
iu
  CS80(cs80)                ***          95d
  M68KSYS(ci)                           856d   864c   865c
ivalue
  SYSDEVS(sysdevs)                      429d   430c
ix
  DGL_POLY(dgl_poly)                  20268d 20300c 20310c 20445d 20451c 20456c 20462c 20464c 20488c 20490c 20495c 20497c 20505c 20507c
  LIB(dgl_lib)                        20049d 20050d 20616d 20628c 20633c 20638c 20646d 20659c 20663c 20669c
iy
  DGL_POLY(dgl_poly)                  20268d 20301c 20311c 20445d 20451c 20456c 20463c 20465c 20489c 20491c 20496c 20498c 20506c 20508c
  LIB(dgl_lib)                        20049d 20050d 20616d 20629c 20634c 20638c 20646d 20660c 20664c 20669c
j
  A804XDVR(a804xdvr)                    233d   236c
  ASCII(asciimodule)                    101d   112c   114c   115c   116c
  CONVERT(convert_text)                  46d    66c    68c    90d   110c   113c   114c   115c   117c   120c
  CRT(crt)                              231d   254c   255c   551d   575c   579c   582c   584c   585c   588c   594c   596c   603c   605c
                                        607c   608c   623c   624c   714d   718c
  C_HOOK                                 79d    95c   100c
  DGL_POLY(dgl_poly)                  20411d 20775c 20776c 20779c 20780c 20784c 20786c 20787c 20790c 20792c 20793c 20795c 20801c 20804c
  DGL_RAS(dgl_raster)                 17380d 17403c 17408c
  ETU                                   852d   995c   998c   999c  1001c  1006c  1104c
  GCRT(crtb)                            152d   171c   176c   372d   377c   409c   420d   470c   471c
  GEN(dgl_gen)                         3170d  3174c  3175c  3180c
  GLE_RGL(gle_ras_out)                 8153d  8205c  8206c
  HEAPT(hpm)                ***         217d
  INIT(fs)                  ***         747d
  INITLOAD                             1737d  1747c  1751c  1753c
  INITLOAD(bootdammodule)               614d   695c   696c   697c   699c   700c
  LIBRARIAN                             109d   117c   118c   752d   787c   790c   796c   799c  1369d  3119d
  MINIT(xminit)                        1444d  1447c  1448c  1464d  1480c  1481c
  MIUI                                  377d   519c
  MOREFSYS(mfs)                         564d   600c   610c   611c   612c   615c   618c   619c   620c   621c   622c   624c   625c   626c
  SRMDAM(srmdammodule)                  467d   509d   575d
  UCSD_AM(ucsd_am)                       71d    78c    79c    80c    82c    83c    84c    87c
  UCSD_DAM(ucsdmodule)                  138d   141c   148c   149c   152c   153c   531d   535c   536c   538c   569d   581c   594c   613c
                                        617c   620c   621c
jmpstates
  LIBRARIAN                             838d   920c   925c
jptr
  LIBRARIAN                            1487d  1513c  1515c  1517c  1518c  1519c
junkint
  LIBRARIAN                             317d   354c   355c   359c   360c  1270c  1279c
k
  A804XDVR(a804xdvr)                    329d   330c   333d   334c
  CLOCK(clock)                           50d    54c    55c
  CRT(crt)                              551d   584c   585c   597c   600c   608c   [illegible]   643c   644c   645c   664d
  LIBRARIAN                             471d   473c   474c   752d   789c   790c   [illegible]
  LIFDAM(lifmodule)                     100d   104c   105c   361d   365c   368c   [illegible]   379c
  MINIT(xminit)                        1453d  1467c
  MOUSE(mouse)                           77d   114c   115c   116c   118c
  UCSD_DAM(ucsdmodule)      ***          56d
```

```
k1
  CLOCK(clock)                           50d    55c    56c    57c
k128
  EPROMS(eproms)            ***          20d
k16
  EPROMS(eproms)                         22d    58c    86c
  ETU                                    35d   959c   978c   979c   981c   999c  1013c  1098c  1101c
k2
  CLOCK(clock)                           50d    58c    59c    60c
  NONUSKBD1(non_us_kbd1)                100d   101d   108d   109d
k256
  EPROMS(eproms)                         21d   107c
kanaalphabet
  NONUSKBD1(non_us_kbd1)                105d   149c   150c
kanatocrtlookup
  CRT(crt)                               62d   390c
kanatocrtlookuptype
  CRT(crt)                               47d    62d
kappend
  KEYS(keys)                            368c   568c
  SYSDEVS(sysdevs)                      232d   531c   534c
kata
  GLE_TYPES(gle_types)      ***        1119d
  LIB(dgl_lib)                        21401c 21402c
katakana_kbd
  A804XDVR(a804xdvr)                    192d   194d
  CRT(crt)                  ***         396c
  GCRT(crtb)                ***         266c
  LIB(dgl_lib)              ***       21401c
  NONUSKBD1(non_us_kbd1)                 77d   115c   130c   143c   200c
  NONUSKBD2                 ***         200c
  SYSDEVS(sysdevs)          ***         150d
katakanatrans
  NONUSKBD1(non_us_kbd1)                 37d   145c
kbd
  C_HOOK                    ***          40d
  DGL_TOOLS(dgl_tools)      ***       20018d
kbd_crt_drivers
  KERNEL(general_0)                     742d  1231c  1232c  1239c  1240c
kbd_disable
  SYSDEVS(sysdevs)          ***         139d
kbd_enable
  SYSDEVS(sysdevs)          ***         139d
kbd_rdb
  KERNEL(general_0)                     780d  1233c
kbd_tm_name
  CTABLE(ctr)                           359d   651c
kbdaltlock
  A804XDVR(a804xdvr)        ***         401c
  KEYS(keys)                ***         263c
  NONUSKBD1(non_us_kbd1)                 49c    52c    56c    58c   200c
  NONUSKBD2                 ***         204c
  SYSDEVS(sysdevs)                      289d   650c
kbdcapslock
  A804XDVR(a804xdvr)        ***         401c
  KEYS(keys)                            216c   262c   480c
  NONUSKBD1(non_us_kbd1)    ***         200c
  NONUSKBD2                 ***         204c
  SYSDEVS(sysdevs)                      289d   650c
kbdconfig
  A804XDVR(a804xdvr)                    213c   214c   216c   218c
  SYSDEVS(sysdevs)          ***         158d
kbddata
  KEYS(keys)                            355d   360c   362c   373d   427c
```

```
kbdhooktype
  A804XDVR(a804xdvr)                    37d    38d
  HPHIL(hphil)                  ***     68d
  KEYS(keys)                    ***     55d
  SYSDEVS(sysdevs)                      37d   155d   181d   224d   287d   333d   340d
kbdio
  CRT(crt)                             415c   423c   439c
  GCRT(crtb)                           285c   293c   309c
  INIT(initunits)                     2151d  2246c
  SYSDEVS(sysdevs)                     372d   432d
kbdiohook
  KEYS(keys)                    ***    582c
  SYSDEVS(sysdevs)                     154d   434c   618c
kbdisrhook
  A804XDVR(a804xdvr)            ***    106c
  KEYS(keys)                           352c   501c   502c   580c
  MOUSE(mouse)                         111c   119c
  SYSDEVS(sysdevs)                     155d   615c
kbdlang
  A804XDVR(a804xdvr)                   223c   224c   225c
  CRT(crt)                      ***    396c
  GCRT(crtb)                    ***    266c
  LIB(dgl_lib)                  ***  21401c
  NONUSKBD1(non_us_kbd1)               115c   117c   128c   130c   138c   139c   143c   175c   181c   200c
  NONUSKBD2                            128c   137c   139c   140c   141c   142c   147c   148c   151c   164c   184c   205c   208c   211c
                                       214c   216c   218c   221c
  SYSDEVS(sysdevs)                     159d   617c
kbdmask
  A804XDVR(a804xdvr)                   172c   173c   198c   199c   385c
  HPHIL(hphil)                         234c   272s
  SYSDEVS(sysdevs)              ***    166d
  TAIL                                  87c    92c
kbdpollhook
  A804XDVR(a804xdvr)            ***    392c
  HPHIL(hphil)                  ***     89c
  SYSDEVS(sysdevs)                     156d   619c
kbdreleasehook
  KEYS(keys)                           369c   568c
  SYSDEVS(sysdevs)                     247d   646c
kbdreqhook
  A804XDVR(a804xdvr)            ***    391c
  SYSDEVS(sysdevs)                     153d   430c   614c
kbdsetup
  KEYS(keys)                           588c   589c
  SYSDEVS(sysdevs)                     371d   428d
kbdstatus
  A804XDVR(a804xdvr)                   114d   119c   120c   121c   132d   134c   135c
  KEYS(keys)                           355d   358c   373d   426c   428c   429c   430c
kbdsysmode
  A804XDVR(a804xdvr)                   355c   398c
  KEYS(keys)                           450c   462c
  SYSDEVS(sysdevs)                     289d   650c
kbdtranshook
  A804XDVR(a804xdvr)                   105c   403c
  SYSDEVS(sysdevs)                     287d   651c
kbdtype
  A804XDVR(a804xdvr)                   214c   216c   217c   222c   341c   399c
  HPHIL(hphil)                  ***    364c
  KEYS(keys)                           310c   442c
  MOUSE(mouse)                  ***    181c
  NONUSKBD1(non_us_kbd1)                54c   116c   154c   165c
  NONUSKBD2                     ***    127c
  SYSDEVS(sysdevs)                     157d   616c
```

```
kbdwaithook
  KEYS(keys)                           112c   579c
  M68KSYS(ci)                   ***    140c
  SYSDEVS(sysdevs)                     246d   645c
kbufptr
  SYSDEVS(sysdevs)                     235d   241d
kbufrec
  SYSDEVS(sysdevs)                     236d   237d
kbufrecptr
  SYSDEVS(sysdevs)                     236d   245d
kbuftype
  SYSDEVS(sysdevs)                     234d   235d
kclear
  KEYS(keys)                            87c   411c
  SYSDEVS(sysdevs)                     232d   540c
kcodes
  MOUSE(mouse)                          43d   107c
kdisplay
  DGL_C_OUT(dgl_confg_out)      ***  11069c
  KEYS(keys)                           457c   469c
  PRINTER(prtdvr)               ***    158c
  SYSDEVS(sysdevs)                     232d   545c
keep_protected_directories
  SRM_DRV                       ***   1874c
  SRM_DRV(srm)                  ***    589d
keepworkdirs
  SRMAM(srmammodule)                   147d   162c   164c   170c   171c
kernel_initialize
  KERNEL                        ***   1329c
  KERNEL(general_0)                    765d   964d
key
  A804XDVR(a804xdvr)                   326d   346c   348c   350c   352c   356c   357c   358c   359c   360c   361c   362c   363c   365c
  KEYS(keys)                           162c   164c   165c   166c   167c   168c   169c   170c   171c   172c   173c   174c   175c   176c
                                       177c   178c   193c   217c   219c   356d   360c   362c   364c   366c   368c   433c   447c   449c
                                       451c   452c   478c   479c   481c   482c   483c   485c   486c   488c   490c   493c   494c   495c
                                       508c   509c   524c   527c   528c   529c   544c   548c   553c   566c   568c
  NONUSKBD1(non_us_kbd1)        ***     69c
  SYSDEVS(sysdevs)              ***    264d
keyboard
  ETU                                   24d    29d   202c   298c
  INIT                                  30d  2516d
  LIBRARIAN                             24d    37d   101c   277c  2095c
  M68KSYS                       ***     28d
  MINIT(midecs)                 ***     47c
  TAPEBKUP                               8d   956d   979c  1504c
keyboardtype
  SYSDEVS(sysdevs)                     145d   157d
keybuffer
  CRT(crt)                      ***    707c
  GCRT(crtb)                    ***    521c
  KEYS(keys)                            83c   110c   118c   159c   181c   366c   409c   456c   467c   563c
  PRINTER(prtdvr)                      157c   181c   184c
  SYSDEVS(sysdevs)                     245d   517c   520c   524c   634c   635c
keybufferaddr
  CRT(crt)                              90d   136d   706c
  GCRT(crtb)                    ***     69d
  SYSDEVS(sysdevs)              ***     90d
keybuffersize
  CRT(crt)                              92d   138d
  GCRT(crtb)                    ***     71d
  SYSDEVS(sysdevs)              ***     91d
```

```
keybuf_ps
  DGL_C_OUT(dgl_config_out)      ***  11069c
  KEYS(keys)                          87c   125c   368c   411c   412c   457c   469c   566c   568c
  PRINTER(prtdvr)                ***   158c
  SYSDEVS(sysdevs)                    389d   512d
keyclass
  KEYS(keys)                          203c   270c   295c   296c   298c   335c   338c   339c   340c
  NONUSKBD1(non_us_kbd1)               48c   152c   194c   196c
  NONUSKBD2                           171c   177c   196c   207c   210c   213c   215c   217c   220c   222c
  SYSDEVS(sysdevs)               ***   273d
keylookup
  NONUSKBD1(non_us_kbd1)               80d   138c   139c
keys
  KEYS                           ***   595d
  KEYS(keys)                           32d   217c   272c   274c   276c   285c   289c   303c   307c   317c   324c   325c   329c   330c
  NONUSKBD1(non_us_kbd1)               69c   138c   139c   141c   149c   150c   156c   157c   158c   159c   160c   161c   162c   163c
                                     168c   169c   184c
  NONUSKBD2                          139c   140c   141c   142c   147c   148c   160c   162c   167c   187c   194c
  SYSDEVS(sysdevs)               ***   274d
keyservice
  KEYS(keys)                         373d   502c   580c
keysinit
  KEYS                           ***    30d
keystate
  MOUSE(mouse)                         62d    70c   115c
keystream
  M68KSYS(ci)                          64d   319c   330c   438c   540c   541c   542c   545c   862c  1054c
keytable
  KEYS(keys)                         201c   273c   275c   277c   278c   291c   295c   296c   304c   307c   317c   320c   321c   335c
                                     338c   339c   340c   343c
  NONUSKBD1(non_us_kbd1)              46c   138c   139c   141c   149c   150c   152c   156c   157c   158c   159c   160c   161c   162c
                                     163c   168c   169c   171c   178c   179c   180c   183c   184c   188c   189c   193c   194c   196c
  NONUSKBD2                          139c   140c   141c   142c   145c   153c   157c   158c   160c   161c   162c   166c   167c   171c
                                     175c   176c   177c   181c   182c   183c   186c   187c   193c   194c   196c   199c   207c   210c
                                     213c   215c   217c   220c   222c
  SYSDEVS(sysdevs)               ***   280d
keytrans
  A804XDVR(a804xdvr)                 326d   403c
keytranstype
  SYSDEVS(sysdevs)                   257d   288d
keytype
  SYSDEVS(sysdevs)                   258d   265d   273d
kgetchar
  KEYS(keys)                     ***   125c
  SYSDEVS(sysdevs)                   232d   526c
kgetlast
  KEYS(keys)                     ***   412c
  SYSDEVS(sysdevs)                   233d   558c
killchar
  INIT(fs)                       ***   793d
  MOREFSYS(mfs)                        67c    78c
kind
  INIT(fs)                           754d   770d
  M68KSYS(ci)                        914d   918c
kmaxbufsize
  SYSDEVS(sysdevs)                   229d   234d   639c
kmtype
  MOUSE(mouse)                         57d    59d
knob_await_locator
  DGL_KNOB(dgl_knob)               18036d 18174c
knob_device_rec
  DGL_C_IN(dgl_config_in)        ***  12023d
  GLE_KNOB(gle_knob_in)              18009d 18010d
```

```
knob_device_rec_ptr
  DGL_C_IN(dgl_config_in)            12029d 12042d
  GLE_KNOB(gle_knob_in)              18009d 18083c 18119c 18186c 18197c 18216c
knob_device_rec_space
  DGL_C_IN(dgl_config_in)            12023d 12053c
knob_dummy
  GLE_KNOB(gle_knob_in)              18075d 18225c 18226c
knob_echo_gcb
  DGL_C_OUT(dgl_config_out)          11161d 11276c 11354c
knob_get_digitize
  GLE_KNOB(gle_knob_in)              18194d 18223c
knob_get_input_p1p2
  GLE_KNOB(gle_knob_in)              18080d 18227c 18263c
knob_input_echo
  GLE_KNOB(gle_knob_in)              18207d 18224c
knob_rec
  DGL_C_IN(dgl_config_in)            12029d 12034c 12042d 12053c 12054c 12055c 12060c
knob_sample
  GLE_KNOB(gle_knob_in)              18106d 18221c
knob_sample_locator
  DGL_KNOB(dgl_knob)                 18024d 18175c
knob_start_digitize
  GLE_KNOB(gle_knob_in)              18183d 18222c
knob_type
  DGL_C_IN(dgl_config_in)        ***  12055c
  GLE_KNOB(gle_knob_in)              18011d 18084c 18229c
knonadvance
  KEYS(keys)                     ***    566c
  SYSDEVS(sysdevs)                    232d   531c
koptype
  SYSDEVS(sysdevs)                    232d   389d   512d
kpass_extc
  A804XDVR(a804xdvr)                  330c   335c
  NONUSKBD1(non_us_kbd1)         ***   127c
  SYSDEVS(sysdevs)               ***   257d
kpassthru
  A804XDVR(a804xdvr)                  343c   400c
  SYSDEVS(sysdevs)                    257d   651c
kputfirst
  SYSDEVS(sysdevs)                    233d   567c
krec
  KEYS(keys)                          254d   265c   266c   267c   268c   270c   272c   273c   274c   275c   276c   277c   284c   285c
                                      289c   291c   298c   299c   300c   303c   304c
kshift_extc
  A804XDVR(a804xdvr)             ***   368c
  KEYS(keys)                     ***   264c
  NONUSKBD2                      ***   136c
  SYSDEVS(sysdevs)               ***   257d
kunit
  INIT(fs)                       ***   760d
kvid
  CTABLE                             1232d  1234c
  INIT(fs)                       ***   760d
l
  A804XDVR(a804xdvr)                  232d   235c   236c
  DGL_POLY(dgl_poly)                20059d 20108d
  DGL_RHS(dgl_raster)            ***  17172d
  INITLOAD(loader)                    1224d  1230c  1232c  1240c
  LIFDAM(lifmodule)                     90c    93c    98d   103c   105c
  M68KSYS(ci)                          592c   599c   604c
l_flag
  MIUI                                 414d   448d
```

```
label_included_flag
  SRM_DRV                    ***   1772c
  SRM_DRV(srm)               ***    504d
ladd_cmd_buf
  AMIGO(csamigo)                    407d   412c
lag_base
  DISCHPIB(dischpib)                211d   279c   337c
lagbase
  PRINTER(prtdvr)                    57d   115c   221c
langcode
  KEYS(keys)                        259c   316c
  NONUSKBD1(non_us_kbd1)            120c   128c
  NONUSKBD2                         131c   137c
  SYSDEVS(sysdevs)           ***    278d
langcom
  KEYS(keys)                        162c   164c   165c   166c   167c   168c   169c   170c   171c   172c   173c   174c   175c   176c
                                    177c   178c   191c   193c   194c   196c   199c   201c   419c   423c   568c
  NONUSKBD1(non_us_kbd1)             42c    44c    46c
  SYSDEVS(sysdevs)           ***    284d
langcomrec
  SYSDEVS(sysdevs)                  261d   284d
langindex
  KEYS(keys)                        160c   201c   208c   209c   210c   434c   487c   555c
  NONUSKBD1(non_us_kbd1)             60c    61c    62c   199c
  NONUSKBD2                  ***    203c
  SYSDEVS(sysdevs)                  286d   649c
langitf
  A804XDVR(a804xdvr)                183d   186d
langkeyrec
  KEYS(keys)                 ***    254d
  SYSDEVS(sysdevs)                  268d   280d
langnonitf
  A804XDVR(a804xdvr)                184d   193d
langptr
  SYSDEVS(sysdevs)                  282d   285d
langrecord
  SYSDEVS(sysdevs)                  276d   282d
langtable
  KEYS(keys)                        160c   201c   203c   256c   257c   312c   313c   314c   434c   487c   555c
  NONUSKBD1(non_us_kbd1)             46c    61c   113c   120c   123c   132c   133c   166c
  NONUSKBD2                         129c   131c   133c
  SYSDEVS(sysdevs)                  285d   648c
langtype
  A804XDVR(a804xdvr)                183d   184d
  SYSDEVS(sysdevs)                  146d   159d   273d
largekbd
  A804XDVR(a804xdvr)                217c   223c
  NONUSKBD1(non_us_kbd1)            116c   165c
  SYSDEVS(sysdevs)           ***    145d
last
  INITLOAD(loader)           ***    866d
  LIBRARIAN                         198c  1522c  1582c  1625c  1836c  2384c  2388c  2396c  2413c  2793c  2903d  2905c  2906c  2908c
                                   2909c  2913c
  SRMDAM(srmdammodule)              659d   707c   743c   756c   793c   805c   811c   830c
  UCSD_DAM(ucsdmodule)              185d   195c   200c   205c   220c   228c   229c   233c   570d   586c   596c   600c   616c
last_access_date
  SRMDAM(srmdammodule)              436c   609c
  SRM_DRV(srm)               ***    283d
last_but
  GLE_KNOB(gle_knob_in)            18018d 18121c 18155c 18157c 18190c 18202c 18203c 18261c
last_char
  PRINTER(prtdvr)                   145d   234c
```

```
last_harddisc_lun
  CTABLE                            1225c  1814c  1819c
  CTABLE(options)            ***     72d
last_index
  DGL_POLY(dgl_poly)               20421d 20648c 20649c 20650c 20661c 20671c 20675c
last_knob_dx
  DGL_KNOB(dgl_knob)               18021d 18066c 18140c 18178c
last_knob_dy
  DGL_KNOB(dgl_knob)               18022d 18067c 18141c 18179c
last_knob_rx
  DGL_KNOB(dgl_knob)               18019d 18030c 18148c 18176c
last_knob_ry
  DGL_KNOB(dgl_knob)               18020d 18031c 18149c 18177c
last_lun
  CTABLE(ctr)                        308d   887d   893c   898c
last_subpoly
  DGL_POLY(dgl_poly)               20916d 20957c 20969c
last_subpolygon
  DGL_POLY(dgl_poly)               21035d 21041c 21061c 21081d 21087c 21106c 21126d 21131c 21137c 21157d 21162c 21168c
last_x
  DGL_HPGLI(dgl_hpgli)             18046d 18066c 18083c 18085c
  DGL_KNOB(dgl_knob)               18046d 18106c 18115c 18117c
  GLE_KNOB(gle_knob_in)            18020d 18159c 18200c 18259c
last_y
  DGL_HPGLI(dgl_hpgli)             18046d 18067c 18083c 18086c
  DGL_KNOB(dgl_knob)               18046d 18107c 18115c 18118c
  GLE_KNOB(gle_knob_in)            18021d 18160c 18201c 18260c
lastblock
  INIT(misc)                         537d   556c   559c   589c   625c   626c   628c   663c
lastblocks
  LIBRARIAN                         2195d  2206c  2207c  2211c  2212c
lastfkind
  INIT(misc)                 ***    703c
  INITLOAD                   ***   1707c
  INITLOAD(bootdammodule)            669c   708c
  INITLOAD(loader)           ***    837d
  INITLOAD(sysglobals)       ***     55d
  LIFDAM(lifmodule)                  145c   543c
  SRMDAM(srmdammodule)       ***     91c
lastior
  INIT(misc)                         209d   210d   300d   303c   359c   365d   389c
  M68KSYS(ci)                        394d   424c   426c
lastioresult
  INIT(ldr)                         2322d  2340c  2342c
  M68KSYS(ci)                        871d   888c   900c
lastline
  LIBRARIAN                          307d   946c   947c   950c  1201c  1252c  1423c
lastmodule
  INIT(ldr)                  ***   2334c
  INITLOAD                   ***   1721c
  INITLOAD(loader)                  1019d  1444c  1448c
  M68KSYS(ci)                ***    882c
  SEGMENTER(asm)             ***    270c
lastopening
  LIFDAM(lifmodule)                  679d   722c   728c   730c   732c   765c   767c   769c   770c   776c   787c   789c   791c   792c
                                     798c   812c   814c   816c   825c   829c   831c   834c   842c   844c   847c   858c   869c   882c
lastpatchptr
  LIBRARIAN                         2364d  2443c  2459c  2468c
lastptr
  INITLOAD(loader)                  1358d  1360c  1362c  1363c  1365c
  LIBRARIAN                         1477d  1498c  1514c  1529c  1535c  1536c  1604c  1609c  1618c  1619c  1624c  1628c  1652d  1659c
                                    1661c  1662c  1664c  1678c  1679c  1867d  1875c  1877c
lasttextrec
  LIBRARIAN                         1851d  1873c  1875c  1877c  1883c  1888c  1913c
```

```
lastused
  LIFDAM(lifmodule)                    679d  719c  774c  796c  809c  817c  855c  860c  869c  887c
lastvol
  INITLOAD(bootdammodule)      ***     564d
  LIFDAM(lifmodule)                     65d  619c  942c
latency_induced
  CS80(cs80)                   ***     186d
  CS80(cs80dsr)                ***     723c
  TAPEBKUP(cs80tbdvr)                  605c  837c
lavailblk
  UCSD_DAM(ucsdmodule)                 354d  361c  362c  365c
lb
  LIBRARIAN                            442d  502c  505c
lc
  LIBRARIAN                           3304d 3349c 3350c 3351c 3353c 3518d 3548c 3552c 3556c 3593c 3607c
lcommand
  HPHIL(hphil)                          76d  198c  298c
lcommandop
  HPHIL(hphil)                 ***     298c
  SYSDEVS(sysdevs)             ***     306d
lcount
  SRM_DRV(srm)                        1302d 1307c 1310c 1312c 1316c 1329c 1337c 1348c
lcursaddr
  CRT(crt)                             280d  284c  285c  286c  287c  288c
ldr
  CTABLE                       ***    1186d
  CTABLE(ctr)                  ***     239d
  INIT                         ***    2514d
  INIT(ldr)                    ***    2268d
  LIBRARIAN                    ***      27d
  M68KSYS(ci)                  ***      33d
  SEGMENTER(segmenter)         ***      18d
  TAIL                         ***      32d
le_configured
  HPHIL(hphil)                 ***     314c
  SYSDEVS(sysdevs)             ***     294d
le_error
  SYSDEVS(sysdevs)             ***     295d
le_loopdown
  HPHIL(hphil)                 ***     311c
  SYSDEVS(sysdevs)             ***     297d
le_timeout
  SYSDEVS(sysdevs)             ***     296d
left
  CRT(crt)                              96d  142d
  GCRT(crtb)                   ***      75d
  SYSDEVS(sysdevs)             ***      93d
leftchar
  CRT(crt)                     ***     461c
  GCRT(crtb)                   ***     332c
  INIT(misc)                   ***     163d
leftinbuf
  ETU                                  424d  432c  448c  454c  472c  481c
leftover
  INITLOAD(loader)                    1209d 1320c 1322c 1328c 1330c
lefttoxfer
  ETU                                   85d  440c  441c  443c  444c  464c 1032c 1045c 1050c 1055c 1102c 1114c
```

```
len
  CRT(crt)                             552d  573c  585c  593c  605c  607c
  CS80(cs80)                           355d  556d  587c
  GLE_KNOB(gle_knob_in)              18045d 18054c
  HEART(hpm)                           217d  220c  221c  223c  226c  230c  237c  245c  254c  255c
  INITLOAD(loader)                    1225d 1236c 1237c 1238c 1239c 1240c 1251d 1270c 1271c 1272c 1273c 1274c 1400d 1423d
                                      1455c 1456c 1457c
  LIBRARIAN                           1165d 1172c 1173c 1174c 1181c 1317d 1330c 1331c 1332c 1655d 1722d 1782c 1783c 1784c
                                      1797c 1852d 1901c 1902c 1903c 2003d 2051c 2052c 2053c 2073c 2074c 2075c 2076c
  TAPEBKUP(cs80tbr)                    194d  201c
  UNITIO(uio)                           28d   30d   53d   65c   74d   86c
len_hibyte
  SRM_DRV(srm)                 ***     303d
len_lobyte
  SRM_DRV(srm)                 ***     302d
length
  AMIGO(amigodvr)                      431d  610c  617c  626c
  CRT(crt)                             403d  415c  421c  424c  427c  437c  440c  441c  455c  457c  475c  529c  531c
  CS80(cs80)                           575d  587c
  CS80(cs80dvr)                        988d 1095d 1110c 1112c 1116c 1215d 1250c 1258c 1272c 1275c 1278c 1280c 1283c 1299c
                                      1302c 1311c 1314c 1318c 1392c 1395c
  F9885(f9885dvr)                       41d  110d  284c  286c
  GCRT(crtb)                           273d  285c  291c  294c  297c  307c  310c  311c  326c  328c  346c
  INITLOAD(bootdammodule)              588d  648d  655c  673c  674c
  INITLOAD(loader)                    1219d 1230c
  INITLOAD(mini)                       310d  385d  421c  433c  436c  441c  445c  512c
  KEYS(keys)                            73d   94s  106c  120c  121c  122c
  M68KSYS(ci)                          732d  737c  738c  739c  740c
  PRINTER(prtdvr)                       41d   48d  305c  307c  309c
  TAPEBKUP(cs80tbr)                     30d  179d  201c  223d  236c
lentitle
  SRMDAM(srmdammodule)                 651d  673c  682c  714c  721c  734c  746c  752c  759c  773c  780c  782c  791c
lerror
  HPHIL(hphil)                          48d  307c
letter
  AMIGO(amigodvr)                      428d  439d  450d  457d  458d  459d  460d  461d  462d  463d  466c  469c  470c  472c
                                       480c  484c
  AMIGO(csamigo)               ***     162c
  CS80(cs80dsr)                ***     619c
  CS80(cs80dvr)                        982d 1027d 1080c
  CTABLE                              1193d 1195d 1251c 1274c 1278c 1364c 1394c 1471c 1526c 1535c 1536c 1542c 1548c 1549c
                                      1556c 1558c 1566c 1595c 1599c 1820c
  CTABLE(brstuff)                      933d 1003c 1004c 1006c 1009c
  CTABLE(ctr)                          280d  309d  310d  312d  412c  482c  496c  525d  542c  560d  564c  595d  599c  838c
                                       883c
  CTABLE(scanstuff)            ***    1053d
  INIT(initunits)              ***    2214c
  INIT(misc)                           282c  288c
  INITLOAD                     ***    1688c
  INITLOAD(sysglobals)         ***     155d
  MIUI                                  37c  170c  319d  323c  351c
  TAPEBKUP                            1011c 1088c
letter_table
  CTABLE(brstuff)                      989d 1006c
letter_table_type
  CTABLE(brstuff)                      987d  990d
level
  SRM_DRV(srm)                 ***     304d
leveltype
  INITLOAD(sysglobals)         ***     200d
lf
  INIT(misc)                   ***     165d
  KEYS(keys)                            360c  482c
```

```
lfb
  CS80(cs80)                          89d   584c
  CS80(cs80dsr)              ***     939c
  CS80(cs80dvr)              ***    1075c
  MINIT(qminit)              ***     927c
  TAPEBKUP                          1240c  1250c  1303c  1320c  1445c
  TAPEBKUP(cs80tbdvr)                637c   639c
  TAPEBKUP(cs80tbr)                  126c   243c   248c
lfib
  ETU                                306d   315c
lfrag
  HEAPT(hpm)                         217d   230c   231c   245c
lheap
  ETU                                 82d   887c  1130c  1135c  1262c
libfound
  INIT(ldr)                         2451c  2459c
  INITLOAD(loader)                  1074d  1338c
librarian
  M68KSYS(ci)                         39d   722c  1130c
library
  M68KSYS(ci)                         39d   595c   610c   619c   621c   693c   699c   717c   728c
lif
  CTABLE(ctr)                        510c   514c
  CTABLE(options)                     60d    63d
lif_dam
  CTABLE                            1596c  1600c  1774s
  CTABLE(ctr)                        294d   517c
lif_dam_name
  CTABLE(ctr)                        351d   498c   511c   516c   518c
lif_dir_entry
  MIUI                               407d   422d   442d
lif_directory
  MIUI                               422d   481c
lif_vol_label
  MIUI                               421d   469c   471c
lif_vol_type
  MIUI                               393d   421d   429d
lifdam
  LIFDAM(lifmodule)                   28d   195d
lifmodule
  LIFDAM                     ***    1185d
  LIFDAM(lifmodule)          ***      25d
lifname
  INITLOAD(bootcammodule)            553d   559d   585d   651d
  LIFDAM(lifmodule)                   36d    60d    90d    99d   131d   160d   633d
lifnametostr
  LIFDAM(lifmodule)                  131d   183c   609c
liftofkind
  LIFDAM(lifmodule)                  540d   610c   954c
lifvol
  LIFDAM(lifmodule)                  242d   260c   281c   326c   343c   491c  1136c  1139c
lim
  LIBRARIAN                         1318d  1327c  1328c
limit
  MOUSE(mouse)                       128d   135c   159c   160c
```

```
line
  CRT(crt)                           184d   187c   194c   197c   198c
  DI_DRV(extdi)                      168d   169d   170d
  G_DRV(extg)                        183d   184d   185d
  H_DRV(exth)                        183d   184d   185d
  IOLIB(hpib_0)                     1710d  1712d  1714d  1723d  1728c  1734d  1739c  1745d  1752c
  IOLIB(serial_C)                   2216d  2218d  2220d  2232d  2242c  2262c  2282c  2295d  2305c  2325c  2345c  2358d  2371c  2419c
                                    2467c
  KERNEL(iodeclarations)             434d   436d
  LIB(dgl_lib)                     20044d 20587d 20689c 20705c
  MIUI                               223d   243c   257c   258c   262c
  TAPEBKUP                          1110d  1130c  1144c  1145c  1149c
line_line_intersection
  DGL_POLY(dgl_poly)               20266d 20746c
line_needs_restoring
  PRINTER(prtdvr)                     79d   155c   159c   174c   180c   285c
line_space
  GLE_STEXT(gle_stext)              4050c  4074c  4078c
  GLE_TYPES(gle_types)       ***    1124d
linedel
  CRT(crt)                           103d   149d
  GCRT(crtb)                 ***      82d
  SYSDEVS(sysdevs)           ***      96d
linefeed
  PRINTER(prtdvr)                     60d   302c
linenum
  LIBRARIAN                           46d   244c   245c   250c   251c   255c   257c   261c  2938c  2942c  3704c  3714c
lineops
  CRT(crt)                           502d   703c
  GCRT(crtb)                         370d   515c
linepattern
  GLE_RGL(gle_ras_out)       ***    8051d
linerange
  LIBRARIAN                          306d  1201c  1252c  1463c
lines
  ETU                               1164d  1179c  1205c  1207c  1210c
linesize
  UCSD_AM(ucsd_am)                    70d    72d    79c
linestart
  LIBRARIAN                         1371d  1375c  1376c  1377c  1395c  1404c
linestyle
  GLE_GEN(gle_gen)           ***    2210c
  GLE_HPGL(gle_hpgl_out)     ***    7534c
  GLE_RGL(gle_ras_out)       ***    8409c
  GLE_SMARK(gle_smark)              5120c  5171c
  GLE_TYPES(gle_types)       ***    1049d
linewidth
  GLE_GEN(gle_gen)           ***    2216c
  GLE_HPGL(gle_hpgl_out)     ***    7553c
  GLE_RGL(gle_ras_out)       ***    8411c
  GLE_TYPES(gle_types)       ***    1048d
linewidths
  DGL_INQ(dgl_inq)           ***    6342c
  GLE_HPGL(gle_hpgl_out)     ***    7694c
  GLE_TYPES(gle_types)       ***    1105d
  LIB(dgl_lib)               ***   20440c
```

```
link
  CTABLE(ctr)                    ***    405c
  HEAP(hpm)                             711d   122d   142c   148c   150c   156c   158c   159c   161c   167c   169c   170c   184c   185c
                                        205c   226c   236c   246c   252c   253c
  INIT(isr)                             738    94c    95c   104c   123c   124c   130c   132c
  INIT(ldr)                            2346c  2367c  2398c
  INITLOAD                       ***   1722c
  INITLOAD(loader                      1003d  1182c  1189c  1202c  1313c  1339c  1362c  1381c  1415c  1450c  1469c  1487c  1488c  1557c
  INITLOAD(sysglobals)           ***    210d
  LIBRARIAN                            1558c  1559c  1560c  1661c  1705c  1743c  1955c  2029c  2718c  2980d  3225c  3639c
  M68KSYS(ci)                          907c   933c
  SEGMENTER(asm)                       137c   270c
linker
  LIBRARIAN                      ***    24d
linkerdate
  LIBRARIAN                             39d   231c  1961c  3710c
linkerrs
  SRM_DRV(srm)                   ***   1404d
linkfiller
  SRM_DRV(srm)                          321d   329d  1426c
linkfillertype
  SRM_DRV(srm)                          295d   321d   329d
linking
  LIBRARIAN                              47d  2837c  2954c  2974c  3009c  3124c  3125c  3350c  3543c  3550c  3565c  3627c  3629c  3633c
                                        3640c  3643c  3648c  3649c  3653c  3688c  3732c
linkmodname
  INIT(ldr)                      ***   2496c
  INITLOAD(loader)                     1056d  1523c  1524c  1525c
lintlevel
  INIT(isr)                             42d    49d    52d    64d    73c    75c    82d    89c    90c    92c   108c   111d   118c   119c
                                        129c   130c
lintregaddr
  INIT(isr)                             39d    46d    61d    69c    79d   100c
lintregmask
  INIT(isr)                             40d    47d    62d    70c    80d   101c
lintregvalue
  INIT(isr)                             41d    48d    63d    71c    81d   102c
linx
  UCSD_DAM(ucsdmodule)                 325d   334c   336c   338c   342c   344c   345c   353d   357c   358c   361c   373c   398d   400c
                                        401c   405d   408c   409c   411c   421c
list
  DGL_RAS(dgl_raster)                17247d 17263c 17264c 17269c
  ETU                                  191d   204c   205c
  HEAP(hpm)                            194d   202c   203c   205c   206c
  INITLOAD(loader)                     1214d  1252d  1265c  1269c
  LIBRARIAN                             248d   327c   332c   348c   351c   352c   353c   361c   366c   368c   370c   373c   375c   377c
                                        383c  1184c  1254c  1285c  1287c  1289c  1306c  1323c  1331c  1351c  1357c  1375c  1385c  2086c
                                       2220c  2350c  2497c  2541c  2550c  2615c  2727c  2987c  2988c  2999c  3000c  3018c
listaddr
  INIT(ldr)                            2393c  2395c
  INITLOAD(loader)                     1021d  1188c  1291c  1293c  1294c  1310c  1348c
  LIBRARIAN                             400c  1354c  1356c  1764c  1767c  1790c
listdefs
  LIBRARIAN                            1315d  1465c
lister
  IOLIB(hpib_2)                        1781d  1853d
lister_constant
  GLE_HPIB(gle_hpib_io)              10215c 10303c
  IOLIB(hpib_1)                         537c   625c
  IOLIB(hpib_2)                  ***   1856c
  KERNEL(iodeclarations)         ***    317d
listener
  IOLIB(hpib_3)                        2039d  2135d  2138c
```

```
listexts
  LIBRARIAN                            1343d  1466c
listfilename
  LIBRARIAN                             43d   2934c  3527c  3725c  3726c
listing
  INIT                                  30d   2517d
  LIBRARIAN                             44d   244c   254c   261c   327c   328c   329c   330c   331c   332c   333c   334c   348c   349c
                                       351c   352c   353c   361c   366c   368c   370c   373c   375c   379c   382c   383c   472c   474c
                                      1184c  1188c  1189c  1255c  1257c  1261c  1263c  1264c  1285c  1287c  1289c  1306c  1323c  1331c
                                      1335c  1351c  1357c  1377c  1385c  2086c  2220c  2351c  2352c  2497c  2500c  2541c  2564c  2575c
                                      2615c  2727c  2729c  2730c  2939c  2940c  2941c  2987c  2988c  2999c  3001c  3002c  3003c  3018c
                                      3705c  3706c  3726c
  MIUI                           ***     8d
listinstruction
  LIBRARIAN                             414d  1308c
listlen
  LIBRARIAN                            1718d  1729c  1735c  1746c  1747c  1748c  1752c  1760c  1761c  1770c  1779c  1792c  1999d  2015c
                                       2021c  2032c  2033c  2034c  2037c  2045c  2046c  2058c  2068c  2081c
listlength
  INITLOAD(loader)                     1174d  1215d  1222d  1242c
listln
  LIBRARIAN                             260d   384c  1183c  1307c  1309c  1325c  1338c  1353c  1360c  1387c  1407c  3004c  3019c  3023c
listptr
  INITLOAD(loader)                     1221d  1229c  1243c
listsize
  INIT(ldr)                      ***   2392c
  INITLOAD(loader)                     1022d  1188c  1292c  1310c  1349c
  LIBRARIAN                             400c  1354c  1758c
listtext
  LIBRARIAN                            1365d  1464c
lite
  GEN(dgl_gen)                         3048d  3049d  3166d  3176c  3177c  3178c  3181c  3182c  3183c  3184c  3185c  3186c  3190d  3219c
                                       3221c  3223c  3227c  3228c  3229c  3231c  3237c
lkeytype
  MOUSE(mouse)                          50d    77d
lkind
  ETU                                  309d   316c   320c
llimit
  M68KSYS(ci)                          209d   224c
llo_message
  IOLIB(hpib_2)                  ***   1891c
  KERNEL(iodeclarations)         ***    305d
lmaxdevices
  SYSDEVS(sysdevs)                     299d   346d   347d
lms_dam_type
  CTABLE(options)                ***     59d
lmstr
  PRINTER(prtdvr)                       166d   186c   187c   188c
lname
  INITLOAD(bootdammodule)        ***    585d
lo_rom
  EDRIVER(edriver)                       81d   161c
  EPROMS(eproms)                         15d    80c
load
  INIT(ldr)                            2274d  2406d
  M68KSYS(ci)                    ***    936c
  TAIL                                   68c    75c    86c
load_heap_segment
  SEGMENTER(asm)                 ***    316d
  SEGMENTER(segmenter)           ***     28d
load_seg
  SEGMENTER(asm)                        175d   251c   313c   330c
```

```
load_segment
  SEGMENTER(asm)              ***     289d
  SEGMENTER(segmenter)        ***      27d
load_tapebuf
  CS80(cs80dvr)                      1168d  1267c  1282c  1293c  1316c
loadaddr
  LIBRARIAN                          2128d  2190c  2236c  2341c  2371c  2502c  2565c  2568c  2576c  2595c  2627c
loadaddr0
  LIBRARIAN                          2128d  2187c  2189c  2190c  2341c  2627c
loadandgo
  M68KSYS(ci)                         912d  1001c  1010c  1012c  1026c  1069c  1112c  1118c  1120c  1130c
loadbuffer
  ASCII(asciimodule)                   43d    59c   114c   119c   158c
loaddelta
  INITLOAD(loader)                   1506d  1516c  1518c  1536c
loader
  A804XDVR                    ***     414d
  AMIGO                       ***     980d
  ASCII                       ***     317d
  BAT                         ***      99d
  BUBBLES                     ***     179d
  CLOCK                       ***     182d
  CRT                         ***     731d
  CS80                        ***    1411d
  CTABLE(ctr)                 ***     239d
  DC_DRV                      ***     719d
  DISCHPIB                    ***     553d
  DI_DRV                      ***     293d
  DMA_DRV                     ***     286d
  EPROMS                      ***     117d
  F9885                       ***     309d
  GCRT                        ***     580d
  G_DRV                       ***     305d
  HPHIL                       ***     386d
  H_DRV                       ***     316d
  INIT                        ***    2514d
  INIT(ldr)                   ***    2270d
  INITLOAD                    ***    1664d
  INITLOAD(loader)            ***     807d
  KERNEL                      ***    1324d
  KEYS                        ***     595d
  LIBRARIAN                   ***      27d
  LIFDAM                      ***    1185d
  M68KSYS(ci)                 ***      33d
  MOUSE                       ***     203d
  NONUSKBD1                   ***     209d
  RS_DRV                      ***     292d
  SEGMENTER(segmenter)        ***      18d
  SRM_DRV(srm)                ***      31d
  TAIL                        ***      32d
  UCSD_DAM                    ***     684d
  UCSD_DAM(ucsdmodule)        ***      29d
loadfib
  INIT(ldr)                          2376c  2377c  2378c  2379c  2427c
  INITLOAD(loader)                   1054d  1141c  1145c  1154c  1164c  1263c  1288c  1325c  1329c  1342c  1387c  1390c  1391c  1392c
                                     1393c  1595c  1603c  1604c  1605c  1610c
  LIBRARIAN                          2838c  2843c  2867c  2961c  2963c  2964c  2965c  3018c  3096c  3736c
  SEGMENTER(asm)              ***     184c
loadgvr
  LIBRARIAN                           87d   165c  1176c  1293c
```

```
loadinfo
  INIT(ldr)                          2438c  2455c
  INITLOAD(loader)                   1082d  1206d  1623c
  LIBRARIAN                           394c  2748c  2845c
  SEGMENTER(asm)              ***     196c
loading
  INITLOAD(loader)                   1587d  1618c
loadproc
  SEGMENTER(asm)                      233d   251c   302d   313c   319d   330c
loadq
  INITLOAD                           1766c  1772c  1790c
  INITLOAD(loader)                   1093d  1583d
loadrom
  INIT(ldr)                          2281d  2351d
  TAIL                        ***      85c
loadtext
  INIT(ldr)                   ***    2474c
  INITLOAD(loader)                   1088d  1494d  1637c
  SEGMENTER(asm)                      239c   305c   321c
loc_dev_adr
  DGL_INQ(dgl_inq)            ***    6483c
  DGL_VARS(dgl_vars)          ***    1112d
  LIB(dgl_lib)                      21079c 21137c 21139c 21328c 21349c 21351c 21411c
loc_init
  DGL_INQ(dgl_inq)                    6233c  6246c  6471c  6525c
  DGL_VARS(dgl_vars)          ***    1242d
  GEN(dgl_gen)                ***    3094c
  LIB(dgl_lib)                      20940c 21046c 21075c 21099c 21308c 21327c 21383c
local
  IOLIB(hpib_2)                      1783d  1861d
  SEGMENTER(asm)                      230d   275c   286c
local_angle
  DGL_POLY(dgl_poly)                20876d 20889c 20898c 20899c 20928d 20992c 20993c
local_hpib_printer_default_dav
  CTABLE                      ***    1452c
  CTABLE(options)             ***     124d
local_index
  DISCHPIB(bkgnd)                      84d    88c    89c    96d    99c   100c
local_lockout
  IOLIB(hpib_2)                      1784d  1888d
local_printer_dav
  CTABLE                             1208d  1419c  1452c  1453c  1507c
local_printer_option
  CTABLE                      ***    1451c
  CTABLE(options)             ***      81d
local_printer_timeout
  CTABLE                      ***    1508c
  CTABLE(options)             ***     101d
local_printer_type
  CTABLE(options)             ***      79d
local_rs232_printer_default_dav
  CTABLE                      ***    1453c
  CTABLE(options)             ***     126d
local_spacing
  DGL_POLY(dgl_poly)                20418d 20596c 20615c 20624c 20630c 20878d 20892c 20894c 20896c 20897c 20898c 20903c
local_x_pos
  GLE_SMARK(gle_smark)               5100d  5156c  5158c
local_y_pos
  GLE_SMARK(gle_smark)               5101d  5157c  5159c
locate_and_verify
  TAPEBKUP(cs80tbdvr)         ***     679c
  TAPEBKUP(cs80tbr)                    30d   179d   205c
locateandverify
  TAPEBKUP(cs80tbr)                   195d   202c
```

```
location
  LIBRARIAN                          511d   513c   516c   519c   532c
locator_init
  LIB(dgl_lib)                     20038d 21295d
locator_limits
  GEN(dgl_gen)                      3018d  3340d  3587c  3594c
  LIB(dgl_lib)                     20920c 21044c 21084c 21361c 21363c 21422c
locator_term
  LIB(dgl_lib)                     20032d 21067d 21099c 21308c
lock
  INIT(fs)                   ***     744d
  LOCKMOD(lockmodule)                30d    37d    52c    55c
lock_on_close
  DGL_C_OUT(dgl_confg_out)         11442c 11448c
  GLE_FILE(gle_file_io)             9014d  9077c
lockable
  SRMDAM(srmdammodule)               196d   240c   294c   907d   919c   936c   937c   971d   976c   980c  1035d  1060c  1159c  1164c
lockdown
  DISCHPIB(bkgnd)            ***     134c
  ETU                                 94c    343c   378c   407c   824c   906c   928c  1140c  1269c
  F9885(f9885dvr)            ***     296c
  INIT(misc)                          216d   226d
  LIFDAM(lifmodule)                  1176c  1178c
  M68KSYS(ci)                ***    1128c
  MIUI                       ***     359c
  SEGMENTER(asm)             ***     217c
  SRMAM(srmammodule)                  223c   227c
  SRMDAM(srmdammodule)               1822c  1826c
  TAPEBKUP                           1198c  1275c
  UCSD_DAM(ucsdmodule)       ***     679c
locked_out
  IOLIB(hpib_3)                     2045d  2163d  2167c
lockedaction
  KEYS(keys)                         395c   399c   516c   521c   522c
  SYSDEVS(sysdevs)                   374d   492d
lockfile
  INITLOAD(sysglobals)       ***     138d
  LOCKMOD(lockmodule)                 49c    90c
  SRMDAM(srmdammodule)               1662c  1779c
lockfiles
  CTABLE(ctr)                ***     853c
  INIT(ldr)                         2277d  2286d
  M68KSYS(ci)                ***    1125c
locklevel
  ETU                                 93c    94c
  INIT                       ***    2542c
  INIT(misc)                          223c   230c   231c
  INITLOAD(sysglobals)       ***     250d
  KEYS(keys)                 ***     520c
  M68KSYS(ci)                ***     860c
  SYSDEVS(sysdevs)           ***     496c
lockmodule
  LOCKMOD(lockmodule)        ***      25d
lockpack
  SRMDAM(srmdammodule)       ***    1477c
  SRM_DRV                    ***    1881d
  SRM_DRV(srm)               ***    1074d
```

```
lockup
  DISCHPIB(bkgnd)            ***     109c
  ETU                                339c   368c   403c   819c   901c   925c  1134c  1261c
  F9885(f9885dvr)            ***     258c
  INIT(misc)                         215d   221d
  LIFDAM(lifmodule)          ***    1117c
  M68KSYS(ci)                ***    1123c
  MIUI                       ***     342c
  SEGMENTER(asm)             ***     211c
  SRMAM(srmammodule)         ***     185c
  SRMDAM(srmdammodule)       ***    1640c
  TAPEBKUP                           1185c  1253c
  UCSD_DAM(ucsdmodule)       ***     629c
log_aspect
  DGL_INQ(dgl_inq)           ***    6229c
  DGL_VARS(dgl_vars)         ***    1049d
  GEN(dgl_gen)                      3480c  3484c  3628c
log_bytes
  MINIT(cs80ir)                      566d   729d   757c
  MINIT(qminit)                     1106d  1187c  1197c  1201c  1203c
log_cs80_msus
  CTABLE                            1370d  1412c
log_disp_lim
  DGL_VARS(dgl_vars)         ***    1051d
  GEN(dgl_gen)                      3476c  3494c  3495c  3504c  3518c  3519c  3528c  3613c
log_entries
  MINIT(cs80ir)              ***     545d
  MINIT(qminit)              ***    1201c
log_entry
  MINIT(cs80ir)                      568d   771d   804c
log_entry_type
  MINIT(cs80ir)                      530d   547d   568d   771d
  MINIT(qminit)              ***    1108d
log_flpy_msus
  CTABLE                            1312d  1384c  1397c  1404c
log_flpy_msus_option
  CTABLE                            1312d  1323c
log_flpy_msus_options
  CTABLE                            1287d  1312d
log_harddisc_msus
  CTABLE                            1332d  1386c  1400c  1406c
log_index
  MINIT(qminit)                     1107d  1201c  1203c
log_info_type
  MINIT(cs80ir)                      542d   566d   729d
  MINIT(qminit)              ***    1106d
log_loc_lim
  DGL_VARS(dgl_vars)         ***    1057d
  GEN(dgl_gen)                      3321c  3323c  3350c  3361c  3364c  3593c
  LIB(dgl_lib)                     20277c 20278c
log_msus
  CTABLE                            1283d  1365c  1458c  1472c  1479c  1486c
log_msus_option
  CTABLE                            1283d  1410c
log_msus_options
  CTABLE                            1200d  1283d
log_specific_msus
  CTABLE                            1290d  1321c  1335c  1382c
logcode
  MINIT(cs80ir)                      563d   701d   715d   721c
logical_addr
  AMIGO(amigodvr)                    819c   917c
  AMIGO(csamigo)                     152d   405d   415c
  MINIT(hminit)                      373c   446c
```

```
logical_cyln
  MINIT(cs80ir)              ***   535d
  MINIT(qminit)                   1209c  1228c  1240c
logical_eof
  SRMDAM(srmdammodule)            606c   1487c  1488c
  SRM_DRV(srm)               ***   285d
logical_head
  MINIT(cs80ir)              ***   536d
  MINIT(qminit)                   1209c  1228c  1240c
logical_sect
  MINIT(cs80ir)              ***   537d
  MINIT(qminit)                   1209c  1228c
long
  INITLOAD(loader)           ***   927d
  LIBRARIAN                       409c   452d   609c   688c   823c   2314c  2323c  2646c  2777c
  MIUI                            455d   467c
  SRMDAM(srmdammodule)            384d   433c
long_boolean
  SRM_DRV(srm)                    205d   276d   339d   459d   460d   499d   504d   522d   571d   589d   610d   631d   636d   677d
                                  711d   733d   735d   737d   752d   754d   770d   789d   801d   807d   808d
longext
  LIBRARIAN                       449d   531c   1062c  1063c
longform
  HEAPT(hpm)                      64d    90c    96c
longoffset
  INITLOAD(loader)                925d   1462c
  LIBRARIAN                       169c   408c   1543c  1596c  1638c  1815c  1817c  1926c  2314c  2322c  2556c  2644c  2776c
longp
  HEAPT(hpm)                      211d   269c   270c
longpatch
  LIBRARIAN                       2368d  2463c  2466c  2486c
longpointer
  HEAPT(hpm)                      86d    90c    176c   299c
longsiz
  LIBRARIAN                       417d   418d   548d   716c   910c   1083c  1084c
longword
  CRT(crt)                        53d    272c   284c   678c
  GCRT(crtb)                 ***  100d
loop_condition
  TAPEBKUP                        1224d  1374c  1376c  1411c
loopcmd
  HPHIL(hphil)                    85c    197c
  SYSDEVS(sysdevs)           ***  348d
loopcmddone
  HPHIL(hphil)                    87c    90c    286c
  SYSDEVS(sysdevs)           ***  352d
loopcontrol
  HPHIL(hphil)                    80c    98c    135c   162c   190c   210c   231c   284c   306c   355c   357c   358c
  MOUSE(mouse)                    79c    153c   194c
  SYSDEVS(sysdevs)                360d   654c
loopcontrolrec
  SYSDEVS(sysdevs)                344d   360d
loopdata
  HPHIL(hphil)                    149c   150c   213c   216c   217c   287c
  SYSDEVS(sysdevs)           ***  349d
loopdevice
  HPHIL(hphil)                    84c    99c    191c   243c   251c   254c   256c   327c   340c   343c   344c   345c   349c
  MOUSE(mouse)                    80c    154c
  SYSDEVS(sysdevs)           ***  347d
loopdevices
  HPHIL(hphil)                    99c    191c   254c   327c   340c   344c   345c
  MOUSE(mouse)                    80c    154c
  SYSDEVS(sysdevs)           ***  346d
```

```
loopdevreading
  HPHIL(hphil)                    235c   318c   328c   339c   342c
  SYSDEVS(sysdevs)           ***  354d
loopdriverlist
  HPHIL(hphil)               ***  252c
  MOUSE(mouse)                    191c   192c
  SYSDEVS(sysdevs)                359d   655c
loopdriverrec
  SYSDEVS(sysdevs)                336d   337d
loopdvrop
  HPHIL(hphil)               ***  73d
  MOUSE(mouse)               ***  126d
  SYSDEVS(sysdevs)                302d   303d
loopdvrproc
  SYSDEVS(sysdevs)                303d   332d   339d
loopdvrptr
  HPHIL(hphil)               ***  227d
  MOUSE(mouse)               ***  61d
  SYSDEVS(sysdevs)                336d   341d   359d
looperror
  HPHIL(hphil)                    87c    90c    92c    245c   309c   349c
  SYSDEVS(sysdevs)           ***  350d
loopinconfig
  HPHIL(hphil)                    81c    239c   269c   317c
  SYSDEVS(sysdevs)           ***  351d
loopinit
  HPHIL                      ***  26d
loopisok
  HPHIL(hphil)                    81c    212c   217c   237c   312c   362c
  MOUSE(mouse)               ***  194c
  SYSDEVS(sysdevs)           ***  353d
loops
  MINIT(cs80ir)                   560d   567d   625d   635d   641c   770d   783d   795c
low
  M68KSYS(ci)                     745d   762c   765c
lowcod
  SEGMENTER(asm)                  72d    78c    80c    81c    90c    98c    148c   163c   294c   297c   298c
lowcode
  SEGMENTER(asm)                  75d    78c    82c
  SEGMENTER(segmenter)       ***  24d
lower3
  HEAPT(hpm)                      66d    89c    96c
lower_bound
  CTABLE(ctr)                     374d   376c
lower_byte
  CTABLE(scanstuff)          ***  1095d
lower_char
  DISCHPIB(dischpib)              389d   401c
lowerleft
  DGL_HPGL(dgl_hpgl)         ***  17263c
  DGL_VARS(dgl_vars)         ***  1079d
lowglob
  SEGMENTER(asm)                  71d    84c    86c    87c    91c    99c    207c
lowglobal
  SEGMENTER(asm)                  76d    84c    88c
  SEGMENTER(segmenter)       ***  25d
```

```
lowheap
  INIT(ldr)                          2430c 2431c 2478c 2479c
  INITLOAD(loader)                   1049d 1114c 1115c 1143c 1151c 1165c 1229c 1276c 1319c 1321c 1332c 1352c 1428c 1452c
                                     1467c 1471c 1546c 1553c 1598c 1599c 1641c 1642c
  LIBRARIAN                           127c  128c  393c 1310c 1340c 1362c 1409c 1418c 1514c 1536c 1597c 1604c 1619c 1643c
                                     1645c 1729c 1870c 1884c 1909c 1949c 1965c 2015c 2035c 2114c 2121c 2320c 2321c 2326c
                                     2331c 2398c 2405c 2437c 2513c 2523c 2525c 2642c 2747c 2798c 2821c 2825c 2840c 2842c
                                     2843c 2849c 2873c 2953c 2975c 2990c 3008c 3094c 3095c 3096c 3111c 3124c 3129c 3692c
                                     3721c 3722c
  SEGMENTER(asm)                      188c  189c  236c  242c  329c
lowheap0
  LIBRARIAN                            61d 3692c 3721c
lowid
  HPHIL(hphil)                  ***   257c
  MOUSE(mouse)                  ***   186c
  SYSDEVS(sysdevs)              ***   338d
lowkeys
  NONE(KBD2                            39d  139c  140c  141c  142c
lowrange
  LIBRARIAN                           307d 1200c 1201c 1251c 1252c 1426c 1433c
lpctritype
  HPHIL(hphil)                         55d  132d  159d
lprompt1
  M68KSYS(ci)                         977d 1090c
lprompt2
  M68KSYS(ci)                         979d 1090c
lptr
  CRT(crt)                            158d  509c  513c  514c  519c  520c  524c  530c  532c  537c  706c
lr
  CTABLE(ctr)                         333d  334d  335d  720d  727c  729c  733c  737d  745c  746c  747c  752c  791d  793c
lrec
  INITLOAD(bootdammodule)             732d  738d
ls_map
  DGL_HPGL(dgl_hpgl)                17191d 17196c
  DGL_RAS(dgl_raster)               17052d 17057c
ls_map_def
  DGL_HPGL(dgl_hpgl)                17189d 17191d
  DGL_RAS(dgl_raster)               17050d 17052d
ls_patterns
  DGL_VARS(dgl_vars)                 1303d 1307d
lsize
  LIBRARIAN                           710a  715c  716c  717c  721c  722c  726c
lt
  LIFDAM(lifmodule)                   540d  543c  544c
ltime
  CLOCK(clock)                         51d   53c   54c  101d  103c  104c  111c  115d  117c  118c  123c
  M68KSYS(ci)                         151d  158c  159c  164c
  SYSDEVS(sysdevs)                    458d  460c
ltitle
  M68KSYS(ci)                         593d  598c  599c  602c  625d  633c  634c  636c  639c  640c  642c
lun
  CS80(cs80dsr)                       613d  616c  618c
  CTABLE                             1213d 1224c 1225c 1268d 1274c 1275c 1278c 1522c 1531c 1540c 1554c 1574c
  CTABLE(ctr)                         890d  896c  898c  900c  902c  903c  905c  907c
  MIUI                                 18d  142c  148c  151c  153c  167c  206c  237c  353c  477c  487c  514c
  TAPEBKUP                           1048d 1072c 1078c 1081c 1083c 1085c 1095c 1106d 1121c 1124c 1426d 1433c 1434c 1436c
                                     1460d 1467c 1468c 1472c
lun1
  CTABLE                             1213d 1816c 1819c 1820c 1822c
  CTABLE(ctr)                         307d  875d  880c
  TAPEBKUP                           1003d 1007c
```

```
lun2
  CTABLE                             1213d 1813c 1814c 1816c 1818c 1819c 1822c
  CTABLE(ctr)                         307d  875d  879c
  TAPEBKUP                           1003d 1008c
lun11
  CTABLE(ctr)                         303d  306d  823d  827c  828c  832d  837c  838c  840c  841c  842c  850d  854c  855c
                                      868d  870c  871c
  M68KSYS(ci)                         588d  612c  613c  615c  617c  953d  957c  958c  962c
lv
  TAPEBKUP(cs80tbr)                   189d  198c  199c  200c  201c  202c  203c
lvheader
  LIFDAM(lifmodule)                    42d  197d  251c  352c  528c 1144c
lx
  GEN(dgl_gen)                       3032d 3169d 3176c 3181c 3182c 3183c 3184c 3185c 3186c 3281d 3293c 3296c 3313d 3321c
lxmax
  GEN(dgl_gen)                       3018d 3340d 3353c
  LIB(dgl_lib)                      20083d 20880d 20902c 20908c
lxmin
  GEN(dgl_gen)                       3018d 3340d 3352c
  LIB(dgl_lib)                      20083d 20880d 20902c 20907c
ly
  GEN(dgl_gen)                       3032d 3169d 3177c 3182c 3184c 3186c 3281d 3295c 3297c 3313d 3323c
lymax
  GEN(dgl_gen)                       3019d 3341d 3355c
  LIB(dgl_lib)                      20083d 20880d 20902c 20910c
lymin
  GEN(dgl_gen)                       3019d 3341d 3354c
  LIB(dgl_lib)                      20083d 20880d 20902c 20909c
lz
  GEN(dgl_gen)                       3169d 3178c 3181c 3183c 3185c
m
  MIUI                                 15d  103d  171c  197c  330c  348c  473c
m9816a
  DGL_TOOLS(dgl_tools)          ***  20067c
  GLE_RGL(gle_ras_out)          ***   8431c
  GLE_TYPES(gle_types)          ***   1254d
m9826a
  DGL_TOOLS(dgl_tools)          ***  20063c
  GLE_RGL(gle_ras_out)          ***   8438c
  GLE_TYPES(gle_types)          ***   1255d
m9836a
  DGL_TOOLS(dgl_tools)               20051c 20057c
  GLE_RGL(gle_ras_out)          ***   8445c
  GLE_TYPES(gle_types)          ***   1256d
m9836c
  DGL_C_OUT(dgl_confg_out)           11218c 11238c
  DGL_TOOLS(dgl_tools)          ***  20048c
  GLE_RGL(gle_ras_out)          ***   8452c
  GLE_TYPES(gle_types)          ***   1257d
m9837a
  DGL_C_OUT(dgl_confg_out)      ***  11255c
  GLE_RGL(gle_ras_out)               8383c  8467c
  GLE_TYPES(gle_types)          ***   1259d
m98627a
  DGL_C_OUT(dgl_confg_out)      ***  11321c
  GLE_RGL(gle_ras_out)          ***   8459c
  GLE_TYPES(gle_types)          ***   1258d
m_none
  KEYS(keys)                           45d   49d   50d  455c  465c  467c  469c
  PRINTER(prtdvr)                     181c  183c
  SYSDEVS(sysdevs)                    151d  620c
m_sysnorm
  KEYS(keys)                           48d   50d  453c  470c
  SYSDEVS(sysdevs)              ***   151d
```

```
marker_size_x
  DGL_INQ(dgl_inq)              ***    6191c
  DGL_VARS(dgl_vars)            ***    1120d
  LIB(dgl_lib)                        21181c 21182c 21183c
marker_size_y
  DGL_INQ(dgl_inq)              ***    6192c
  DGL_VARS(dgl_vars)            ***    1121d
  LIB(dgl_lib)                        21182c 21184c
marker_type
  GLE_SMARK(gle_smark)                 5123c  5135c  5153c  5182c
  GLE_TYPES(gle_types)          ***    1115d
marker_width
  GLE_SMARK(gle_smark)                 5129c  5136c  5158c  5190c
  GLE_TYPES(gle_types)          ***    1116d
markers
  GLE_SMARK(gle_smark)                 5032d  5154c
markuser
  A804XDVR                      ***     416c
  AMIGO                         ***     983c
  ASCII                         ***     322c
  BAT                           ***     103c
  BUBBLES                       ***     183c
  CLOCK                         ***     186c
  CRT                           ***     734c
  CS80                          ***    1415c
  DC_DRV                        ***     724c
  DISCHPIB                      ***     557c
  DI_DRV                        ***     298c
  DMA_DRV                       ***     291c
  EPROMS                        ***     119c
  F9885                         ***     312c
  GCRT                          ***     583c
  G_DRV                         ***     310c
  HPHIL                         ***     389c
  H_DRV                         ***     321c
  INIT                          ***    2576c
  INITLOAD(loader)                     1079d  1124d  1658c
  KERNEL                        ***    1330c
  KEYS                          ***     598c
  LIFDRM                        ***    1188c
  M68KSYS(ci)                           831c   937c   948c  1175c
  MOUSE                         ***     206c
  NONUSKBD1                     ***     212c
  RS_DRV                        ***     297c
  SRM_DRV(srm)                  ***    1218c
  TAIL                                  40c    70c
  UCSD_DAM                      ***     688c
mask
  A804XDVR(a804xdvr                     69d   165c   166c   384c
  AMIGO(csamigo)                       279d   286d   287d
  DGL_RAS(dgl_raste )                17306d 17333c
  IOLIB(hpib_2)                      1790d  1917d  1928c
  LIBRARIAN                           455d   790c   799c
masked
  CS80(cs80dsr)                        792d   852d   853d   854d   855d   856d   857d   858d   859d   860d   861d   862d   863d   864d
maskops
  A804XDVR(a804xdvr                    163d   172c   173c   198c   199c   385c   404c
maskopshook
  A804XDVR(a804xdvr             ***    404c
  HPHIL(hphil)                         234c   272s
  MINIT(mminit)                        154c   163c   167c
  SYSDEVS(sysdevs)                     168d   600c
  TAIL                                  87c    92c
```

```
match
  A804XDVR(a804xdvr                    279c   280c   307c   308c   309c
  SYSDEVS(sysdevs)              ***    218d
match1
  INITLOAD(loader)                    1178d  1201c  1350c
match2
  INITLOAD(loader)                    1300d  1335c
matchdefext
  INITLOAD(loader)                    1169d  1186c  1308c  1346c
matched
  INITLOAD(loader)                    1170d  1179d  1186c  1208d  1308c  1334c  1336c  1346c
matchfile
  INIT(ldr)                     ***   2439c
  INITLOAD(loader)                    1086d  1195d  1624c
  SEGMENTER(asm)                ***    197c
matchflag
  INITLOAD(loader)              ***   1171d
matcht
  A804XDVR(a804xdvr)                   278c   304c
  SYSDEVS(sysdevs)              ***    213d
max
  DGL_POLY(dgl_poly)            ***  20480c
  GEN(dgl_gen)                        3016d  3106d  3112c  3113c
  MINIT(hminit)                        263d   274d
  MINIT(midecs)                 ***     22d
  MINIT(mminit)                 ***    125d
  MINIT(qminit)                        912c   913c
  MIUI                                 295c   300c   301c   313c
  SRM_DRV(srm)                        1011d  1019d  1553d  1565c  1591d  1603c
max3
  GEN(dgl_gen)                        3195d  3198c  3200c  3202c  3219c
max_bt
  MINIT(hminit)                        335d   352d   366c   379c   384c
max_buffer
  GLE_HPGL(gle_hpgl_out)              7009d  7018d  7522c
  GLE_HPGLI(gle_hpgl_in)             18009d 18018d 18335c
max_disp_lim
  DGL_INQ(dgl_inq)              ***   6212c
  DGL_VARS(dgl_vars)            ***   1053d
  LIB(dgl_lib)                       20183c 20256c 20507c 21123c 21124c 21125c 21126c
max_file_size
  SRM_DRV(srm)                         281d   501d   662d  1769c
max_loc_lim
  DGL_INQ(dgl_inq)              ***   6247c
  DGL_VARS(dgl_vars)            ***   1059d
  LIB(dgl_lib)                       20281c 20282c 20904c 21330c
max_num_files
  SRM_DRV(srm)                         373d  1565c
max_num_passwords
  SRM_DRV(srm)                         396d  1603c
max_record_size
  SRMDAM(srmdammodule)          ***    943c
  SRM_DRV(srm)                         280d   497d   661d  1765c
max_sparing_tries
  MINIT(qminit)                       1115d  1222c
max_tfr_length
  AMIGO(amigodvr)                      716d   727c   736c   740c   741c   745c   748c   750c
max_vols
  CTABLE(options)                      170d   173d   175d   177d   179d   181d   183d   185d   187d
max_vp
  MINIT(hminit)                        419d   458c
maxbuf
  ETU                                  420d   432c
```

```
maxbytes
  BUBBLES(bubble)              ***    111c
maxcadd
  CS80(cs80)                   ***    112d
  CS80(cs80dvr)                ***   1073c
  MINIT(qminit)                ***    924c
maxcountx
  SYSDEVS(sysdevs)             ***    321d
maxcounty
  SYSDEVS(sysdevs)             ***    322d
maxcountz
  SYSDEVS(sysdevs)             ***    323d
maxdata
  A804XDVR(a804xdvr)                   65d    82c   128c   148c   387c
maxhadd
  CS80(cs80)                   ***    113d
  CS80(cs80dvr)                ***   1073c
  MINIT(qminit)                        923c  1192c
maxhead
  MINIT(xminit)                       1403d  1528c  1531c  1533c  1535c  1556c
maximum
  GLE_FILE(gle_file_io)        ***   9032d
  GLE_HPGL(gle_hpgl_out)              7016d  7061c  7522c
  GLE_HPGLI(gle_hpgl_in)             18016d 18335c
  GLE_HPIB(gle_hpib_io)              10043d 10544c
maximum_polygon_vertices
  DGL_HPGL(dgl_hpgl)           ***  17267c
  DGL_INQ(dgl_inq)             ***   6390c
  DGL_RAS(dgl_raster)          ***  17600c
  DGL_VARS(dgl_vars)           ***   1102d
maximum_term_count
  DISCHPIB(dischpib)                   506d   511c   512c   513c   515c
maxkana
  CRT(crt)                              41d    47d   389c
maxlevel
  INITLOAD(sysglobals)                  24d   200d
maxlines
  MIUI                                 227d   258c   261c
  TAPEBKUP                            1114d  1145c  1148c
maxmodules
  LIBRARIAN                             75d  2852c  2886c  3181c  3183c  3185c  3188c  3189c  3190c  3562c  3731c
maxparm
  MINIT(qminit)                       1034d  1047c
maxptr
  HEAPT(hpm)                            54d    61d
maxrealisc
  DC_DRV(init_dc)              ***    703c
  DC_DRV(intdc)                ***    238d
  DGL_C_OUT(dgl_confg_out)     ***  11313c
  GLE_HPIB(gle_hpib_io)        ***  10496c
  KERNEL(iodeclarations)       ***    244d
maxrec
  SRMDAM(srmdammodule)                1029d  1050c  1051c  1052c  1055c  1088c  1125c  1136c  1148c
  SRM_DRV(srm)                        1051d  1738d  1765c
maxrefsize
  INITLOAD(loader)             ***   1496d
  LIBRARIAN                           2280d  2284c
maxromex
  CRT(crt)                             357d   358d
maxsadd
  CS80(cs80)                   ***    114d
  CS80(cs80dvr)                ***   1076c
  MINIT(qminit)                        925c   926c
```

```
maxsc
  ETU                                   37d   700c   779c
  INITLOAD(sysglobals)                  22d   199d
maxsize
  CRT(crt)                     ***    707c
  GCRT(crtb)                   ***    521c
  KEYS(keys)                   ***    563c
  SYSDEVS(sysdevs)                      240d   517c   520c   639c
maxspare
  MINIT(xminit)                       1403d  1525c  1528c  1531c  1543c  1570c  1809c
maxsvadd
  CS80(cs80)                   ***    115d
  CS80(cs80dsr)                ***    939c
  CS80(cs80dvr)                ***   1075c
  MINIT(qminit)                ***    927c
  TAPEBKUP                            1250c  1303c  1320c  1445c
maxtimes
  MINIT(xminit)                       1381d  1567c
maxtries
  AMIGO(amigodvr)                      788d   871c
  F9885(f9885dvr)                       69d   233c
maxunit
  CS80(cs80dsr)                ***    616c
  CTABLE(ctr)                          815c   827c   854c
  GLE_KNOB(gle_knob_in)        ***  18035c
  INIT(initunits)              ***   2239c
  INITLOAD(sysglobals)                  16d    40d
  M68KSYS(ci)                          613c   797c   846c   856d   864c   957c
  MIUI                                 233c   512c
  SRMAM(srmammodule)           ***    166c
  TAPEBKUP                     ***   1120c
  UNITIO(uio)                  ***     42c
maxx
  CRT(crt)                             163d   300c   323c   334c   344c   461c   463c   486c   513c   514c   519c   524c   531c   537c
                                       671c   707c
  GCRT(crtb)                           110d   209c   230c   231c   332c   334c   354c   386c   395c   402c   406c   505c   521c
maxy
  CRT(crt)                             163d   303c   324c   335c   449c   463c   468c   488c   672c
  GCRT(crtb)                           111d   212c   236c   319c   334c   339c   356c   506c
mb
  CTABLE(ctr)                          334d   335d   336d   720d   732c   733c   737d   751c   752c   756d   768c
mbptr
  INITLOAD(bootdammodule)              734d   743c
mbsize
  INITLOAD(bootdammodule)              735d   743c
mdataproc
  MOUSE(mouse)                          75d   190c
mdp
  INITLOAD(loader)                     905d  1319c  1352c  1434c  1445c  1451c  1485c
  LIBRARIAN                           3222c  3225c
media
  INITLOAD(bootdammodule)      ***    592d
media_origin
  SRM_DRV(srm)                 ***    347d
media_wear
  CS80(cs80)                   ***    185d
  CS80(cs80dsr)                ***    722c
  TAPEBKUP(cs80tbdvr)                  604c   836c
mediavalid
  LIFDAM(lifmodule)                    199d   214c   215c   270c   380c   972c   983c  1091c  1099c  1118c  1160c
medium_copy
  TAPEBKUP                            1214d  1509c
```

```
medium_parameters
  CTABLE                                1535c  1548c  1658s  1668s  1678s  1689s  1707s  1723s  1734s
  CTABLE(ctr)                            309d   525d   543c   544c   545c   546c   547c   548c   549c   550c   551c   552c   553c   554c
                                         555c   599c
  CTABLE(scanstuff)              ***    1144c
medium_size
  CTABLE(ctr)                            722d   729c   730c   732c   739d   746c   748c   749c   751c
memavail
  ETU                            ***     938c
  LIBRARIAN                      ***    3722c
  LIFDAM(lifmodule)              ***     399c
  SEGMENTER(asm)                 ***     186c
  UCSD_DAM(ucsdmodule)           ***     582c
memblock
  HEAPT(hpm)                             68d    69d    78d   122d
membytes
  M68KSYS(ci)                           785d   806c   807c   808c
memtoeprom
  ETU                                   603d  1088c
menu
  LIBRARIAN                            3517d  3620c
menu1
  KEYS(keys)                             44d    45d    48d    49d    50d
menu2
  KEYS(keys)                             45d    47d
menustate
  KEYS(keys)                            453c   455c   465c   466c   467c   468c
  PRINTER(prtdvr)                       181c   183c
  SYSDEVS(sysdevs)                      162d   620c
menutype
  KEYS(keys)                     ***     44d
  SYSDEVS(sysdevs)                      151d   162d
merged
  LIBRARIAN                            1866d  1877c  1879c  1881c
mergedefs
  LIBRARIAN                            1995d  2993c
mergeexts
  LIBRARIAN                            1713d  2991c
mergetext
  LIBRARIAN                            1865d  1934c  1951c
merging
  LIBRARIAN                            2132d  2171c  2201c  2202c  2533c
mesasurement
  CTABLE(scanstuff)              ***    1088d
message
  MIUI                                   23d    25c    26c    27c
  TAPEBKUP                              987d   989c   990c   991c   996d   998c
  TAPEBKUP(cs80tbdvr)                   715d   746c   747c
message_length
  CS80(cs80)                     ***     143d
  SRM_DRV(srm)                          322d   330d  1442c  1484c
message_sequence
  CS80(cs80)                     ***     141d
mfclose
  INITLOAD(bootdammodule)               684d   723c
mfopen
  INITLOAD(bootdammodule)               648d   655c   659c   678c   705c
mfs
  MOREFSYS                       ***     680d
  MOREFSYS(mfs)                  ***       5d
mgator_c
  GLE_TYPES(gle_types)           ***    1260d
```

```
mgbase
  INITLOAD(loader)                     1399d  1405c  1412c  1413c
  LIBRARIAN                            1653d  1669c  1685c  1686c  1708c
mi_controller
  AMIGO(amigodvr)                       513c   530c   678c   734c   822c   898c   956c
  AMIGO(csamigo)                        138d   177d   179c   311c   413c
microsec_value
  DISCHPIB(dischpib)             ***     216d
middlebytes
  INIT(misc)                            538d   586c   600c   601c   605c   606c   640c   654c   667c
midecs
  MINIT(bminit)                  ***      57d
  MINIT(hminit)                  ***     248d
  MINIT(midecs)                  ***      10d
  MINIT(mminit)                  ***     111d
  MINIT(qminit)                  ***     878d
  MINIT(xminit)                  ***    1372d
  MIUI                           ***      11d
mif
  CS80(cs80)                     ***     108d
  MINIT(qminit)                  ***     913c
min
  ASCII(asciimodule)                    141d   145c   149c   202c   222c   248c
  CLOCK(clock)                          115d   120c   121c
  DGL_POLY(dgl_poly)             ***   20481c
  GEN(dgl_gen)                         3014d  3097d  3102c  3103c
  LIB(dgl_lib)                   ***   21216c
  MINIT(hminit)                         262d   273d
  MINIT(midecs)                  ***      21d
  MINIT(mminit)                  ***     124d
  MINIT(qminit)                  ***     910c
  MIUI                                  295c   301c   313c
min3
  GEN(dgl_gen)                         3205d  3208c  3210c  3212c  3220c
min_csvp
  MINIT(hminit)                         420d   458c   460c
min_hd_size
  CTABLE                               1378d  1383c
min_size
  CTABLE(options)                       168d   173d   175d   177d   183d   185d   187d
min_vp
  MINIT(hminit)                         418d   458c
minhead
  MINIT(xminit)                        1403d  1525c  1531c  1556c
mini
  CS80(cs80dvr)                  ***     972d
  F9885(f9885dvr)                ***      37d
  INIT(initunits)                ***    2135d
  INITLOAD(bootdammodule)        ***     535d
  INITLOAD(mini)                 ***     300d
  MINIT(mminit)                  ***     111d
  PRINTER(prtdvr)                ***      37d
mini_fm_name
  CTABLE(ctr)                           360d   658c
minidrive
  INITLOAD(mini)                        391d   492c
miniio
  INITLOAD(bootdammodule)        ***     773c
  INITLOAD(mini)                        309d   384d
minindex
  LIBRARIAN                            1720d  1768c  1774c  2001d  2056c  2062c
minit
  INITLOAD(bootdammodule)               628d   632c   636c   642c   702c
```

```
minkana
  CRT(crt)                          40d    47d   389c
minlevel
  INITLOAD(sysglobals)              23d   200d
minrealisc
  DC_DRV(init_dc)            ***   703c
  DC_DRV(intdc)              ***   238d
  DGL_C_OUT(dgl_confg_out)   ***  11313c
  GLE_HPIB(gle_hpib_io)      ***  10495c
  KERNEL(general_0)          ***   987c
  KERNEL(iodeclarations)     ***   243d
minromex
  CRT(crt)                          356d   358d
minsc
  ETU                                36d   700c   779c
minspare
  MINIT(xminit)                     1403d  1525c  1528c  1531c  1543c  1777c  1787c
mintdata
  MINIT(mminit)                      114d   121d   128c
  MIUI                       ***     330c
minute
  A804XDVR(a804xdvr)                279c   308c
  CLOCK(clock)                       80c   120c
  INITLOAD(sysglobals)       ***     225d
  LIBRARIAN                  ***     235c
  LIFDAM(lifmodule)                  124c   556c   588c
  M68KSYS(ci)                        130c   165c   276c   277c
  MIUI                               81c    83c
  SRMDAM(srmdammodule)               374c   388d
  UCSD_DAM(ucsdmodule)               523c   553c
minwidth
  MOREFSYS(mfs)                      564d   571c   577c   589c   590c
misc
  CLOCK(clock)               ***      33d
  CONVERT(convert_text)      ***      27d
  CRT(crt)                   ***      33d
  DISCHPIB(bkgnd)            ***      31d
  ETU                        ***      26d
  F9885(f9885dvr)            ***      37d
  GCRT(crtb)                 ***      33d
  INIT                       ***    2514d
  INIT(fs)                   ***     739d
  INIT(ldr)                  ***    2270d
  INIT(misc)                 ***     151d
  KEYS(keys)                 ***      33d
  LIBRARIAN                  ***      27d
  LIFDAM(lifmodule)          ***      26d
  M68KSYS(ci)                ***      33d
  MIUI                       ***      11d
  MOREFSYS(mfs)              ***      10d
  NONUSKBD1(non_us_kbd1)     ***      31d
  NONUSKBD2                  ***      30d
  PRINTER(prtdvr)            ***      37d
  SEGMENTER(segmenter)       ***      18d
  SRMAM(srmammodule)         ***      33d
  SRMDAM(srmdammodule)       ***      34d
  TAIL                       ***      32d
  TAPEBKUP                   ***     953d
  TAPEBKUP(cs80tbdvr)        ***     305d
  UCSD_AM(ucsd_am)           ***      29d
  UCSD_DAM(ucsdmodule)       ***      29d
```

```
miscinfo
  CRT(crt)                   ***     110d
  GCRT(crtb)                 ***      43d
  SYSDEVS(sysdevs)           ***     105d
miscop
  LIBRARIAN                          734d   966c
miscoptype
  LIBRARIAN                          731d   735d
mitr
  CS80(cs80)                 ***      96d
miui
  MIUI                       ***       8d
mkeymap
  MOUSE(mouse)                        59d   118c
ml
  SRM_DRV(srm)                      1478d  1484c
mm
  CLOCK(clock)                        50d    59c    60c    62c    64c    68c
mmaxint
  INITLOAD(loader)           ***     822d
  INITLOAD(sysglobals)                15d    30d
  UCSD_DAM(ucsdmodule)       ***      46d
mminint
  INITLOAD(sysglobals)                14d    30d
mminit
  MINIT(mminit)              ***     108d
  MIUI                       ***      12d
mminitialize
  MINIT(mminit)                       116d   143d
  MIUI                       ***     348c
mmx
  LIB(dgl_lib)                      20091d 20092d 20246d 20258c 20263d 20281c
mmy
  LIB(dgl_lib)                      20091d 20092d 20246d 20259c 20263d 20282c
mname
  LIBRARIAN                         2736d  2752c  2753c  2788c
mnotfound
  LIBRARIAN                         3320d  3449c  3487c
mnus
  MOREFSYS(mfs)                       33d   423c
mnv
  CTABLE                     ***    1821c
  CTABLE(ctr)                         562d   582c   583c   589c   590c
  CTABLE(options)                     136d   173d   175d   177d   179d   181d   183d   185d   187d
mod_power_of_2
  CS80(cs80dvr)                      1245c  1302c
modblock
  INITLOAD(loader)                   1210d  1264c  1288c  1318c  1325c  1330c  1343c
moddescptr
  CTABLE(ctr)                ***     384d
  INIT(ldr)                         2325d  2352d  2384d
  INITLOAD                   ***    1713d
  INITLOAD(loader)                   875d   905d  1003d  1007d  1036d  1059d  1067d  1069d  1071d  1073d  1105d  1109d  1178d  1179d
                                    1196d  1207d  1301d  1358d  1369d  1398d  1420d  1475d  1498d
  LIBRARIAN                           54d    55d    81d   316d  1474d  1482d  1652d  1721d  1849d  2002d  2129d  2362d
  M68KSYS(ci)                        874d   915d
  SEGMENTER(asm)                      67d   111d   259d
moddirptr
  INITLOAD(loader)                   878d   909d
mode
  SRMDAM(srmdammodule)               201d   258c   285c
  SRM_DRV(srm)                      1050d  1737d  1763c
model
  LIB(dgl_lib)                      20034d 20399d 20404c 20405c
```

```
modeonly
  SRMDAM(srmdammodule)                  199d   261c
modification_ok
  MINIT(qminit)                        1038d  1043c  1044c  1047c  1048c  1052c
modified
  CS80(cs80dvr)                        1149c  1294c  1317c
  CS80(tapebuf)                 ***     42d
modlist
  LIBRARIAN                            3302d  3369c  3370c  3379c  3381c  3382c  3385c  3386c  3387c  3392c  3402c
modnum
  INIT(ldr)                            2408d  2437c  2438c  2452c  2453c  2455c  2457c
  INITLOAD(loader)                     1082d  1206d  1318c  1589d  1622c  1623c
  LIBRARIAN                            2722d  2727c  2734d  2748c  2829d  2836c  2845c  2854c  2855c  3016d  3021c  3022c  3027d  3029c
                                       3030c  3055d  3064c  3065c  3069c
  SEGMENTER(asm)                        177d   195c   196c
modp
  CTABLE(ctr)                           384d   390c   391c   392c   405c
  INITLOAD(loader)              ***    1036d
  LIBRARIAN                            1739c  1754c  1786c  2025c  2039c  2077c  2080c
  M68KSYS(ci)                           915d   922c   923c   928c   929c   933c
  SEGMENTER(asm)                        111d   118c   119c   120c   137c
modptr
  INIT(ldr)                            2325d  2332c  2334c  2335c  2337c  2346c  2352d  2358c  2359c  2364c  2365c  2367c  2384d  2388c
                                       2389c  2398c
  INITLOAD                             1713d  1717c  1719c  1720c  1721c  1722c
  INITLOAD(loader)                     1105d  1109d  1178d  1180c  1196d  1198c  1199c  1201c  1202c  1207d  1256c  1283c  1309c  1310c
                                       1319c  1321c  1323c  1339c  1350c  1352c  1358d  1360c  1361c  1363c  1369d  1372c  1374c  1381c
                                       1398d  1404c  1406c  1415c  1420d  1427c  1429c  1437c  1438c  1439c  1441c  1443c  1465c  1469c
                                       1475d  1481c  1482c  1485c  1488c  1498d  1520c  1521c  1531c  1538c  1543c  1546c  1557c
  LIBRARIAN                            1474d  1482d  1492c  1493c  1495c  1496c  1553c  1555c  1557c  1558c  1559c  1560c  1572c  1573c
                                       1606c  1652d  1659c  1660c  1662c  1667c  1670c  1677c  1678c  1679c  1701c  1705c  1721d  1730c
                                       1731c  1739c  1743c  1849d  1915c  1916c  1949c  1955c  2002d  2016c  2017c  2025c  2029c  2129d
                                       2258c  2299c  2339c  2534c  2535c  2608c  2656c  2718c
  M68KSYS(ci)                           874d   880c   882c   883c   887c   907c
  SEGMENTER(asm)                        259d   261c   262c   270c
modsave
  LIBRARIAN                              55d   391c  1339c  1361c  1408c  1417c
module_addressing
  CS80(cs80)                    ***     137d
  CS80(cs80dsr)                 ***     676c
  TAPEBKUP(cs80tbdvr)                   556c   788c
moduledescriptor
  INITLOAD(loader)                      875d  1002d  1102d  1320c  1433c
  LIBRARIAN                     ***    3221c
moduledirectory
  INITLOAD(loader)                      878d   940d  1441c  1523c
  LIBRARIAN                             326c  1277c  1695c  1894c  1938c  2339c  2527c  2612c  2752c
modulesize
  INITLOAD(loader)              ***     947d
  LIBRARIAN                             353c  2882c
monitortype
  GLE_RGL(gle_ras_out)          ***    8033d
monochrome
  GLE_TYPES(gle_types)          ***    1246d
month
  CLOCK(clock)                           68c   106c
  INITLOAD(sysglobals)          ***     220d
  LIBRARIAN                              113c   117c  3711c
  LIFDAM1(lifmodule)                     121c   332c   347c   551c   553c   554c   737c
  M68KSYS(ci)                            135c   156c   265c   779d
  MIUI                                    44d    65c    67c    89c
  SRMDAM1(srmdammodule)                  366c   387d
  SRM_DRV(srm)                  ***     252d
  UCSD_DAM(ucsdmodule)                   522c   551c
```

```
monthname
  LIBRARIAN                              108d   118c
monthnumber
  M68KSYS(ci)                            230d   250c
months
  LIBRARIAN                              107d   108d
  M68KSYS(ci)                             84d   135c   249c
monthstr
  MIUI                                    49d    67c
monthtype
  M68KSYS(ci)                             82d    84d
moon
  DGL_C_OUT(dgl_config_out)            11040d 11272c 11320c 11330c
mopsproc
  MOUSE(mouse)                           126d   189c
morealpha
  KEYS(keys)                             380d   491c   496c
moregraphics
  KEYS(keys)                             387d   489c   497c
most_recent_escapecode
  INITLOAD(mini)                         319d   376c
most_recent_status
  AMIGO(csamigo)                         157d   313c
  CS80(cs80)                             362d   467c
mostavail
  LIFDAM(lifmodule)                      681d   689c   690c   743c   746c   747c   748c   754c   756c   758c   759c   769c   771c   813c
                                         814c   826c   827c   858c   870c   871c   872c   873c   885c   891c   896c   898c   900c   905c
                                         908c
mounted
  ETU                                     61d   282c   365c   388c   886c   898c   899c
mountvolume
  ETU                                    219d   364c   391c
mouse
  MOUSE                         ***     203d
  MOUSE(mouse)                  ***      28d
mouseinit
  MOUSE                         ***      26d
move
  DGL_POLY(dgl_poly)                   20824c 21106c 21168c
  GLE_GEN(gle_gen)              ***     2060c
  GLE_HPGL(gle_hpgl_out)        ***     7527c
  GLE_RGL(gle_ras_out)          ***     8396c
  GLE_SMARK(gle_smark)                  5138c  5163c  5174c
  GLE_TYPES(gle_types)          ***     1052d
  LIB(dgl_lib)                         20043d 20130c 20557d 20578c 20638c 20703c 21018c
  LIBRARIAN                              708d  1027c
move_it
  DGL_POLY(dgl_poly)                   20422d 20819c 20824c 20826c
movebytes
  LIBRARIAN                     ***    1855d
movedefs
  INIT(ldr)                     ***    2476c
  INITLOAD(loader)                      1091d  1474d  1639c
  SEGMENTER(asm)                         244c   306c   322c
```

```
moveleft
  ASCII(asciimodule)                    74c   203c   223c   251c
  CONVERT(convert_text)      ***        73c
  CRT(crt)                             331c   585c   605c   642c   643c   645c
  CS80(cs80dvr)                       1268c  1283c  1295c  1318c
  EDRIVER(edriver)           ***       199c
  EPROMS(eproms)             ***        68c
  INIT(misc)                           600c   646c   649c   659c   664c   675c   677c
  INITLOAD(loader)           ***       1152c
  INITLOAD(mini)                       441c   445c
  LIBRARIAN                            481c   482c  2403c  2894c  3157c  3241c
  MOREFSYS(mfs)              ***       657c
  SRM_DRV                    ***       2104c
  UCSD_AM(ucsd_am)                      83c    98c
moveright
  CRT(crt)                             341c   607c
  INITLOAD(loader)                    1159c  1617c
  LIBRARIAN                           2319c  2411c
movesize
  ETU                                  858d  1050c  1051c  1053c  1055c  1097c  1100c  1102c  1106c  1110c
  LIFDAM(lifmodule)                    387d   442c   443c   444c   445c   447c   448c   450c
moving
  GLE_HPGL(gle_hpgl_out)              7021d  7104c  7110c  7339c  7342c
mp
  CS80(cs80dvr)                        985d  1018d  1023c
  CTABLE                              1217d  1535c  1537c  1541c  1543c  1548c  1551c  1555c  1557c  1559c  1568c  1569c  1570c  1571c
                                      1575c  1577c  1658s  1662s  1664s  1668s  1672s  1674s  1678s  1682s  1684s  1689s  1694s  1696s
                                      1697s  1701s  1703s  1707s  1712s  1714s  1717s  1719s  1723s  1728s  1730s  1734s  1739s  1741s
                                      1746s  1755s  1757s
  CTABLE(ctr)                          311d   313d   314d   315d   578d   585c   586c   587c   597d   599c   600c   604d   608c   610c
                                       611c   615d   617c   621d   623c
  HEAPT(hpm)                            68d   102c   104c   112c   148c   150c   156c   158c   159c   167c   169c   170c   184c   185c
                                       196c   226c   236c   246c   252c   253c
  LIBRARIAN                            316d   323c
  SRM_DRV(srm)                         875d  1216c  1379c  1380c  1381c  1382c  1383c  1385c  1387c  1388c  1390c  1391c  1392c  1428c
                                      1430c  1431c
mp_type
  CS80(cs80dvr)                        976d   985d   996d  1018d
  CTABLE                              1217d  1375d
  CTABLE(ctr)                          285d   309d   311d   313d   314d   315d   525d   527d   528d   529d   530d   531d   532d   533d
                                       534d   535d   536d   537d   538d   540d   578d   597d   604d   615d   621d
  CTABLE(scanstuff)                   1043d  1057d  1135d
mphydata
  MINIT(mminit)                        115d   132d   139c
  MIUI                       ***       473c
mrbase
  INITLOAD(loader)                    1399d  1405c  1410c  1411c
  LIBRARIAN                           1653d  1669c  1674c  1675c  1683c  1684c  1707c
mscode
  INITLOAD(bootdammodule)              572d   800c
msg
  ETU                                  100d   104c   105c
msg_packet_ptr
  SRM_DRV(srm)                         308d   823d
msg_packet_type
  SRM_DRV(srm)                         307d   308d  1389c  1429c
msgline
  ETU                                  866d   943c   980c   988c   991c
msize
  ETU                                  859d  1096c  1097c  1098c  1099c  1100c
mstates
  KEYS(keys)                            47d   466c
mstring
  ETU                                 1148d  1151c
```

```
msus
  CTABLE                              1283d  1298c  1301c  1302c  1308c  1316c  1334c  1344c  1358c  1380c  1392c  1394c  1416c  1419c
                                      1422c
  CTABLE(brstuff)                      937d   985d   997c  1003c  1004c  1006c  1007c  1008c  1009c  1011c  1012c  1016c  1017c
  INITLOAD(bootdammodule)              587d   628d   632c
msus1
  CTABLE(ctr)                          317d   410d   412c   413c   414c   415c   416c
msus2
  CTABLE(ctr)                          317d   410d   412c   413c   414c   415c   416c
msus_array_size
  CTABLE                              1196d  1199d  1299c  1440c  1524c
msus_array_type
  CTABLE                              1199d  1203d  1204d  1205d  1290d
msus_type
  CTABLE                              1193d  1195d  1199d  1206d  1239d  1283d  1356d
  CTABLE(brstuff)                      932d   937d   985d
  CTABLE(ctr)                          277d   298d   317d   410d
msuss_match
  CTABLE                     ***      1298c
  CTABLE(ctr)                          317d   410d   412c
msustype
  INITLOAD(bootdammodule)              569d   577d   587d   628d
mth
  CLOCK(clock)                         101d   106c   107c   108c   109c
mtype
  INITLOAD(bootdammodule)              570d   800c
multiport
  CS80(cs80)                 ***        82d
multiunit
  CS80(cs80)                 ***        83d
munit
  INITLOAD(bootdammodule)    ***       571d
munknown
  DGL_TOOLS(dgl_tools)       ***      20053c
  GLE_RGL(gle_ras_out)       ***       8430c
  GLE_TYPES(gle_types)       ***       1253d
mvb
  UCSD_DAM(ucsdmodule)                  68d   109c   116c   139c   154c   622c   633c   651c   677c
mvs
  CTABLE(ctr)                          562d   584c   586c
  CTABLE(options)                      135d   173d   175d   177d   179d   181d   183d   185d   187d
my_address
  GLE_HPIB(gle_hpib_io)              10119d 10127c 10215c 10301c
  IOLIB(hpib_1)                        395d   443d   451c   537c   623c
  IOLIB(hpib_2)              ***       1827c
my_boolean
  IOLIB(hpib_0)                       1747d  1753c  1754c
my_byte
  IOLIB(hpib_3)                       2082d  2087c  2088c
my_code
  KERNEL(iodeclarations)               655d   663d   667c
my_isc
  DC_DRV(intdc)              ***       418c
  KERNEL(general_0)                   1208c  1266c
  KERNEL(iodeclarations)               500d   514d   528d   681c
my_msg
  IOLIB(general_3)                    1112d  1114c  1121c  1122c  1123c  1124c  1125c  1126c  1127c  1128c  1129c  1130c  1131c  1132c
                                      1133c  1134c  1135c  1136c  1137c  1138c  1139c  1140c  1141c  1142c  1143c  1144c  1153c  1155c
                                      1156c  1157c  1158c  1159c  1160c  1161c  1162c  1163c  1164c  1165c  1170c  1172c  1173c  1175c
my_num
  IOLIB(general_1)                     306d   311c   312c
my_ptr
  KERNEL(iodeclarations)               673d   680c   681c
```

```
my_special_pad
  MINIT(cs80ir)                  ***    544d
my_status_mask
  CS80(cs80dsr)                         795d   888c   950c
my_sysflag
  C_HOOK                                 52d   143c
my_value
  IOLIB(general_1)                      319d   321c   325c
  KERNEL(general_0)                     947d   949c   954c
mybit
  IOLIB(serial_0)                      2233d  2244c  2246c  2248c  2254c  2264c  2266c  2268c  2275c  2296d  2307c  2309c  2311c  2317c
                                       2327c  2329c  2331c  2338c  2360d  2375c  2380c  2385c  2390c  2395c  2400c  2405c  2412c  2423c
                                       2428c  2433c  2438c  2443c  2448c  2453c  2460c
mybool
  IOLIB(hpib_1)                         498d   503c   504c
  IOLIB(serial_0)                      2363d  2412c  2460c  2468c  2476c
mycharptr
  GLE_KNOB(gle_knob_in)               18110d 18118c 18132c
  KERNEL(general_0)                     893d   903c   912d
mycount
  DC_DRV(intdc)                         413d   458c   464c   465c   466c   475c   476c   477c
myisrib
  DC_DRV(init_dc)                       648c   652c   660c
  DI_DRV(init_discint)                  253c   257c   265c
  G_DRV(init_gpio)                      268c   272c   280c
  H_DRV(init_hpib)                      276c   280c   288c
  KERNEL(general_0)              ***   1216c
  KERNEL(iodeclarations)                493d   507d   521d
  RS_DRV(init_rs)                       256c   260c   268c
myrate
  IOLIB(serial_3)                      2533d  2535c  2536c  2537c  2538c  2539c  2540c  2541c  2542c  2543c  2544c  2545c  2546c  2547c
                                       2548c  2549c  2550c  2552c
myval
  IOLIB(serial_3)                      2600d  2612c  2617c  2622c  2632c  2641c  2648c  2653c  2664c  2684d  2693c  2694c  2695c  2696c
                                       2706c  2716c  2738d  2750c  2751c  2752c  2753c  2754c  2760c  2771c  2772c  2773c  2774c  2775c
                                       2782c
n
  A804XDVR(a804xdvr)                    151d   154c
  DISCHPIB(bkgnd)                        72d    75d    88c    99c   113c
  GCRT(crtb)                     ***    121d
  GLE_RGL(gle_ras_cut)                 8320d  8332c  8333c  8334c  8335c  8355c  8356c  8357c  8360c  8361c  8363c  8364c
  INITLOAD(loader)                     1038d  1225d  1229c  1234c  1242c  1243c
  LIBRARIAN                            1740c  1758c  1764c  1767c  1790c  1791c  3270d  3275c  3279c  3283c  3285c  3289d  3291c
  LIFDAM(lifmodule)                    1060d  1071c  1072c
  MTUI                                   15d   172c   331c   349c   474c
  SRMDAM(srmdammodule)                  136d   163c   184d   187c   189c   390d   417c
  SYSDEVS(sysdevs)                      391d   580d   582c
n3
  GLE_RGL(gle_ras_cut)           ***   8030d
n_glines
  DGL_C_OUT(dgl_confg_out)       ***  11062c
  GLE_RGL(gle_ras_cut)                 8038d  8475c
name
  GLE_FILE(gle_file_io)                9052d  9057c  9058c
  INIT(fs)                       ***    770d
  INIT(ldr)                            2281d  2351d  2355c  2357c  2362c
  M68KSYS(ci)                           843d   848c   849c   914d   918c   920c   926c
  UCSD_DAM(ucsdmodule)                   42d   440c   443c   452c   461c   465c   661c
name_ptr
  SEGMENTER(asm)                 ***     58d
name_set_array
  SRMDAM(srmdammodule)                  662d   903d
  SRM_DRV(srm)                          927d   928d
```

```
name_set_array_three
  SRMDAM(srmdammodule)                  134d  1025d  1269d  1333d
  SRM_DRV(srm)                   ***    926d
name_size
  DGL_C_IN(dgl_confg_in)         ***  12112c
  DGL_C_OUT(dgl_confg_out)            11440c 11481c
  GLE_FILE(gle_file_io)                9012d  9057c
  GLE_HPIB(gle_hpib_io)               10013d 10493c
name_type
  SRMDAM(srmdammodule)                   97d   111d   462d   505d   571d   650d   664d   905d   965d   993d  1024d  1175d  1213d  1214d
                                       1248d  1266d  1313d  1332d
  SRM_DRV                              1785d  1790d  1907d  1970d
  SRM_DRV(srm)                          230d   271d   275d   290d   291d   316d   337d   338d   341d   349d   410d   440d   441d   442d
                                        457d   458d   605d   607d   608d   614d   616d   791d  1010d  1018d  1026d  1030d  1031d  1032d
                                       1046d  1060d  1065d  1082d  1094d  1516d  1552d  1590d  1628d  1659d  1660d  1661d  1733d
name_type_len
  SRMDAM(srmdammodule)                  188c   673c  1544c  1591c  1593c
  SRM_DRV(srm)                           52d   230d
namelen
  SRMDAM(srmdammodule)                 1524d  1532c  1565c  1584c  1591c
nb
  AMIGO(csamigo)                        246d   251d   252d   253d   254d   255d   256d   257d   258d   259d   260c   261d   262d   263d
                                        269c
  DISCHPIB(dischpib)                    196d   198d   202d   287d   296c   300c   357d   367c   373c   440d   461c   478c
nbb
  CS80(cs80)                     ***    102d
nblocks
  INIT(fs)                       ***    790d
nbpb
  CS80(cs80)                     ***    101d
  CS80(cs80dsr)                         926c   939c
  CS80(cs80dvr)                        1075c  1076c
  MINIT(qminit)                  ***    927c
  TAPEBKUP                             1250c  1302c  1303c  1319c  1320c  1445c  1446c
nbuttons
  SYSDEVS(sysdevs)               ***    325d
nbytes
  ETU                            ***   1162d
nc
  KEYS(keys)                            157d   159c   160c   162c   165c   166c   167c   168c   169c   170c   171c   172c   173c   174c
                                        175c   176c   177c   178c
nchars
  MOREFSYS(mfs)                          57d    66c    68c    73c    77c    79c    83c    88c   119c   120c   127c   128c   133c   145c
                                        146c   159c   171c   172c   177c   178c   191c   204c   205c   210c   211c   218c   219c   225c
                                        237c   238c   242c   243c   249c   250c   258c   263c   275c   276c   280c   281c   287c   288c
                                        296c   297c   302c   314c   315c   320c   321c   329c   334c   340c   347c   348c   353c   354c
                                        359c   360c   390c   391c
ndac_line
  KERNEL(iodeclarations)         ***    393d
ndfs
  CRT(crt)                       ***    122d
  GCRT(crtb)                     ***     55d
  SYSDEVS(sysdevs)               ***     77d
ndrives
  CTABLE(brstuff)                       970d   981c
needc
  M68KSYS(ci)                           409d   445c   477c   521c
needcar
  SRMDAM(srmdammodule)                  202d   254c   269c   274c   278c
needmodeorpass
  SRMDAM(srmdammodule)                  202d   241c   248c
needpass
  SRMDAM(srmdammodule)                  202d   259c   264c   280c
```

```
neg
  GLE_UTLS(gle_utls)              21075d 21088c 21091c 21102c
new_cat_organization
  SRM_DRV(srm)              ***    608d
new_cursor
  GLE_HPGLI(gle_hpgl_in)    ***  18234d
new_file_name_header
  SRM_DRV                   ***   1811c
  SRM_DRV(srm)              ***    521d
new_password
  SRM_DRV(srm)              ***    614d
new_state
  GLE_HPGL(gle_hpgl_out)          7094d  7102c  7108c
new_vol_password
  SRM_DRV(srm)                     442d  1673c
new_volume_name
  SRM_DRV(srm)                     441d   607d  1672c
newbytes
  DGL_POLY(dgl_poly)              20940c 20985c
  INITLOAD(bootdammodule)   ***    743c
  LIBRARIAN                       2152d  2309d  2326c  2327c  2328c  2329c  2330c  2646c  2651c  2654c  2655c  2707c  2708c  2709c
  M68KSYS(ci)                      455c   808c
  SEGMENTER(asm)                    45d   327c
newdeftable
  LIBRARIAN                       2010d  2035c  2114c  2118c
newdirectory
  LIBRARIAN                         86d  1990c  2111c  2527c  2530c  2880c  2883c  2990c
neweof
  SRMDAM(srmdammodule)            1314d  1318c  1319c  1322c
newextsize
  LIBRARIAN                         85d  1750c  1789c  1797c  1801c  1983c  1984c
newexttable
  LIBRARIAN                         84d   195c   196c   397c  1749c  1985c
newindex
  LIBRARIAN                       2148d  2171c  2181c  2199c  2216c  2217c  2527c  2528c  2529c  2531c
newlistname
  LIBRARIAN                       2919d  2929c  2930c  2931c  2932c  2934c  2936c  2941c
newmodname
  LIBRARIAN                         82d  1697c  1895c  1896c  1897c  1901c  1903c  2894c  2975c  3001c  3002c  3236c  3240c  3241c
                                  3568c  3569c
newmodptr
  INITLOAD(loader)                1421d  1433c  1434c  1445c  1451c
newmods
  INIT(ldr)                       2388c  2432c
  INITLOAD(loader)                1059d  1198c  1303c  1339c  1360c  1365c  1372c  1404c  1427c  1520c  1580c  1600c
  LIBRARIAN                        194c   391c   392c   395c   402c  1168c  1169c  1275c  1321c  1339c  1349c  1361c  1383c  1408c
                                  1417c  1555c  1659c  1664c  1667c  1730c  1915c  2016c  2534c  2749c  2826c  3009c  3225c  3533c
                                  3542c  3627c  3639c  3644c  3687c  3688c  3732c
  SEGMENTER(asm)            ***    191c
newname
  SRM_DRV(srm)                    1031d  1660d  1672c
newnfns
  SRM_DRV                         1786d  1799c  1812c
  SRM_DRV(srm)              ***   1061d
newnsaptr
  SRM_DRV                         1787d  1796c
  SRM_DRV(srm)              ***   1062d
newparm
  MINIT(qminit)                   1037d  1046c  1047c  1049c
newpass
  SRM_DRV(srm)                    1032d  1661d  1673c
newpath
  SRM_DRV                         1788d  1814c
  SRM_DRV(srm)              ***   1063d

newptr
  LIBRARIAN                       1858d  1868d  1876c  1877c  1929c  1930c  2144d  2312c  2313c  2318c  2326c  2328c  2331c  2374c
                                  2404c  2412c  2434c  2437c  2451c  2475c  2485c  2486c  2488c  2581c  2582c  2585c  2642c  2647c
                                  2652c  2657c
newrtpass
  SRM_DRV                         1790d  1815c
  SRM_DRV(srm)              ***   1065d
newstartdefs
  INITLOAD(loader)                1091d  1474d  1479c  1490c  1491c
newstrptr
  LIBRARIAN                       1726d  1783c  1784c  1787c  2007d  2074c  2075c  2082c
newsysunit
  M68KSYS(ci)                      841d  1132c
newtextrec
  LIBRARIAN                       1850d  1870c  1876c  1883c  1884c  1888c  1920c  1921c  1945c  1947c
newtextrecs
  LIBRARIAN                       1854d  1887c  1914c  1978c
newunit
  M68KSYS(ci)                      842d   846c   848c
newwd
  SRM_DRV                         1789d  1813c
  SRM_DRV(srm)              ***   1064d
newwords
  ETU                              888c   940c
  HEAPT(hpm)                       128c   269c   290c   295c
  INITLOAD                  ***   1675c
  LIFDAM(lifmodule)         ***    403c
  UCSD_DAM(ucsdmodule)      ***    584c
next
  ETU                               47d   682c   690c
  HEAPT(hpm)                       216d   241c   242c   244c   246c   248c   249c   252c   253c   255c
  HPHIL(hphil)                     264c   265c
  MOUSE(mouse)              ***    191c
  SYSDEVS(sysdevs)          ***    341d
next_subpolygon
  DGL_POLY(dgl_poly)              20201d 20221c 20225c 20230c
nextavail
  LIFDAM(lifmodule)                681d   758c   781c   783c   791c   793c   815c   816c   839c   840c   870c   871c   872c   873c
                                   892c   905c
nextblock
  LIBRARIAN                         71d  1980c  1982c  1985c  1986c  2113c  2118c  2119c  2173c  2174c  2175c  2214c  2215c  2882c
                                  2891c
nextchar
  MOREFSYS(mfs)                    401d   416c   418c   419c   424c   425c   430c   431c   440c   441c   447c   448c   453c   454c
                                   463c   484c   469c   470c   478c   479c   486c   487c
nextchr
  MOREFSYS(mfs)                     38d    91c    92c    95c    97c   539c   547c
nextfreeword
  HEAPT(hpm)                        48d    49d   276c   281c
nextptr
  INITLOAD(loader)                1358d  1362c  1363c
  LIBRARIAN                       1652d  1661c  1662c
nextref
  LIBRARIAN                        405d   525c   536c  1303c
nfns
  SRMDAM(srmdammodule)             661d   758c   759c   761c   799c   807c   832c   839c   855c   902d   922c  1268d  1292c  1297c
                                  1301c  1302c
  SRM_DRV                         1903d  1916c  1923c  1966d  1977c  1984c
  SRM_DRV(srm)                    1006d  1014d  1022d  1042d  1078d  1090d  1548d  1561c  1574c  1586d  1599c  1612c  1624d  1635c
                                  1638c  1645c  1729d  1747c  1750c  1757c
nfns1
  SRMDAM(srmdammodule)            1177d  1216d
nfns2
  SRMDAM(srmdammodule)            1180d  1219d
```

```
nilgv:
  LIBRARIAN                          303d   539c  1063c  1415c
nilptr
  HEAPT(hpm)                          53d    61d   122d   123d   142c   145c   151c   162c   224c   234c   241c   242c   249c   251c
                                     257c   265c   267c   282c   289c   307c   312c   315c
nindx
  SRMDAM(srmdammodule)               654d   771c   782c   786c
nintdata
  MINIT(hminit)                      252d   270d   277c   279c
  MIUI                        ***    331c
nlem
  MORIFSYS(mfs)                       27d    29d    88c
no
  GLE_TYPES(gle_types)        ***   1263d
no_capslock
  KEYS(keys)                         216c   265c   284c   300c   320c
  NONUSKBD1(non_us_kbd1)             171c   178c   179c   180c   188c   189c   193c
  NONUSKBD2                          153c   157c   158c   175c   176c   181c   182c   183c   199c
  SYSDEVS(sysdevs)            ***    269d
no_card
  KERNEL(general_0)           ***   1014c
  KERNEL(iodeclarations)      ***    323d
  PRINTER(prtdvr)             ***    290c
no_control
  KEYS(keys)                         214c   267c   278c   299c
  SYSDEVS(sysdevs)            ***    271d
no_dam_name
  CTABLE(ctr)                        349d   633c
no_data_found
  CS80(cs80)                  ***    170d
  CS80(cs80dsr)               ***    691c
  TAPEBKUP(cs80tbdvr)                571c   803c
no_extension
  KEYS(keys)                         205c   268c   278c   300c   343c
  SYSDEVS(sysdevs)            ***    272d
no_id
  KERNEL(general_0)           ***    986c
  KERNEL(iodeclarations)      ***    348d
no_isc
  DC_DRV(intdc)               ***    533c
  DISCHPIB(dischpib)          ***    467c
  DMA_DRV(init_dma)                  224c   227c
  IOLIB(general_4)                  1270c  1300c  1307c  1320c  1333c  1374c  1378c  1381c  1564c  1569c  1573c  1591c  1596c  1600c
                                    1620c  1626c  1647c  1662c
  KERNEL(general_0)                 1294c  1299c
  KERNEL(iodeclarations)             295d   677c
  SRM_DRV(srm)                      1197c  1275c  1276c  1290c  1317c  1318c  1319c
no_kbc
  A804XDVR(a804xdvr)                 186d   187d   188d   189d   225c
  SYSDEVS(sysdevs)                   146d   617c
no_parity
  IOLIB(serial_3)                   2750c  2771c
  KERNEL(iodeclarations)      ***    401d
no_proc
  SEGMENTER(asm)                     102c   106c   116c
no_shift
  KEYS(keys)                         215c   266c   299c
  NONUSKBD1(non_us_kbd1)      ***     68c
  SYSDEVS(sysdevs)            ***    270d
no_spares_available
  CS80(cs80)                  ***    167d
  CS80(cs80dsr)               ***    685c
  TAPEBKUP(cs80tbdvr)                565c   797c
```

```
no_tfr
  IOLIB(general_4)                  1269c  1420c
  KERNEL(iodeclarations)      ***    559d
  SRM_DRV(srm)                      1196c  1343c
nobreak
  CRT(crt)                    ***    111d
  GCRT(crtb)                  ***     44d
  SYSDEVS(sysdevs)            ***     69d
nocapabilities
  SRMDAM(srmdammodule)               64d    304c  1068c  1401c
nochain
  M68KSYS(ci)                        58d    141c  1037c  1068c  1098c  1171c
nocrt
  DGL_C_OUT(dgl_confg_out)         11286c 11296c 11335c 11342c 11346c
  SYSDEVS(sysdevs)                   113d   631c
nodam
  INIT(initunits)                   2141d  2232d  2241c
node
  INITLOAD(bootdammodule)            783d   788c   789c   790c
nodeallocate
  SRM_DRV(srm)                       460d  1694c
nodeccm
  INITLOAD(bootdammodule)            779d   786c
nodestr
  INITLOAD                          1764c  1765c  1770c  1771c  1789c
  INITLOAD(bootdammodule)     ***    540d
  TAIL                                68c    75c
nodevice
  CTABLE                      ***   1193d
  CTABLE(brstuff)                    991d   992d   993d   994d  1004c
  CTABLE(ctr)                 ***    261d
  CTABLE(scanstuff)                 1111c  1144c
nodisc
  F9885(f9885dvr)                     48d   219c
noerror
  F9885(f9885dvr)                     48d   206c
noerrs
  HPHIL(hphil)                        63d   143c   169c
nointhpib
  INITLOAD(sysglobals)        ***    284d
  KERNEL(general_0)           ***   1015c
noisr
  INIT(initunits)                   2142d  2167d
  MINIT(xminit)               ***   1520c
nokbd
  SYSDEVS(sysdevs)                   145d   616c
nokeyboard
  DGL_C_IN(dgl_confg_in)      ***  12050c
  INITLOAD(sysglobals)        ***    285d
nomap
  CRT(crt)                           171d   363c   384c   686c   689c
non_char
  KEYS(keys)                         159c   181c
  SYSDEVS(sysdevs)                   239d   536c   538c   543c   555c   556c   560c   563c   641c
non_dominant
  GLE_HGL(gle_ras_out)        ***   8079d
non_dominant_support
  GLE_HPGL(gle_hpgl_out)      ***   7698c
  GLE_TYPES(gle_types)        ***   1093d
non_dominate_mode
  DGL_VARS(dgl_vars)          ***   1202d
non_us_kbd1
  NONUSKBD1                   ***    209d
  NONUSKBD1(non_us_kbd1)      ***    210c
```

```
non_us_kbd2init
  NONUSKBD2                    ***     28d
nona_alpha_key
  KEYS(keys)                           439c   565c
  NONUSKBD1(non_us_kbd1)       ***     194c
  NONUSKBD2                            171c   177c   210c   215c   222c
  SYSDEVS(sysdevs)             ***     258d
nonabortive_ioresult_set
  MIUI                                 120d   209c
  TAPEBKUP                            1051d  1098c
nonadv_key
  KEYS(keys)                           194c   335c   438c   565c
  NONUSKBD1(non_us_kbd1)       ***     196c
  NONUSKBD2                            196c   207c   213c   217c   220c
  SYSDEVS(sysdevs)             ***     258d
nonadvkeys
  KEYS(keys)                           142d   196c   221c
none
  LIBRARIAN                           3294d  3345c  3530c  3538c  3568c  3587c  3602c
  SRMDAM(srmdammodule)         ***     201d
nonus1init
  NONUSKBD1                    ***      27d
nonzero
  MOREFSYS(mfs)                        402d   414c   435c   449c   459c   498c
nop
  CS80(cs80)                           542d   547c
  MINIT(cs80ir)                        830d   836c
  TAPEBKUP(cs80tbr)                    115d   123c   192d   199c   224d   237c
nop1
  CS80(cs80)                           570d   581c
nop2
  CS80(cs80)                           573d   585c
nopassyet
  SRMDAM(srmdammodule)                1530d  1538c  1563c  1569c  1571c
nopatch
  LIBRARIAN                           2368d  2419c  2475c
nopower
  F9885(f9885dvr)                       48d   216c
norange
  LIBRARIAN                            306d  1199c  1250c  1460c  1461c
noreconfig
  HPHIL(hphil)                          62d   144c   170c
norecord
  F9885(f9885dvr)                       48d   230c   235c
nores
  LIBRARIAN                            143d   194c
normal_completion
  AMIGO(csamigo)               ***      69d
normal_vertex
  DGL_POLY(dgl_poly)                 20386d 20503c 20504c 20726c 20736c
normalized_cos
  DGL_POLY(dgl_poly)                 20320d 20618c 20621c 20624c 20629c 20835c 20836c
normalized_one
  DGL_POLY(dgl_poly)                 20043d 20611c 20624c 20630c 20900c 20901c 20994c 20995c
normalized_sin
  DGL_POLY(dgl_poly)                 20320d 20611c 20618c 20621c 20629c 20630c 20834c 20835c
normshift
  HPHIL(hphil)                          156d   270c   295c
normshiftop
  HPHIL(hphil)                 ***     295c
  SYSDEVS(sysdevs)             ***     306d
```

```
norwegian_kbd
  A804XDVR(a804xdvr)           ***     191d
  NONUSKBD2                            179c   206c   209c
  SYSDEVS(sysdevs)             ***     147d
not_ready
  CS80(cs80)                   ***     168d
  CS80(cs80dsr)                ***     687c
  TAPEBKUP(cs80tbdvr)                  567c   799c
notbusy
  GLE_RGL(gle_ras_out)         ***    8117c
  GLE_TYPES(gle_types)         ***    1246d
notcancels
  LIBRARIAN                           1489d  1498c  1505c  1511c
notdone
  LIBRARIAN                           1488d  1498c  1499c  1507c
notendofparms
  M68KSYS(ci)                           409d   436c   447c   469c   471c
notice
  INITLOAD(loader)             ***     945d
  LIBRARIAN                            333c   334c  1964c
notrack
  F9885(f9885dvr)                       49d   227c
notready
  F9885(f9885dvr)                       62d   208c
notype
  LIBRARIAN                            298d  1440c
nounit
  INIT(initunits)                     2139d  2228d  2241c
nowopen
  LIFDAM(lifmodule)                   1105d  1130c  1131c  1132c
nprompts
  SYSDEVS(sysdevs)             ***     324d
nps
  SRMDAM(srmdammodule)                 171d   187c   197d   242c   300c   301c   310c   464d   487c   490c   493c   969d   976c  1030d
                                      1060c  1061c  1062c  1064c  1065c  1069c  1070c  1087c  1124c  1135c  1147c
  SRM_DRV(srm)                        1027d  1047d  1629d  1638c  1650c  1734d  1746c  1750c  1771c
nrecords
  F9885(f9885dvr)                       58d   165c
nrfd_line
  KERNEL(iodeclarations)       ***     392d
ns1_kbd
  SYSDEVS(sysdevs)             ***     150d
ns2_kbd
  SYSDEVS(sysdevs)             ***     150d
ns3_kbd
  SYSDEVS(sysdevs)             ***     150d
nsa
  SRMDAM(srmdammodule)                 134d   142c   148c   153c   158c   662d   761c   807c   839c   855c   903d   922c  1025d  1038c
                                      1079c  1081c  1085c  1086c  1094c  1095c  1119c  1123c  1129c  1134c  1143c  1146c  1158c  1159c
                                      1164c  1269d  1286c  1301c  1302c
nsa1
  SRMDAM(srmdammodule)                1333d  1359c  1391c
nsaptr
  SRM_DRV                             1904d  1913c  1967d  1974c
  SRM_DRV(srm)                        1007d  1015d  1023d  1043d  1079d  1091d  1549d  1558c  1587d  1596c  1625d  1634c  1730d  1745c
nsectors
  MINIT(iramigo)                        190d   196d   204c
nspm
  MINIT(bminit)                ***      99c
  MINIT(hminit)                ***     291c
  MINIT(midecs)                ***      29d
  MINIT(mminit)                ***     136d
  MINIT(qminit)                ***     923c
  MIUI                         ***     438d
```

```
nspi
  MINIT(bminit)                  ***    100c
  MINIT(hminit)                  ***    293c
  MINIT(midecs)                  ***     30d
  MINIT(mminit)                  ***    137d
  MINIT(qminit)                         926c   927c
  MIUI                           ***    438d
nta
  CS80(cs80)                     ***    215d
ntps
  MINIT(bminit)                  ***     99c
  MINIT(hminit)                  ***    292c
  MINIT(midecs)                  ***     28d
  MINIT(mminit)                  ***    135d
  MINIT(qminit)                  ***    924c
  MIUI                           ***    438d
nul
  LIBRARIAN                             3120d  3137c
null
  INIT(misc)                     ***    159d
null_bcd12
  MIUI                                  426d   439d   447d
null_dav
  CTABLE                                1190d  1193d
null_mp
  CTABLE(ctr)                           540d   555c
null_msus
  CTABLE                                1192d  1442c  1443c  1444c
null_pp
  CTABLE(ctr)                           562d   573c
null_tm_name
  CTABLE(ctr)                           356d   633c   639c
nullchar
  CONVERT(convert_text)                  82c   111c   114c   118c
  INIT(misc)                     ***    159d
  LIBRARIAN                      ***   1399c
num
  DGL_POLY(dgl_poly)                   20274d 20298c 20300c 20301c
  INITLOAD(mini)                        391d   492c
  IOLIB(general_1)                      243d   305d   312c
  IOLIB(general_2)                      798d   812d   833d   887c  1013d  1017c
  SRM_DRV(srm)                         1513d  1520c
num_bits
  IOLIB(serial_3)                      2507d  2599d  2610c  2615c  2620c  2639c  2644c  2651c
num_char_bit
  IOLIB(serial_3)                      2510d  2683d  2691c
num_file_name_sets
  SRM_DRV(srm)                          267d  1520c
num_integer
  GEN(dgl_gen)                         3046d  3136d  3163c
num_points
  DGL_POLY(dgl_poly)                   20021d 20024d 20027d 20029d 20192d 20223c 20909d 20935c 20940c 20960c 21025d 21040c 21060c 21071d
                                       21086c 21105c 21116d 21130c 21136c 21147d 21161c 21167c
  LIB(dgl_lib)                         20045d 20047d 20676d 20685c 20688c 20692d 20701c 20704c
num_protect_code_sets
  SRM_DRV(srm)                          424d   503d   606d  1650c  1771c
num_real
  GEN(dgl_gen)                         3046d  3136d  3162c
num_vert
  DGL_POLY(dgl_poly)                   20420d 20644c 20646c 20648c 20659c 20661c 20662c 20688c
numaxes
  HPHIL(hphil)                          104c   116c   118c
  MOUSE(mouse)                           45d    84c    96c   104c
  SYSDEVS(sysdevs)               ***    319d
```

```
number
  CRT(crt)                              315d   319c   322c
  GCRT(crtb)                            224d   229c   230c   233c   235c
  MOREFSYS(mfs)                          47d   120c   127c   128c   133c   146c   159c   172c   178c   191c   205c   211c   219c   225c
                                        238c   243c   250c   263c   276c   281c   288c   296c   297c   302c   315c   321c   334c   341c
                                        348c   354c   360c   391c   398d   418c   424c   430c   440c   447c   453c   463c   469c   478c
                                        486c   517d   526c   528c   535d   544c   548c
number_conversion_ok
  MIUI                                  292d   304c   305c   310c   313c
number_dgl_linestyles
  DGL_HPGL(dgl_hpgl)                   17268c 17269c
  DGL_INQ(dgl_inq)               ***    6336c
  DGL_POLY(dgl_poly)             ***   20136c
  DGL_RAS(dgl_raster)            ***   17599c
  DGL_VARS(dgl_vars)             ***    1084d
  LIB(dgl_lib)                   ***   20479c
number_markers
  DGL_HPGL(dgl_hpgl)             ***   17270c
  DGL_INQ(dgl_inq)               ***    6348c
  DGL_RAS(dgl_raster)            ***   17598c
  DGL_VARS(dgl_vars)             ***    1085d
number_of_tracks
  INIT(misc)                            265d   267c
number_polygon_styles
  DGL_HPGL(dgl_hpgl)             ***   17281c
  DGL_INQ(dgl_inq)                      6073c  6374c
  DGL_POLY(dgl_poly)                   20152c 20178c
  DGL_RAS(dgl_raster)            ***   17652c
  DGL_VARS(dgl_vars)             ***    1086d
  LIB(dgl_lib)                   ***   21453c
number_vols
  CTABLE                                1537c  1551c  1571c
  CTABLE(ctr)                            311d   313d   314d   578d   591c   604d   608c   609c   610c   615d   617c
numberofspares
  MINIT(xminit)                         1411d  1543c  1645c  1748c
numblocks
  LIBRARIAN                             2832d  2857c  2861c  2863c  2865c  2871c
numbuilt
  IOLIB(general_2)                       838d   855c   889c   904c
numdata
  BAT(bat)                                67d    77c    78c    79c    80c    81c
  SYSDEVS(sysdevs)                       185d   380d   437d   438c   445d
numdigits
  MOREFSYS(mfs)                          564d   590c   593c   594c   604c   605c   610c   642c   643c   644c   645c
numeric
  IOLIB(general_2)                       840d   845c   847c   870c   878c
numfiles
  LIBRARIAN                             3015d  3020c  3021c
numopcodes
  DGL_INQ(dgl_inq)                      6097d  6118d  6174c
numsign
  M68KSYS(ci)                            746d   755c
nvols
  CTABLE                                1212d  1537c  1538c  1541c  1543c  1551c  1552c  1555c  1557c  1571c  1572c  1575c  1577c  1659s
                                        1660s  1662s  1664s  1669s  1670s  1672s  1674s  1679s  1680s  1682s  1684s  1691s  1692s  1694s
                                        1696s  1698s  1699s  1701s  1703s  1709s  1710s  1712s  1714s  1715s  1717s  1719s  1725s  1726s
                                        1728s  1730s  1736s  1737s  1739s  1741s  1746s  1753s  1755s  1757s
  CTABLE(ctr)                            580d   586c   587c   588c   589c   590c   591c
nxt_edge
  DGL_POLY(dgl_poly)                   20430d 20664c 20671c 20672c 20674c 20675c 20677c 20678c 20679c 20681c 20683c 20685c
nxtlink
  INIT(isr)                              85d    89c    92c    94c    95c   104c   114d   118c   121c   123c   124c   127c   129c   130c
                                        132c
```

```
ob
  CS80(cs80)                          527d  531c
obj
  INITLOAD(loader)                   1118d 1120c
  LIBRARIAN                           131d  133c  137d  139c 2740d 2742c
object
  HEAPT(hpm)                           46d   47d  290c  295c  299c  311c  315c
  INITLOAD(loader)                   1108d 1501d 1531c 1534c 1535c 1536c 1539c 1545c 1546c 1549c
  LIBRARIAN                          2137d 2179c 2190c 2235c 2244c 2250c 2341c 2371c 2373c 2436c 2455c 2491c 2507c 2565c
                                     2566c 2567c 2568c 2569c 2576c 2577c 2578c 2579c 2580c 2592c 2595c 2596c 2627c 2654c
                                     2663c 2664c 2665c 2670c 2671c 2672c 2677c 2683c 2684c 2685c 2690c 2691c 2692c 2707c
                                     2709c 2738d 2773c 2777c 2780c 2786c
object0
  INITLOAD(loader)                   1501d 1545c 1549c
objectaddr
  LIBRARIAN                          2359d 2371c 2373c 2432c 2446c
obsize
  LIBRARIAN                          2241d 2244c
oc
  AMIGO(csamigo)                      245d  251d  252d  253d  254d  255d  256d  257d  258d  259d  260d  261d  262d  263d
                                      267c  278d  286d  287d
occurrences
  MINIT(cs80ir)              ***      539d
oct_10000
  MIUI                                398d  433d
odd_parity
  IOLIB(serial_3)                    2751c 2772c
  KERNEL(iodeclarations)     ***      402d
odd_unit
  CTABLE                             1314d 1318c 1319c 1326c
oddbytefiller
  SRM_DRV(srm)               ***      299d
off
  SRM_DRV                            1999d 2015c 2085d 2101c
  SRM_DRV(srm)                       1099d 1112d
offline
  AMIGO(amigodvr)            ***      566c
  CS80(cs80dvr)              ***     1359c
  CTABLE(ctr)                         483c  843c  864c
  F9885(f9885dvr)            ***      255c
  INIT(initunits)            ***     2215c
  INITLOAD                   ***     1689c
  INITLOAD(mini)             ***      482c
  INITLOAD(sysglobals)       ***      156d
  M68KSYS(ci)                         815c  960c  964c
  SRMAM(srmammodule)         ***      188c
  SRMDAM(srmdammodule)       ***     1644c
offset
  A804XDVR(a804xdvr)                  126d  128c  151d  154c
  LIBRARIAN                           511d  523c  524c  529c  530c  531c  532c  535c  539c 1805d 1831c
  MINIT(cs80ir)                       784d  796c
  SRMAM(srmammodule)                   57d   63c   67c   71c   75c   84c  104d  110c  114c  118c  122c  131c
  SRM_DRV                            2015c 2101c
  SRM_DRV(srm)                        716d  806d
offsetbytes
  LIBRARIAN                          2153d 2179c 2314c 2315c 2323c 2325c 2570c 2588c 2590c 2602c 2655c 2708c
ok
  CTABLE                             1219d 1829c 1831c 1837c 1838c 1839c 1840c
  LIFDAM(lifmodule)                   199d  253c  257c  260c  261c  273c 1063d 1136c
  SRMDAM(srmdammodule)                215d  309c  314c  321c  323c
  UCSD_DAM(ucsdmodule)                 71d   73c  102c  105c  121c
ok_to_config
  CS80(cs80dsr)                       604d  635d  760c
```

```
oksize
  ETU                                 487d  494c  513c  517c
old_a5
  GLE_TYPES(gle_types)       ***     1157d
old_a6
  GLE_TYPES(gle_types)       ***     1158d
old_dump_hook
  C_HOOK                     ***       57d
old_file_name_header
  SRM_DRV                    ***     1805c
  SRM_DRV(srm)               ***      520d
old_toggle_hook
  C_HOOK                     ***       56d
oldclrio
  HPHIL(hphil)                         69d  279c  369c
oldfilename
  LIBRARIAN                          3301d 3359c 3411c
oldfleof
  INIT(misc)                          537d  560c  574c
oldfpos
  INIT(misc)                          537d  574c  613c  618c
oldindex
  INITLOAD(loader)                   1503d 1524c 1526c 1528c 1530c 1531c
  LIBRARIAN                          1860d 1938c 1939c 1940c 1941c 1943c 1948c 1949c 2147d 2612c 2615c 2619c 2620c 2621c
                                     2624c 2628c 2714c 2715c
oldmask
  HPHIL(hphil)                        229d  233c  271c
oldnfns
  SRM_DRV                            1781d 1796c 1799c 1806c
  SRM_DRV(srm)               ***     1056d
oldnsaptr
  SRM_DRV                            1782d 1795c
  SRM_DRV(srm)               ***     1057d
oldpatch
  LIBRARIAN                          2368d 2424c 2429c 2438c 2476c
oldpath
  SRM_DRV                            1783d 1808c
  SRM_DRV(srm)               ***     1058d
oldpattern
  GLE_RGL(gle_ras_out)       ***     8054d
oldposit
  ETU                                 567d  577c  592c
oldpriority
  HPHIL(hphil)                        131d  138c  152c  158d  165c  175c  184d  187c  201c  208d  214c  218c
oldptr
  LIBRARIAN                          1473d 1482d 1541c 1543c 1544c 1547c 1548c 1549c 1559c 1560c 1570c 1581c 1593c 1606c
                                     1861d 1908c 1909c 1922c 1923c
oldrtpass
  SRM_DRV                            1785d 1809c
  SRM_DRV(srm)               ***     1060d
oldscode
  ETU                                 712d  760c  763c  777c  780c  783c
oldstart
  ETU                                1159d 1177c 1185c 1197c 1211c 1219c 1222c
oldstate
  LIBRARIAN                           308d 1216c 1228c
oldstatus5isr
  HPHIL(hphil)                         68d  316c  366c
oldtextrec
  INITLOAD(loader)                   1507d 1527c 1550c
  LIBRARIAN                          2154d 2342c 2622c
oldtracknumber
  MINIT(xminit)                      1405d 1583c 1688c
```

```
oldvmodname
  LIBRARIAN                        3299d  3361c  3416c  3419c
oldvmodnum
  LIBRARIAN                        3300d  3360c  3413c  3417c
oldwd
  SRM_DRV                          1784d  1807c
  SRM_DRV(srm)              ***    1059d
oldx
  ETU                               568d   588c   590c
oldy
  ETU                               568d   588c   590c
on
  DGL_RAS(dgl_raster)              17294d 17299c 17300c 17301c 17436d 17491c 17492c 17493c
on_same_controller
  TAPEBKUP                         1015d  1017c  1312c
on_same_medium
  CTABLE                    ***    1819c
  CTABLE(ctr)                       307d   875d   881c
  MIUI                               32d    35c   234c   514c
  TAPEBKUP                         1003d  1009c  1121c  1310c
one_parity
  IOLIB(serial_3)                  2754c  2774c
  KERNEL(iodeclarations)    ***     405d
one_with_lm
  GLE_RGL(gle_ras_out)             8082d  8229c
onheap
  INITLOAD(loader)                 1088d  1494d  1513c  1563c
op
  A804XDVR(a804xdvr)                255d   260c
  CRT(crt)                          502d   508c   544d   559c   574c   595c   604c
  EDRIVER(edriver)                   37d    75d   144c   219c   234c   248c   307d   315c
  ETU                                70d   713d   771c   773c   775c   791c   796c   813c   814c   827c   830c  1246c  1247c  1249c
                                   1251c  1253c  1255c  1257c  1272c  1274c
  GCRT(crtb)                        370d   379c   413d   425c
  HPHIL(hphil)                       73d   291d   293c
  INITLOAD(loader)          ***     865d
  LIBRARIAN                         188c   191c   193c  1505c  1523c  1574c  1579c  1614c  2384c  2388c  2395c  2413c
  MOUSE(mouse)                      126d   155c
  SYSDEVS(sysdevs)                   52d    64d   221d   303d   308d   378d   389d   477d   478c   480c   487d   489d   512d   525c
                                    534c   593d
op_description
  TAPEBKUP                         1182d  1190c  1194c  1205c
opc
  DGL_VARS(dgl_vars)               1123d  1128d
opcode
  AMIGO(csamigo)                     63d   267c
  DGL_HPGL(dgl_hpgl)               17016d 17041c 17061d 17125c 17128c 17131c 17140c 17153c 17156c 17162c 17165c 17168c
  DGL_INQ(dgl_inq)                  6008d  6084d  6115d  6122d  6123d  6124d  6125d  6126d  6127d  6128d  6129d  6130d  6131d  6132d
                                    6133d  6134d  6135d  6136d  6137d  6138d  6139d  6140d  6141d  6142d  6143d  6144d  6145d  6146d
                                    6147d  6148d  6149d  6150d  6151d  6152d  6153d  6154d  6155d  6156d  6157d  6158d  6159d  6160d
                                    6161d  6171c  6175c
  DGL_RAS(dgl_raster)              17347d 17426d 17442c 17447c 17452c 17457c 17462c 17477c 17486c 17498c 17511c 17517c 17522c 17532c
  F9885(f9885dvr)                    111d   125c   126c   127c   147c   148c   149c   150c   151c   164c   165c   169c   174c   178c
  GEN(dgl_gen)                      3046d  3136d  3157c  3158c
  LIB(dgl_lib)                     20030d 20056d 20063d 20386d 20390c 20391c 20394c 20750d 20763c 20764c 20768d 20781c 20782c
  LIBRARIAN                          429d   714c  1025c  1075c  1078c  1094c  1101c  1103c  1120c  1121c
opcode0
  LIBRARIAN                          648d  1026c
opcode4
  LIBRARIAN                          730d  1028c
opcode_ck
  DGL_INQ(dgl_inq)          ***     6171c
  GEN(dgl_gen)                      3046d  3136d  3161c  3162c  3163c
  LIB(dgl_lib)                     20763c 20781c
```

```
opcode_def
  DGL_INQ(dgl_inq)                 6113d  6115d
opcode_list
  DGL_INQ(dgl_inq)                 6121d  6175c  6182c
opcode_list_def
  DGL_INQ(dgl_inq)                 6118d  6121d
opcode_list_entry_def
  DGL_INQ(dgl_inq)                 6114d  6118d  6122d  6123d  6124d  6125d  6126d  6127d  6128d  6129d  6130d  6131d  6132d  6133d
                                   6134d  6135d  6136d  6137d  6138d  6139d  6140d  6141d  6142d  6143d  6144d  6145d  6146d  6147d
                                   6148d  6149d  6150d  6151d  6152d  6153d  6154d  6155d  6156d  6157d  6158d  6159d  6160d  6161d
opcode_type
  DGL_INQ(dgl_inq)                 6116d  6122d  6123d  6124d  6125d  6126d  6127d  6128d  6129d  6130d  6131d  6132d  6133d  6134d
                                   6135d  6136d  6137d  6138d  6139d  6140d  6141d  6142d  6143d  6144d  6145d  6146d  6147d  6148d
                                   6149d  6150d  6151d  6152d  6153d  6154d  6155d  6156d  6157d  6158d  6159d  6160d  6161d  6182c
opcode_type_def
  DGL_INQ(dgl_inq)                 6100d  6116d
opcodes
  DGL_POLY(dgl_poly)               20023d 20026d 20028d 20031d 20194d 20237c 20912d 20936c 20962c 21027d 21041c 21060c 21073d 21087c
                                   21105c 21117d 21131c 21136c 21149d 21162c 21167c
open_data
  SRMDAM(srmdammodule)              922c  1081c  1096c
  SRM_DRV(srm)              ***     225d
open_directory
  SRMDAM(srmdammodule)             1185c  1224c  1228c  1690c  1703c  1712c  1723c  1738c  1750c  1770c
  SRM_DRV(srm)              ***     227d
open_flag
  SRM_DRV(srm)              ***     276d
open_logical_eof
  SRMDAM(srmdammodule)              932c   933c
  SRM_DRV(srm)              ***     664d
open_old
  UCSD_DAM(ucsdmodule)              283d   636c
open_protected_directory
  SRMDAM(srmdammodule)      ***     882c
  SRM_DRV(srm)              ***     226d
open_type
  SRM_DRV                   ***    1933c
  SRM_DRV(srm)              ***     653d
opendir
  ETU                               349d   918c
  LIFDAM(lifmodule)                 279d   393c   533c   563c   592c   670c   968c  1066c  1085c  1107c
  UCSD_DAM(ucsdmodule)              510d   644c
opendirectory
  ETU                       ***     370c
  INITLOAD(sysglobals)      ***     136d
  LIFDAM(lifmodule)         ***    1153c
  SRMDAM(srmdammodule)             1648c  1688c
  UCSD_DAM(ucsdmodule)      ***     644c
openf
  LIFDAM(lifmodule)                 968d  1127c  1128c
openfib
  GLE_KNOB(gle_knob_in)            18033d 18050c 18064c
  UNITIO(uio)                        40d    93c    80c    96c   106c
openfile
  ETU                       ***     926c
  INIT(misc)                ***     451c
  INITLOAD(bootdammodule)   ***     693c
  INITLOAD(loader)          ***    1610c
  INITLOAD(sysglobals)      ***     135d
  LIFDAM(lifmodule)         ***    1127c
  SRMDAM(srmdammodule)             1654c  1720c
  UCSD_DAM(ucsdmodule)      ***     636c
```

```
openfileptr
  INITLOAD(sysglobals)        ***    240d
  SEGMENTER(asm)                     151c   166c
openfiles
  CTABLE                      ***   1852c
  INIT                        ***   2571c
  INIT(ldr)                         2278d  2294d
  M68KSYS(ci)                        892c  1127c
openin
  LIBRARIAN                         3075d  3364c  3412c  3635c
opening
  LIFDAM(lifmodule)                  678d   722c   728c   765c   787c   822c   824c   858c   859c   874c
openlinkf
  INIT(ldr)                   ***   2380c
  INITLOAD(loader)                  1083d  1136d  1613c
openlinkfile
  INIT(ldr)                         2273d  2372d  2435c  2445c
  LIBRARIAN                   ***   3086c
  SEGMENTER(asm)              ***    193c
openmod
  LIBRARIAN                         3245d  3466c  3642c
opennew
  LIFDAM(lifmodule)                  927d   973c   976c  1100c
openold
  LIFDAM(lifmodule)                  951d   982c  1092c
openout
  LIBRARIAN                         3118d  3626c  3645c
openpack
  SRMDAM(srmdammodule)               807c   839c   922c  1081c  1095c
  SRM_DRV                     ***   1902d
  SRM_DRV(srm)                ***   1077d
openparent
  SRMDAM(srmdammodule)               648d   805c   814c   830c
openparentdir
  INITLOAD(sysglobals)        ***    137d
  SRMDAM(srmdammodule)              1649c  1689c  1690c
opentype
  SRMDAM(srmdammodule)               647d   808c   840c
openunit
  GLE_KNOB(gle_knob_in)       ***  18041c
  INIT(misc)                         448c   454c
  INITLOAD(sysglobals)        ***    138d
  LIFDAM(lifmodule)           ***   1156c
  UCSD_DAM(ucsdmodule)        ***    648c
  UNITIO(uio)                 ***     48c
openvolume
  ETU                         ***    904c
  INIT(misc)                  ***    449c
  INITLOAD(sysglobals)        ***    137d
  LIFDAM(lifmodule)           ***   1157c
  UCSD_DAM(ucsdmodule)        ***    649c
operator_release_required
  CS80(cs80)                  ***    158d
  CS80(cs80dsr)               ***    709c
  TAPEBKUP(cs80tbdvr)                591c   823c
operator_request
  CS80(cs80)                         182d   311d   327d
  CS80(cs80dsr)               ***    716c
  TAPEBKUP(cs80tbdvr)                598c   830c
opmode
  LIBRARIAN                          434d   661c   663c   720c   724c  1127c
opn
  SRM_DRV                           1909d  1933c
  SRM_DRV(srm)                ***   1084d
```

```
opndtype
  LIBRARIAN                          546d   740d
opsize
  LIBRARIAN                          420d   568c   721c   893c   899c   913c  1083c  1087c
opsizetype
  LIBRARIAN                          418d   420d
opsproc
  HPHIL(hphil)                       194c   259c   261c   327c   340c   345c
  MOUSE(mouse)                ***    189c
  SYSDEVS(sysdevs)                   332d   339d
opt
  M68KSYS(ci)                        746d   751c
option
  INIT(fs)                           764d   768d   769d
  MINIT(cs80ir)                      559d   590d   618c
optionbyte
  MINIT(cs80ir)                      609d   618c
options
  CTABLE                      ***   1186d
  CTABLE(brstuff)             ***    928d
  CTABLE(ctr)                 ***    239d
  CTABLE(options)             ***     38d
  CTABLE(scanstuff)           ***   1035d
  MINIT(cs80ir)                      561d   649d   662d   668c
  TAPEBKUP(cs80tbr)                   25d    73d    86d    92c
options_byte
  CS80(cs80)                         346d   522d   531c
optype
  LIBRARIAN                          740d   758c   805c
order_values
  GEN(dgl_gen)                ***   3116d
org
  LIBRARIAN                         2197d  2217c
orient
  DGL_HPGL(dgl_hpgl)               17227d 17228d 17229d 17230d 17231d 17232d 17233d 17234d 17235d 17236d 17237d 17238d 17239d 17240d
                                   17241d 17242d
  DGL_INQ(dgl_inq)            ***   6077c
  DGL_POLY(dgl_poly)               20158c 20185c
  DGL_RAS(dgl_raster)              17562d 17563d 17564d 17565d 17566d 17567d 17568d 17569d 17570d 17571d 17572d 17573d 17574d 17575d
                                   17576d 17577d
  DGL_VARS(dgl_vars)          ***   1034d
origpathid
  SRMDAM(srmdammodule)               658d   713c   866c
ort
  CS80(cs80)                  ***    106d
os
  CTABLE(ctr)                        334d   335d   336d   688d   692c   693c   694c   700d   705c   706c   707c   720d   730c   731c
                                     732c   733c   737d   749c   750c   751c   752c   756d   768c
osinit
  M68KSYS(ci)                        855d   897c  1150c  1193c
osize
  LIBRARIAN                          564d   686c   703c   813c  1012c  1050c  1107c  1112c  1120c
other_card
  KERNEL(general_0)                 1066c  1101c  1115c  1152c
  KERNEL(iodeclarations)      ***    324d
other_info
  MINIT(cs80ir)               ***    546d
other_volume_check
  MIUT                               219d   547c
  TAPEBKUP                          1106d  1297c  1314c  1436c  1472c
otherlist
  HEAPT(hpm)                          57d    60d   113c   123d   144c   182c   202c   221c   241c   252c
out2type
  SYSDEVS(sysdevs)                    39d   168d   172d
```

```
out_bufptr
  DC_DRV(intdc)                          427c   531c
  DISCHPIB(dischpib)          ***        251c
  IOLIB(general_4)                      1405c  1416c  1672c
  KERNEL(general_0)           ***       1212c
  KERNEL(iodeclarations)                 498d   512d   526d
  PRINTER(prtdvr)             ***        291c
  SRM_DRV(srm)                          1335c  1339c
outblock
  LIBRARIAN                               70d   268c  1985c  2118c  2173c  2175c  2818c  2819c  2861c  2868c  2870c  2883c  2890c  2891c
                                        2892c  3108c  3144c  3162c
outdirectory
  LIBRARIAN                               67d  2854c  2859c  2888c  3106c  3111c  3112c  3129c  3152c  3153c
outdirectsize
  LIBRARIAN                               74d  3112c  3152c  3155c  3159c  3162c  3189c  3190c  3731c
outfile
  LIBRARIAN                               68d  1985c  2118c  2207c  2212c  2232c  2271c  2754c  2818c  2868c  2883c  3112c  3113c  3128c
                                        3141c  3142c  3149c  3511c  3689c
outfilename
  LIBRARIAN                               72d  3133c  3134c  3135c  3140c  3141c  3142c  3148c  3149c  3157c  3158c  3171c  3537c
outmodnum
  LIBRARIAN                               73d   267c  2852c  2853c  2854c  2859c  2886c  2887c  2888c  3109c  3110c  3514c  3532c  3636c
                                        3644c  3690c  3733c
outorder
  LIBRARIAN                             3317d  3462c  3496c
outopen
  LIBRARIAN                               48d   267c  3110c  3126c  3131c  3143c  3163c  3166c  3513c  3537c  3539c  3565c  3591c  3593c
                                        3607c  3623c  3625c  3627c  3629c  3631c  3633c  3634c  3640c  3643c  3648c  3649c  3650c  3653c
                                        3689c  3690c  3733c
outp
  KEYS(keys)                  ***        118c
  SYSDEVS(sysdevs)                       240d   529c   543c   549c   575c   640c
outposition
  ETU                                     86d   542c   543c   545c   547c   548c   550c   572c   577c   578c   592c   593c  1031c  1096c
                                        1101c  1126c
output_data
  DC_DRV(extdc)               ***        202d
  DC_DRV(intdc)                          325c   343c   445c
output_end
  DC_DRV(extdc)               ***        204d
  DC_DRV(intdc)               ***        450c
output_esc
  LIB(dgl_lib)                         20063d 20768d
output_escapei
  GLE_GEN(gle_gen)            ***       2054c
  GLE_HPGL(gle_hpgl_out)      ***       7548c
  GLE_RGL(gle_ras_out)        ***       8422c
  GLE_TYPES(gle_types)        ***       1053d
output_escapeo
  GLE_GEN(gle_gen)            ***       2048c
  GLE_HPGL(gle_hpgl_out)      ***       7547c
  GLE_RGL(gle_ras_out)        ***       8423c
  GLE_TYPES(gle_types)        ***       1054d
output_file
  GLE_FILE(gle_file_io)                 9013d  9058c  9067c  9077c  9078c
outrefindex
  LIBRARIAN                             2141d  2178c  2209c  2211c  2269c  2273c  2275c  2288c  2289c  2292c  2327c  2328c  2329c
outsize
  ETU                                    861d   909c   931c   995c  1013c  1015c  1032c  1127c
outstarta
  ETU                                     87d   548c   949c   954c   959c   961c   967c   969c   971c   974c   976c   978c   979c   985c
                                         986c   987c   991c   992c  1001c  1031c  1126c
outstate
  ETU                                    855d  1037c  1068c  1084c
```

```
overlap
  IOLIB(general_4)            ***       1510c
  KERNEL(iodeclarations)      ***        558d
overlap_dma
  DC_DRV(intdc)               ***        422c
  IOLIB(general_4)            ***       1511c
  KERNEL(iodeclarations)      ***        555d
overlap_fastest
  DISCHPIB(dischpib)          ***        470c
  IOLIB(general_4)            ***       1509c
  KERNEL(iodeclarations)      ***        557d
overlap_fhs
  IOLIB(general_4)            ***       1509c
  KERNEL(iodeclarations)      ***        556d
overlap_intr
  DC_DRV(intdc)               ***        437c
  IOLIB(general_4)            ***       1510c
  KERNEL(iodeclarations)      ***        554d
override
  MINIT(iramigo)                         192d   221d   234c
overrun
  AMIGO(amigodvr)                        811c   817c
  AMIGO(csamigo)              ***         76d
overwrite
  INIT(fs)                    ***        745d
  UCSD_DAM(ucsdmodule)                   301d   637c
overwritefile
  INITLOAD(sysglobals)        ***        135d
  LIFDAM(lifmodule)           ***       1129c
  SRMDAM(srmdammodule)                  1656c  1733c  1736c
  UCSD_DAM(ucsdmodule)        ***        637c
owner_id_type
  SRM_DRV(srm)                           310d   651d
p
  AMIGO(csamigo)              ***         99d
  ETU                                    111d   112c   123d   125c   191d   200c   213d   215c
  HEAPT(hpm)                              86d    89c    93d    96c    99d   102c   104c   110d   112c   193d   196c   204c   206c   210d
                                         264c   265c   266c   267c   270c
  INIT(fs)                               811d   812d   833d   835d
  INIT(ldr)                             2430c  2472c  2479c
  INITLOAD(loader)                       889d  1084d  1111d  1114c  1154c  1264c  1273c  1288c  1456c  1539c  1544c  1563c  1570c  1598c
                                        1635c  1642c
  INIT_OAD(mini)                         335d   342d   343d   344d   345d   346d   347d   348d   349d   350d   351d   352d   353d   354d
                                         355d   356d   357d   358d   359d   360d   361d   362d   363d   364d   365d   366d   367d   368d
                                         369d   370d   380c
  IOLIB(general_4)                      1260d  1559d  1575c  1576c  1586d  1602c  1603c
  KERNEL(general_0)                      855d   857c   858c   864c   865c   874d   876c   877c   883c   884c   894d   896c   897c   903c
                                         913d   915c   916c   922c   923c
  LIB(dgl_lib)                         20850d 20853c 20854c
  LIBRARIAN                              125d   127c  1296c  1298c  1393c  1644c  1666c  1698c  1784c  1903c  1906c  1967c  2075c  2207c
                                        2212c  2232c  2237c  2258c  2271c  2274c  2297c  2319c  2328c  2403c  2411c  2582c  2771c  2807c
                                        2809c  2814c  2817c  2818c  2840c  2867c  2868c  3095c  3721c  3723c
  MIUI                                    15d   180c   351c   474c
  SEGMENTER(asm)                          45d    53d   104d   106c   133c   175d   188c   189c   209c   230d   237c   243c   246c   268c
                                         279d   284c   285c   329c
  SEGMENTER(segmenter)        ***         36d
  SYSDEVS(sysdevs)                       516d   517c   519d   520c
```

```
p1
  CRT(crt)                          ***     54d
  DGL_POLY(dgl_poly)                      20442d 20467c 20470c 20472c 20478c 20482c
  DGL_VARS(dgl_vars)                ***   1137d
  GCRT(crtb)                        ***    101d
  GEN(dgl_gen)                            3014d  3016d  3097d  3102c  3103c  3106d  3112c  3113c  3116d  3123c  3125c  3130c
  GLE_UTLS(gle_utls)                      21013d 21014d 21019d 21024c 21028d 21033c
  INIT                              ***   2529c
  INITLOAD(sysglobals)              ***    270d
  LIBRARIAN                               1318d  1326c  1327c  1328c  1330c  1331c  1332c  1334c  1346d  1356c  1357c
p1x
  DGL_POLY(dgl_poly)                      20266d 20287c 20290c
p1y
  DGL_POLY(dgl_poly)                      20266d 20288c 20291c
p2
  CRT(crt)                          ***     54d
  DGL_POLY(dgl_poly)                      20442d 20468c 20470c 20472c 20479c 20482c
  DGL_VARS(dgl_vars)                ***   1137d
  F9885(f9885dvr)                   ***     61d
  GCRT(crtb)                        ***    101d
  GEN(dgl_gen)                            3014d  3016d  3097d  3102c  3106d  3112c  3116d  3123c  3126c  3131c
  GLE_UTLS(gle_utls)                      21013d 21014d 21019d 21024c 21025c 21028d 21033c 21034c
  INIT                              ***   2529c
  INIT(fs)                                817d   819d   820d   822d   825d   826d   828d   830d   833d   835d   836d   838d
  INITLOAD(sysglobals)              ***    270d
  IOLIB(general_2)                        836d   887c   916d   925c   926c
  MOREFSYS(mfs)                           21d    22d    533d   541c   544c   547c   560d   661c
p2x
  DGL_POLY(dgl_poly)                      20266d 20287c
p2y
  DGL_POLY(dgl_poly)                      20266d 20288c
p3
  DGL_VARS(dgl_vars)                ***   1137d
  INIT                              ***   2529c
  INITLOAD(sysglobals)              ***    270d
p3x
  DGL_POLY(dgl_poly)                      20267d 20290c 20293c 20300c 20310c
p3y
  DGL_POLY(dgl_poly)                      20267d 20291c 20294c 20301c 20311c
p4x
  DGL_POLY(dgl_poly)                      20267d 20293c 20300c 20310c
p4y
  DGL_POLY(dgl_poly)                      20267d 20294c 20301c 20311c
p_am
  INIT(initunits)                         2183d  2205c
p_ba
  CTABLE(ctr)                             425d   474c   497c
p_byteoffset
  CTABLE(ctr)                             428d   477c   499c
p_count
  DGL_POLY(dgl_poly)                      20410d 20710c 20728c 20729c 20730c 20738c 20739c 20740c 20745c 20749c 20757c 20758c 20765c 20766c
                                          20767c 20768c 20769c 20770c 20776c 20795c 20800c 20807c 20811c 20814c 20815c 20820c
p_dam
  INIT(initunits)                         2182d  2204c
p_devid
  CTABLE(ctr)                             429d   478c
p_du
  CTABLE(ctr)                             426d   475c   497c
p_dv
  CTABLE(ctr)                             427d   476c   497c
p_letter
  CTABLE(ctr)                             433d   482c   496c   498c
p_lif_leod
  MIUI                                    441d   481c
```

```
p_lif_vl
  MIUI                                    428d   471c
p_list_ptr
  DGL_POLY(dgl_poly)                      20409d 20568c 20571c 20574c 20575c 20577c 20578c 20579c 20694c 20729c 20730c 20739c 20740c 20749c
                                          20766c 20767c 20769c 20770c 20778c 20779c 20780c 20781c 20783c 20784c 20786c 20787c 20789c 20790c
                                          20792c 20793c 20797c 20798c 20822c 20823c
p_sc
  CTABLE(ctr)                             424d   473c   497c
p_uisblkd
  CTABLE(ctr)                             441d   486c   490c
  INIT(initunits)                         2200d  2218c  2222c
p_uisfixed
  CTABLE(ctr)                             438d   487c
p_uisinteractive
  CTABLE(ctr)                             435d   484c
  INIT(initunits)                         2194d  2216c
p_umaxbytes
  CTABLE(ctr)                             442d   492c
p_uvid
  CTABLE(ctr)                             430d   479c
  INIT(initunits)                         2190d  2212c
packed_pixel_3_plane_display
  GLE_RGL(gle_ras_out)              ***   8073d
packed_pixel_display
  GLE_RGL(gle_ras_out)              ***   8072d
packed_pixel_odd_byte_display
  GLE_RGL(gle_ras_out)              ***   8071d
packeddate
  A804XDVR(a804xdvr)                      245c   251c
  CLOCK(clock)                            54c    94c    109c   140c   150c
  SYSDEVS(sysdevs)                  ***   196d
packedtime
  A804XDVR(a804xdvr)                      243c   250c
  CLOCK(clock)                            95c    121c   139c   156c
  SYSDEVS(sysdevs)                        196d   460c
packet_ptr
  SRMAM(srmammodule)                      60c    107c
  SRMDAM(srmdammodule)                    338c   397c   422c   519c   582c   755c   917c   1101c  1103c  1445c  1475c  1485c
  SRM_DRV                                 1793c  1829c  1848c  1867c  1886c  1911c  1946c  1972c  2002c  2027c  2048c  2066c  2088c
  SRM_DRV(srm)                            993d   1216c  1379c  1380c  1381c  1382c  1383c  1385c  1387c  1388c  1390c  1391c  1392c  1423c
                                          1428c  1430c  1431c  1531c  1543c  1556c  1594c  1632c  1663c  1684c  1709c  1743c
packetin
  SRMAM(srmammodule)                      78c    85c    89c    125c   132c   136c
  SRM_DRV                                 1776c  1821c  1841c  1860c  1898c  1937c  1961c  1991c  2041c  2060c  2077c
  SRM_DRV(srm)                            1002d  1399d  1541c  1581c  1619c  1654c  1677c  1698c  1724c
packetout
  SRM_DRV                                 1775c  1820c  1840c  1859c  1877c  1897c  1936c  1960c  1990c  2018c  2040c  2059c  2076c  2107c
  SRM_DRV(srm)                            1001d  1371d  1539c  1580c  1618c  1653c  1676c  1697c  1723c
packetouttimeout
  SRM_DRV(srm)                            1121d  1378c
pactostr
  LIFDAM(lifmodule)                       90d    139c   268c
pad
  BUBBLES(bubble)                   ***   139d
  CS80(cs80)                              508d   515c
  CTABLE(brstuff)                   ***   961d
  CTABLE(ctr)                       ***   489c
  EDRIVER(edriver)                  ***   55d
  F9885(f9885dvr)                         61d    131c   201c
  INIT(initunits)                   ***   2221c
  INITLOAD                          ***   1695c
  INITLOAD(bootdammodule)           ***    733d
  INITLOAD(sysglobals)              ***    162d
```

```
pad1
  A804XDVR(a804xdvr)              ***     53d
  BAT(bat)                        ***     41d
pad3
  HPHIL(hphil)                    ***     61d
pad5
  HPHIL(hphil)                    ***     59d
pad6
  HPHIL(hphil)                    ***     59d
pad7to
  INITLOAD(sysglobals)            ***    290d
padding
  HPHIL(hphil)                    ***     49d
pageblocks
  LIBRARIAN                              31d  1366d  1394c
pagebuffer
  CONVERT(convert_text)                  35d    36d    40d    64c    73c    78c    82c    87d    96c   103c   107c   115c   117c
pagebuftype
  CONVERT(convert_text)                  33d    35d    36d    40d    87d    89d
pageeject
  LIBRARIAN                             241d   250c   325c  2937c  3701c
pagelines
  LIBRARIAN                              30d   250c
pagenum
  LIBRARIAN                              45d   236c   253c  2938c  2942c  3704c  3714c
pageptr
  CONVERT(convert_text)                  89d   117c
pages
  LIBRARIAN                            1369d  1388c  1389c
pagesize
  CONVERT(convert_text)                  31d    33d    60c    82c   100c   106c   114c   121c
  LIBRARIAN                            1366d  1382c  1388c  1391c  1392c  1399c  1403c
  UCSD_AM(ucsd_am)                       43d    52c   106c   218c   219c   237c
palette
  DGL_INQ(dgl_inq)                ***   6324c
  GLE_HPGL(gle_hpgl_out)               7622c  7629c  7638c  7654c  7664c  7668c  7670c  7686c
  GLE_TYPES(gle_types)            ***   1101d
paoc
  SRMDAM(srmdammodule)                   97d   102c   103c   106c   111d   113c   115c   664d   678c
paoc16tostr
  SRMDAM(srmdammodule)                   97d   404c   432c   541c   602c   728c   857c   860c
parameter
  CTABLE(ctr)                           374d   376c
  MINIT(cs80ir)                         605d   615c
parameter_bounds
  CS80(cs80)                      ***   139d
  CS80(cs80dsr)                   ***   673c
  TAPEBKUP(cs80tbdvr)                   553c   785c
parameter_field_owner
  CS80(cs80dsr)                         641d   654c   662c   754c
  TAPEBKUP(cs80tbdvr)                   521d   534c   542c   635c   644c   701d   762c   770c   867c   877c
parindex
  M68KSYS(ci)                           389d   452c   498c
parity_mode
  IOLIB(serial_3)                      2513d  2737d  2748c  2769c
parm
  MINIT(qminit)                        1034d  1041c  1049c
parm1
  DGL_HPGL(dgl_hpgl)              ***  17213d
  DGL_INQ(dgl_inq)                      6018d  6035d  6051c  6056c
  DGL_RAS(dgl_raster)                  17198d 17210c 17215c 17223c 17251d 17263c 17267c 17268c 17273c
  LIB(dgl_lib)                         20087d 20409d 20421c 20424c
```

```
parm2
  DGL_HPGL(dgl_hpgl)              ***  17214d
  DGL_INQ(dgl_inq)                      6019d  6036d  6052c  6056c
  DGL_RAS(dgl_raster)                  17199d 17210c 17216c 17224c 17252d 17264c 17267c 17269c 17274c
  LIB(dgl_lib)                         20088d 20410d 20422c 20424c
parm3
  DGL_HPGL(dgl_hpgl)              ***  17215d
  DGL_INQ(dgl_inq)                      6020d  6037d  6053c  6056c
  DGL_RAS(dgl_raster)                  17200d 17210c 17217c 17225c 17253d 17265c 17267c 17270c 17275c
  LIB(dgl_lib)                         20089d 20411d 20423c 20424c
parptr
  M68KSYS(ci)                            393d   433c   455c   456c   499c
parseoptparm
  SRMDAM(srmdammodule)                   194d   976c  1060c
part
  DGL_HPGL(dgl_hpgl)                   17248d 17290c
  DGL_RAS(dgl_raster)                  17552d 17609c
  ETU                                    491d   498c   499c   501c   502c   504c   507c   509c   511c   513c   517c   518c   521c   524c
                                         526c   534d   543c   549c   550c
  GLE_HPGLI(gle_hpgl_in)               18231d 18268c 18271c
partial_length
  CS80(cs80dvr)                         1220d  1257c  1258c  1268c  1269c  1270c  1272c  1295c  1296c  1297c  1299c  1302c  1303c  1307c
                                        1308c  1309c  1311c
partitioning_parameters
  CTABLE                                1536c  1549c  1566c  1820c
  CTABLE(ctr)                            310d   560d   565c   566c   567c   568c   569c   570c   571c   572c   573c
pass
  ETU                                     88d   570c  1019c  1030c  1066c  1116c  1118c
  SRMDAM(srmdammodule)                   201d   253c   268c   299c
  TAPEBKUP                              1225d  1343c  1374c  1407c
pass_control
  IOLIB(hpib_2)                         1786d  1896d
pass_rgb
  DGL_POLY(dgl_poly)                   20058d 20067c 20071c 20107d
passarray
  SRMDAM(srmdammodule)                    50d    53d    75d
passentry
  INIT(misc)                      ***   201d
  SRMDAM(srmdammodule)                    50d    54d    55d    56d    57d    58d    59d    60d    61d   459d   502d
passfailed
  ETU                                    532d   573c   582c   583c   595c   596c
passler
  SRMDAM(srmdammodule)                  1525d  1533c  1559c  1566c  1593c
passleng
  INITLOAD(sysglobals)                    17d    44d
  SRMDAM(srmdammodule)            ***   1671c
passtype
  ETU                                     43d   532d
  INIT(misc)                      ***   203d
  INITLOAD(sysglobals)                    44d   101d
  SRMDAM(srmdammodule)            ***   1632d
password
  F9885(f9885dvr)                         70d   124c   146c   163c
  SRMDAM(srmdammodule)                   128c   150c   155c   160c   189c   303c   484c   541c   768c   778c   859c   860c  1003c  1067c
                                        1085c  1094c  1122c  1142c  1145c  1158c  1163c  1192c  1301c  1397c  1400c  1419c
  SRM_DRV(srm)                           291d   316d   410d   440d   605d  1671c
password_index
  SRM_DRV(srm)                           398d  1605c
password_info
  SRMDAM(srmdammodule)            ***   539c
  SRM_DRV(srm)                    ***   409d
passwordarrayptr
  SRMDAM(srmdammodule)                    75d   525c  1620c
```

```
patchable
  INITLOAD(loader)                  923d  1461c
  LIBRARIAN                        1926c  2053c  2376c  2699c  2702c
patchaddr
  LIBRARIAN                        2360d  2427c  2432c  2441c  2442c  2446c  2450c  2453c  2464c
patchbase
  INITLOAD(loader)          ***    1009d
  LIBRARIAN                        1674c  1927c  1930c  2427c  2450c  2464c  2541c  2595c
patchbytes
  LIBRARIAN                          79d  3217c  3218c  3220c  3224c  3573c  3575c  3730c
patchdelta
  LIBRARIAN                        2366d  2432c  2433c  2436c  2446c  2448c  2455c  2458c  2467c
patchdescptr
  INITLOAD(loader)                  881d   915d   998d  1008d
  LIBRARIAN                        2127d  2365d
patchdescriptor
  INITLOAD(loader)                  881d   997d
  LIBRARIAN                 ***    2479c
patcherror
  LIBRARIAN                        2334d  2475c  2700c
patching
  LIBRARIAN                        2131d  2182c  2183c  2699c  2702c
patchlink
  INITLOAD(loader)          ***    1007d
  LIBRARIAN                        1678c  2472c  2606c  2607c  3226c
patchlist
  INITLOAD(loader)                  998d  1008d
  LIBRARIAN                        2428c  2444c  2480c  2481c  2484c  2544c  2593c  3226c
patchmod
  INITLOAD(loader)                 1005d  1185c  1306c  1340c  1436c
  LIBRARIAN                        1558c  1672c  1733c  1918c  2019c  2537c  3224c
patchmodptr
  LIBRARIAN                        2362d  2422c  2423c  2424c  2425c  2460c  2469c  2472c
patchptr
  LIBRARIAN                        2127d  2365d  2428c  2429c  2430c  2443c  2444c  2544c  2545c  2593c
patchref
  INITLOAD(loader)          ***     999d
  LIBRARIAN                        2434c  2440c  2485c  2498c  2545c  2551c  2556c  2557c  2582c  2585c
patchsize
  INITLOAD(loader)          ***    1010d
  LIBRARIAN                        1675c  1676c  1921c  2450c  2464c  2541c  2595c  2604c  3224c
patchstate
  LIBRARIAN                        2368d  2419c  2424c  2429c  2438c  2447c  2457c  2463c  2466c  2475c  2476c  2486c
path
  ETU                                57d   340c   374c   924c
  SRMDAM(srmdammodule)              656d   732c   737c   741c   807c   825c   839c   904d   922c  1267d  1291c  1296c  1302c
  SRM_DRV                          1905d  1925c  1968d  1986c
  SRM_DRV(srm)                     1008d  1016d  1024d  1044d  1080d  1092d  1550d  1576c  1588d  1614c  1626d  1647c  1731d  1759c
path1
  SRMDAM(srmdammodule)             1176d  1215d
path2
  SRMDAM(srmdammodule)             1179d  1218d
path_start_type
  SRMDAM(srmdammodule)              656d   904d  1176d  1179d  1215d  1218d  1267d
  SRM_DRV                          1783d  1788d  1905d  1968d
  SRM_DRV(srm)                      232d   270d  1008d  1016d  1024d  1044d  1058d  1063d  1080d  1092d  1515d  1550d  1588d  1626d
                                   1731d
path_type
  SRM_DRV(srm)                      270d  1523c
```

```
pathid
  ETU                                325c   340c   374c   924c
  INITLOAD(sysglobals)      ***     103d
  MIUI                      ***     530c
  SRMDAM(srmdammodule)              348d   352c   492c   530c   589c   713c   716c   740c   742c   744c   808c   821c   824c   834c
                                    840c   844c   850c   851c   866c   867c   889c   890c   893c   922c  1007c  1081c  1086c  1095c
                                   1123c  1129c  1134c  1143c  1146c  1202c  1203c  1205c  1235c  1236c  1239c  1240c  1257c  1302c
                                   1305c  1367c  1382c  1391c  1392c  1405c  1420c  1428c  1433c  1448c  1676c  1692c  1697c  1707c
                                   1716c  1728c  1744c  1749c  1754c  1756c  1769c  1774c  1776c  1803c
pattern_ert
  MINIT(cs80ir)                      567d   770d   814c
  MINIT(qminit)                     1187c  1216c  1247c
patternert
  MINIT(cs80ir)                      782d   794c
patterngenerate
  MINIT(xminit)                     1453d  1717c
pausekey
  KEYS(keys)                          69d   522c
pbits
  INIT(misc)                ***     202d
  SRMDAM(srmdammodule)                54d    55d    56d    57d    58d    59d    60d    61d   485c   543c
pbyte
  EDRIVER(edriver)                    97d   117c   161c   166c   170c   172c   177c   182c   186c   187c   199c   213c   215c   217c
                                    220c   223c   230c   232c   235c   238c   244c   246c   249c   252c   256c   260c   271c   272c
                                    273c   275c   285c   290c
pc
  LIBRARIAN                          309d   513c  1062c  1154c  1156c  1179c  1200c  1220c  1222c  1227c  1251c  1255c  1266c  1292c
                                    1294c  1303c  1304c  1308c
  LIFDAM(lifmodule)                   90d    93c    99d   105c
pclimit
  LIBRARIAN                         1271d  1304c  1308c
pcrange
  LIBRARIAN                          306d  1200c  1251c  1462c
pcrel
  LIBRARIAN                          143d   165c   175c   204c   206c   207c   510d   532c   535c   539c   601d   603c   606c
pcs
  SRMDAM(srmdammodule)              1340d  1398c  1405c
pctemp
  LIBRARIAN                          511d   516c   519c   521c   529c   533c
pd
  MINIT(cs80ir)                      685d   692c   693c   694c   695c
pdata
  MINIT(bminit)                       97d    99c   100c   101c
pdensity
  DGL_INQ(dgl_inq)                  6023d  6061d  6076c
  DGL_POLY(dgl_poly)               20016d 20166d 20180c 20184c
pdp
  INITLOAD(loader)          ***     915d
  LIBRARIAN                         2480c  2481c  2482c
pdptr
  MINIT(xminit)                     1411d  1565c  1668c  1672c  1673c  1698c  1703c
pdtable
  MINIT(xminit)                     1410d  1669c  1673c  1706c  1745c  1754c  1756c
pedge
  DGL_INQ(dgl_inq)                  6025d  6063d  6078c  6079c
  DGL_POLY(dgl_poly)               20018d 20168d 20179c 20186c
pen_down
  GLE_HPGLI(gle_hpgl_in)           18233d 18271c
pen_draw_mode
  GLE_RGL(gle_ras_out)              8050d  8513c
pen_number
  GLE_RGL(gle_ras_out)              8052d  8515c
```

```
penup
  GLE_SMARK(gle_smark)              5026d  5033d  5034d  5035d  5036d  5037d  5038d  5039d  5040d  5041d  5042c  5043d  5044d  5045d
                                    5046d  5047d  5048d  5049d  5050d  5051d  5052d  5053d  5054d  5055d  5056d  5057d  5058d  5059d
                                    5060d  5061d  5062d  5063d  5064d  5065d  5066d  5067d  5068d  5069d  5070c  5071d  5072d  5073d
                                    5074d  5075d  5076d  5077d  5078d  5079d  5162c
peof
  UCSD_DAM(ucsdmodule)               355d   365c   366c   368c
periodict
  A804XDVR(a804xdvr)                 271c   298c   319c
  SYSDEVS(sysdevs)          ***      213d
permanent
  INIT(ldr)                         2274d  2406d  2481c
  M68KSYS(ci)                        912d   917c   936c   937c   987d   990c  1001c
permittable
  INIT(initunits)           ***     2177c
  INIT(isr)                           92c   108c   129c   130c
  INITLOAD(sysglobals)      ***      275d
  M68KSYS(ci)               ***      296c
permissrlink
  A804XDVR(a804xdvr)        ***      386c
  BUBBLES(bubble)           ***      166c
  DC_DRV(init_dc)           ***      655c
  DI_DRV(init_discint)      ***      260c
  DMA_DRV(init_dma)                  266c   272c
  G_DRV(init_gpio)          ***      275c
  H_DRV(init_hpib)          ***      283c
  INIT(isr)                           45d    78d
  RS_DRV(init_rs)           ***      263c
pfname
  INIT(fs)                  ***      753d
phile
  MOREFSYS(mfs)                       30d    64c    75c   526c
phipor!
  MINIT(asmr)                       1332d  1336d  1343d  1347d
php
  INIT(ldr)                         2379c  2427c
  INITLOAD(loader)                   904d  1145c  1154c  1263c  1288c  1325c  1329c  1387c  1538c  1543c  1595c
  LIBRARIAN                         2963c  3736c
  SEGMENTER(asm)            ***      184c
phydata
  MINIT(hminit)                      286d   289c   295c
  MINIT(mminit)                      134d   139c
  MIUI                               403d   438d   473c   474c   475c   476c   477c
phyle
  INITLOAD(loader)                   874d   937d
  LIBRARIAN                 ***       68d
physical_cyln
  MINIT(cs80ir)             ***      532d
physical_data
  MINIT(bminit)                       60d    95d    97d
  MINIT(hminit)                      253d   283d   286d
  MINIT(midecs)             ***       26d
  MINIT(mminit)                      115d   132d   134d
  MINIT(qminit)                      883c   919d
  MIUI                               403d   438d
physical_head
  MINIT(cs80ir)             ***      533d
physical_sect
  MINIT(cs80ir)             ***      534d
physical_sector_size
  SRM_DRV(srm)              ***      613d
physical_size
  SRMDAM(srmdammodule)      ***      605c
  SRM_DRV(srm)              ***      286d
```

```
pindex
  CRT(crt)                  ***      231d
pindx
  SRMDAM(srmdammodule)               653d   673c   678c   679c   725c   769c   771c   777c   786c
pio_tmp_ptr
  BUBBLES(bubble)           ***      126d
  DC_DRV(intdc)             ***      411d
  F9885(f9885dvr)           ***       81d
  KERNEL(iodeclarations)             535d   540d   571d   673d
pisrib
  A804XDVR(a804xdvr)                  89d   115d
  BUBBLES(bubble)                     32d   124d   144d
  DC_DRV(extdc)             ***      199d
  DC_DRV(intdc)                      245d   305d
  DI_DRV(extdi)             ***      156d
  DMA_DRV(init_dma)                  169d   170d   182d   199d
  G_DRV(extg)               ***      173d
  H_DRV(exth)               ***      171d
  INIT(initunits)                   2142d  2167d
  INIT(isr)                           43d    50d    53d    56d    65d    83d    85d    86d   112d   114d   115d   137d
  INITLOAD(sysglobals)               201d   202d   210d   213d
  RS_DRV(rs)                ***      173d
pk_ptr
  SRM_DRV(srm)                       874d   993d
pl
  ETU                                295d   297c
  M68KSYS(ci)                         69d   563c   984d  1090c  1092c  1094c
plane1_addr
  GLE_RGL(gle_ras_out)      ***     8034d
plane1_offset
  GLE_RGL(gle_ras_out)      ***     8035d
plane2_offset
  GLE_RGL(gle_ras_out)      ***     8036d
plane3_offset
  GLE_RGL(gle_ras_out)      ***     8037d
plfirst
  M68KSYS(ci)                        985d  1062c  1090c  1092c  1111c
plinkp
  INIT(isr)                           86d    90c    94c   107c   115d   119c   123c   132c
pls
  MOREFSYS(mfs)                       33d   412c
pm6845addrreg
  CRT(crt)                           169d   265c   676c
pm6845cmdreg
  CRT(crt)                           170d   266c   677c
pmod
  LIBRARIAN                         3214d  3221c  3222c  3225c
pnsa
  SRM_DRV                           1782d  1787d  1795c  1796c  1904d  1913c  1967d  1974c
  SRM_DRV(srm)                       928d  1007d  1015d  1023d  1043d  1057d  1062d  1079d  1091d  1549d  1558c  1587d  1596c  1625d
                                    1634c  1635c  1730d  1745c  1747c
point
  ETU                                490d   506c   511c
  LIBRARIAN                           35d   196c  1644c  1903c  1985c  2118c  2237c  2274c  2297c  2299c  2319c  2403c  2411c
point_array
  DGL_POLY(dgl_poly)                20396d 20409d
point_count
  DGL_POLY(dgl_poly)                20926d 20956c 20964c 20967c 20971c 20982c
point_def
  DGL_POLY(dgl_poly)                20394d 20401d 20402d
point_def1
  DGL_POLY(dgl_poly)                20393d 20396d
point_ready
  GLE_HPGL1(gle_hpgl_in)            18241d 18268c
```

```
pointer
  HEAPT(hpm)                          61d    66d    71d    75d    86c    93d    99d   110d   135d   193d   210d   213d   216d   286d
                                    305d
  INITLOAD(loader)                  1098d  1330c  1490c  1563c
  SRMAM(srmammodule)          ***     47d
pointertrick
  HEAPT(hpm)                          62d    87d    94d
pollblock
  MOUSE(mouse)                        39d    64d
polling
  HPHIL(hphil)                        50d   324c   336c
pollisr
  A804XDVR(a804xdvr)                 113d   148c   392c
poly_entry_def
  DGL_HPGL(dgl_hpgl)                17223d 17227d 17228d 17229d 17230d 17231d 17232d 17233d 17234d 17235d 17236d 17237d 17238d 17239d
                                    17240d 17241d 17242d
  DGL_RAS(dgl_raster)               17547d 17562d 17563d 17564d 17565d 17566d 17567d 17568d 17569d 17570d 17571d 17572d 17573d 17574d
                                    17575d 17576d 17577d
  DGL_VARS(dgl_vars)                 1032d  1038d
poly_table_def
  DGL_VARS(dgl_vars)                 1038d  1039d  1343d
poly_table_def_space
  DGL_HPGL(dgl_hpgl)          ***   17283c
  DGL_RAS(dgl_raster)         ***   17653c
  DGL_VARS(dgl_vars)          ***    1343d
poly_table_ptr
  DGL_HPGL(dgl_hpgl)                17283c 17285c
  DGL_INQ(dgl_inq)            ***    6074c
  DGL_POLY(dgl_poly)                20153c 20182c
  DGL_RAS(dgl_raster)               17653c 17655c
  DGL_VARS(dgl_vars)          ***    1116d
poly_table_ptr_def
  DGL_VARS(dgl_vars)                 1039d  1116d
poly_table_size
  DGL_HPGL(dgl_hpgl)          ***   17284c
  DGL_RAS(dgl_raster)               17652c 17654c
  DGL_VARS(dgl_vars)                 1006d  1038d
polygon
  DGL_POLY(dgl_poly)                20029d 21147d
  GLE_GEN(gle_gen)            ***    2186c
  GLE_HPGL(gle_hpgl_out)      ***    7551c
  GLE_RGL(gle_ras_out)        ***    8419c
  GLE_TYPES(gle_types)        ***    1055d
polygon_dev_dep
  DGL_POLY(dgl_poly)                20024d 21071d
polygon_fill_factor
  DGL_POLY(dgl_poly)                20895c 21004c
  GLE_HPGL(gle_hpgl_out)      ***    7703c
  GLE_TYPES(gle_types)        ***    1098d
polygon_simulation
  DGL_POLY(dgl_poly)                20195d 20210c 20243c
polygon_solid_fill
  DGL_POLY(dgl_poly)                20892c 20998c
  GLE_HPGL(gle_hpgl_out)      ***    7704c
  GLE_RGL(gle_ras_out)        ***    8305c
  GLE_TYPES(gle_types)        ***    1099d
polygon_support
  DGL_INQ(dgl_inq)            ***    6402c
  DGL_POLY(dgl_poly)                21046c 21091c
  GLE_HPGL(gle_hpgl_out)      ***    7687c
  GLE_TYPES(gle_types)        ***    1096d
polyline
  LIB(dgl_lib)                      20047d 20692d
```

```
porient
  DGL_INQ(dgl_inq)                   6024d  6062d  6077c
  DGL_POLY(dgl_poly)                20017d 20167d 20181c 20185c
pos_absolute
  SRM_DRV(srm)                ***     238d
pos_seq
  SRM_DRV(srm)                ***     239d
posit
  ASCII(asciimodule)                  43d    49c    55d    59c    60c
  EDRIVER(edriver)                    40d    78d   115c   116c   117c   200c   256c   260c   275c   276c   290c   294c   310d   322c
position_type
  SRM_DRV                     ***    1943d
  SRM_DRV(srm)                        237d   680d  1087d
pospack
  SRM_DRV                     ***    1941d
  SRM_DRV(srm)                ***    1085d
possible_bytes_transferred
  AMIGO(amigodvr)                     797d   866c   867c   868c
possibly_correctable_data_error
  AMIGO(amigodvr)                     812c   834c   838c   872c
  AMIGO(csamigo)              ***      76d
power_fail
  CS80(cs80)                  ***     162d
  CS80(cs80dsr)                       682c   699c
  TAPEBKUP(cs80tbdvr)                 562c   579c   794c   811c
powerup
  INITLOAD                    ***    1779c
pp
  CTABLE                            1218d  1536c  1537c  1549c  1551c  1566c  1571c  1820c  1821c
  CTABLE(ctr)                         311d   578d   582c   583c   584c   586c   589c   590c
pp_type
  CTABLE                      ***    1218d
  CTABLE(ctr)                         310d   311d   560d   562d   578d
  CTABLE(options)                     133d   173d   175d   177d   179d   181d   183d   185d   187d
ppath
  INIT(fs)                    ***     753d
ppc_message
  IOLIB(hpib_2)                     1927c  1941c
  KERNEL(iodeclarations)      ***     302d
ppd_message
  IOLIB(hpib_2)               ***    1942c
  KERNEL(iodeclarations)      ***     313d
ppe_message
  IOLIB(hpib_2)               ***    1928c
  KERNEL(iodeclarations)      ***     312d
ppoll
  IOLIB(hpib_3)                     2031d  2080d  2088c
ppoll_configure
  IOLIB(hpib_2)                     1788d  1915d
ppoll_unconfigure
  IOLIB(hpib_2)                     1791d  1934d
pprb
  DISCHPIB(dischpib)                  258d   263c   264c   423d   428c   429c
ppsa
  SRM_DRV(srm)                        931d  1028d  1048d  1630d  1635c  1735d  1747c
ppu_message
  IOLIB(hpib_2)               ***    1945c
  KERNEL(iodeclarations)      ***     307d
predecr
  LIBRARIAN                           733d   740d   786c   900c
prefix
  CRT(crt)                            105d   151d
  GCRT(crtb)                  ***      84d
  SYSDEVS(sysdevs)            ***      98d
```

```
prefixed
  CRT(crt)                          106d   131d   152d
  GCRT(crtb)                         64d    85d
  SYSDEVS(sysdevs)                   84d    99d
premature_eoi
  DISCHPIB(dischpib)                349d   371c   400c
  TAPEBKUP(cs80tbr)                 271d   282c   288c   293c
prepunassem
  LIBRARIAN                         388d  1320c  1348c  1381c  1414c
present
  ETU                                51d   504c   509c   676c   680c   749c   751c   806c   952c  1182c
preserving_status_do
  TAPEBKUP                          1162d  1198c  1204c  1274c  1275c
preset_drive
  MINIT(cs80ir)                      562d   676d   697c
  MINIT(qminit)                 ***  1167c
presetdrive
  MINIT(cs80ir)                      689d   694c
prev
  HEAPT(hpm)                         216d   241c   246c   251c   253c
previd
  MIUI                                44d    89c    90c
previous
  INITLOAD(loader)                  1477d  1480c  1484c  1485c  1487c
previous_char_ptr
  PRINTER(prtdvr)                    68d   232c   255c   276c
previous_hs_completed
  PRINTER(prtdvr)                   149d   211c   218c   252c
previously_timed_out
  PRINTER(prtdvr)                    73d   169c   237c   242c   250c   283c   315c
prevno
  MIUI                               45d    89c
primary
  F9885(f9885dvr)                    56d   116d   147c   169c   174c
primary_dam
  CTABLE                            1497c  1502c  1514c  1531c  1540c  1554c  1574c  1638s  1639s  1643s  1644s  1648s  1649s  1654s
                                    1661s  1671s  1681s  1693s  1700s  1711s  1716s  1727s  1738s  1754s  1764s  1769s  1781s  1787s
  CTABLE(ctr)                        292d   509c
primary_lms_dam
  CTABLE(ctr)                        510c   514c
  CTABLE(options)               ***   62d
primarycommands
  F9885(f9885dvr)                    51d    56d
primarytype
  INITLOAD(loader)                   921d  1460c
  LIBRARIAN                          178c  1176c  1551c  1564c  1610c  1616c  1617c  1813c  1832c  1925c  2182c  2377c  2380c  2387c
                                    2389c  2398c  2451c  2696c  2791c
primitive_access
  SRM_DRV(srm)                  ***  170d
printbr
  ETU                                556d   589c   765c   786c   793c
printdirectentry
  LIBRARIAN                         2722d  3022c
printdirectory
  LIBRARIAN                         3014d  3632c
printepinfo
  ETU                                725d   766c   787c
printer
  CTABLE                        ***  1418c
  CTABLE(ctr)                   ***   257d
  CTABLE(scanstuff)             ***  1126c
printer_tm_name
  CTABLE(ctr)                        362d   682c
```

```
printeron
  LIBRARIAN                          51d   228c  2084c  2218c  2349c  2495c  2539c  2549c  2563c  2574c  2613c  2925c  2926c  2944c
                                    2947c  2985c  2997c  3525c  3728c
printerror
  INIT(ldr)                         2342c  2504c
  INIT(misc)                         210d   365d
  M68KSYS(ci)                        426c   463c   494c   506c   835c   850c   900c  1198c  1205c
  TAIL                          ***   89c
printheader
  LIBRARIAN                          221d   254c  3717c
printinstrword
  LIBRARIAN                          470d  1258c  1259c
printioerrmsg
  ETU                                 98d   270c   825c  1141c  1270c
printmessage
  LIBRARIAN                         2336d  2348c  2352c
printopen
  LIBRARIAN                          50d   227c  2943c  2944c  2945c  2947c  3523c  3727c
printprocboundary
  LIBRARIAN                         1163d  1202c
printsc
  ETU                                717d   764c   785c
printsockets
  ETU                                739d   767c   788c   807c
printunres
  INIT(ldr)                         2383d  2464c
printwhy
  INIT(ldr)                         2411d  2436c  2447c
priority
  BUBBLES(bubble)               ***  169c
proc
  DISCHPIB(dischpib)                 201d   202d   416d   432c   440d   462c
  INIT(isr)                           72c   103c   144c
  INITLOAD(sysglobals)          ***   209d
  SEGMENTER(asm)                      50d    53d   175d   230d
  TAPEBKUP                          1162d  1169c
proc_await_locator
  DGL_HPGLI(dgl_hpgli)          ***  18128c
  DGL_KNOB(dgl_knob)            ***  18174c
  DGL_VARS(dgl_vars)            ***   1141d
  LIB(dgl_lib)                  ***  20996c
proc_color
  DGL_HPGL(dgl_hpgl)            ***  17275c
  DGL_RHS(dgl_raster)           ***  17605c
  DGL_VARS(dgl_vars)            ***   1135d
  LIB(dgl_lib)                      20426c 20463c 20990c 21014c 21049c
proc_color_table
  DGL_HPGL(dgl_hpgl)            ***  17276c
  DGL_RHS(dgl_raster)           ***  17606c
  DGL_VARS(dgl_vars)            ***   1137d
  LIB(dgl_lib)                  ***  20424c
proc_input_esc
  DGL_HPGL(dgl_hpgl)            ***  17273c
  DGL_RHS(dgl_raster)           ***  17603c
  DGL_VARS(dgl_vars)            ***   1128d
  LIB(dgl_lib)                  ***  20764c
proc_linestyle
  DGL_HPGL(dgl_hpgl)            ***  17274c
  DGL_RHS(dgl_raster)           ***  17604c
  DGL_VARS(dgl_vars)            ***   1133d
  LIB(dgl_lib)                  ***  20483c
proc_name
  SEGMENTER(asm)                      58d   109d   278d
  SEGMENTER(segmenter)                22d    33d    35d
```

```
proc_output_esc
  DGL_HPGL(dgl_hpgl)          ***  17272c
  DGL_RAS(dgl_raster)         ***  17602c
  DGL_VARS(dgl_vars)          ***   1123d
  LIB(dgl_lib)                ***  20782c
proc_rec
  SEGMENTER(asm)                    114d   129c   258d   264c
proc_sample_locator
  DGL_HPGLI(dgl_hpgli)        ***  18129c
  DGL_KNOB(dgl_knob)          ***  18175c
  DGL_VARS(dgl_vars)          ***   1144d
  LIB(dgl_lib)                ***  20943c
proc_type
  CTABLE(scanstuff)                1059d  1070d  1071d  1072d  1073d  1074d  1076d  1077d  1078d  1099d
  TAPEBKUP                          959d  1162d
procentry
  INIT(isr)                          38d    45d    55d    60d    72c    78d   103c   136d   144c
process_errors
  AMIGO(amigodvr)                   790d   946c
processor
  CTABLE(scanstuff)           ***   1087d
proctype
  INITLOAD(sysglobals)        ***    198d
prod_num
  CS80(cs80dsr)                     788d   917c   919c   921c
producer
  INITLOAD(loader)            ***    943d
  LIBRARIAN                         335c   336c   346c  1962c
producername
  LIBRARIAN                         318d   337c   338c   339c   340c   341c   342c   343c   344c   345c   346c   348c
product_number
  MINIT(qminit)                     943d  1006c  1008c  1011c
  TAPEBKUP(cs80tbdvr)               462d   484c   486c   488c
progchain
  M68KSYS(ci)                        58d   104c  1066c
progname
  INIT(ldr)                   ***   2362c
  INITLOAD(loader)                 1018d  1437c
  M68KSYS(ci)                 ***    926c
programname
  MIUI                               46d    89c
progstateinfoaddr
  CRT(crt)                           91d   137d
  GCRT(crtb)                  ***    70d
  SYSDEVS(sysdevs)            ***    90d
prompresent
  INITLOAD(sysglobals)        ***    291d
prompt1
  M68KSYS(ci)                       972d  1092c
prompt2
  M68KSYS(ci)                       974d  1092c
promptforchar
  ETU                               295d   771c  1246c
promptread
  ETU                               191d   215c   245c   291c  1008c
prompttype
  ETU                                40d   111d   123d   191d   213d   219d   295d   351d
  M68KSYS(ci)                       970d   972d   974d   977d   979d   984d
promptyorn
  ETU                               213d   259c
protect_code_set
  SRMDAM(srmdammodule)        ***   1340d
  SRM_DRV(srm)                      314d   930d
```

```
protectcode_set_array
  SRMDAM(srmdammodule)              172d   198d   474d   970d  1032d
  SRM_DRV(srm)                      930d   931d
proximity
  GLE_HPGLI(gle_hpgl_in)      ***  18235d
prtdvr
  PRINTER(prtdvr)             ***     32d
prtio
  PRINTER(prtdvr)                    40d    47d
psa
  SRMDAM(srmdammodule)              172d   189c   198d   301c   310c   474d   482c   490c   493c   970d   976c  1032d  1060c  1061c
                                   1065c  1072c  1087c  1124c  1135c  1147c
psaptr
  SRM_DRV(srm)                     1028d  1048d  1630d  1635c  1735d  1747c
psimask
  A804XDVR(a804xdvr)          ***    385c
  SYSDEVS(sysdevs)            ***    166d
psize
  UCSD_DAM(ucsdmodule)               570d   599c   603c   608c   611c
pt
  SRM_DRV(srm)                     1515d  1523c
ptix
  DGL_POLY(dgl_poly)              20443d 20462c 20467c 20473c 20486c 20488c 20495c 20505c
ptiy
  DGL_POLY(dgl_poly)              20443d 20463c 20467c 20474c 20489c 20498c 20506c
pt2x
  DGL_POLY(dgl_poly)              20444d 20464c 20468c 20473c 20486c 20490c 20497c 20507c
pt2y
  DGL_POLY(dgl_poly)              20444d 20465c 20468c 20474c 20491c 20498c 20508c
pt_1_index
  DGL_POLY(dgl_poly)              20433d 20462c 20463c
pt_2_index
  DGL_POLY(dgl_poly)              20434d 20464c 20465c
ptp
  INITLOAD(loader)            ***    913d
  LIBRARIAN                        1572c  1691c  1692c
ptr
  CTABLE(ctr)                        385d   394c   395c   397c   398c   399c   400c   403c
  DGL_C_OUT(dgl_confg_out)        11030d 11050d 11056c 11094c 11097c 11101c 11115c 11120c 11123c 11128c
  EPROMS(eproms)                     34d    86c    88c    89c
  GCRT(crtb)                        529d   538c   541c   543c
  HEAPT(hpm)                        286d   298c   299c   305d   311c   312c   313c
  INITLOAD(loader)                 1226d  1235c  1568d  1570c  1571c  1573c  1574c
  LIBRARIAN                        1368d  1395c  1397c  1399c  1400c  1402c  1403c  1862d  1909c  1910c  1949c  1950c  2736d  2753c
                                   2754c  2759c  2762c  2765c  2801c  2802c  2806c  2810c
  SEGMENTER(asm)                    112d   122c   123c   125c   126c   127c   128c   135c
  UCSD_AM(ucsd_am)                   44d   186c   189c   214c   225c   256c
ptr1
  INITLOAD(loader)                 1422d  1431c  1432c  1453c  1455c  1456c  1457c  1465c
ptr2
  INITLOAD(loader)                 1422d  1456c  1458c  1459c  1464c  1465c
ptrmp
  SRM_DRV(srm)                      823d   875d
ptrrareyoualive
  SRM_DRV(srm)                      825d   877d
ptrrcat
  SRM_DRV(srm)                      827d   879d
ptrrcatpass
  SRM_DRV(srm)                      829d   881d
ptrrchangeprotect
  SRM_DRV(srm)                      831d   883d
ptrrchangevolume
  SRM_DRV(srm)                      833d   885d
```

```
ptrrclose
  SRM_DRV(srm)                    835d   887d
ptrrcopy
  SRM_DRV(srm)                    837d   889d
ptrrcreatefile
  SRM_DRV(srm)                    839d   891d
ptrrcreatelink
  SRM_DRV(srm)                    841d   893d
ptrrexchange
  SRM_DRV(srm)                    843d   895d
ptrrfileinfo
  SRM_DRV(srm)                    845d   897d
ptrrgangclean
  SRM_DRV(srm)                    847d   899d
ptrrhead
  SRM_DRV(srm)                    872d   923d
ptrrinit
  SRM_DRV(srm)                    849d   901d
ptrrlock
  SRM_DRV(srm)                    851d   903d
ptrropen
  SRM_DRV(srm)                    853d   905d
ptrrpos
  SRM_DRV(srm)                    855d   907d
ptrrpurge
  SRM_DRV(srm)                    857d   909d
ptrrread
  SRM_DRV(srm)                    859d   911d
ptrrsetdate
  SRM_DRV(srm)                    861d   913d
ptrrseteof
  SRM_DRV(srm)                    863d   915d
ptrrunlock
  SRM_DRV(srm)                    865d   917d
ptrrvol
  SRM_DRV(srm)                    867d   919d
ptrrwrite
  SRM_DRV(srm)                    869d   921d
ptrsareyoualive
  SRM_DRV(srm)                    824d   876d
ptrscat
  SRM_DRV(srm)                    826d   878d
ptrscatpass
  SRM_DRV(srm)                    828d   880d
ptrschangeprotect
  SRM_DRV(srm)                    830d   882d
ptrschangevolume
  SRM_DRV(srm)                    832d   884d
ptrsclose
  SRM_DRV(srm)                    834d   886d
ptrscopy
  SRM_DRV(srm)                    836d   888d
ptrscreatefile
  SRM_DRV(srm)                    838d   890d
ptrscreatelink
  SRM_DRV(srm)                    840d   892d
ptrsexchange
  SRM_DRV(srm)                    842d   894d
ptrsfileinfo
  SRM_DRV(srm)                    844d   896d
ptrsgangclean
  SRM_DRV(srm)                    846d   898d
ptrshead
  SRM_DRV(srm)                    871d   922d

ptrsinit
  SRM_DRV(srm)                    848d   900d
ptrslock
  SRM_DRV(srm)                    850d   902d
ptrsopen
  SRM_DRV(srm)                    852d   904d
ptrspos
  SRM_DRV(srm)                    854d   906d
ptrspurge
  SRM_DRV(srm)                    856d   908d
ptrsread
  SRM_DRV(srm)                    858d   910d
ptrssetdate
  SRM_DRV(srm)                    860d   912d
ptrsseteof
  SRM_DRV(srm)                    862d   914d
ptrsunlock
  SRM_DRV(srm)                    864d   916d
ptrsvol
  SRM_DRV(srm)                    866d   918d
ptrswrite
  SRM_DRV(srm)                    868d   920d
ptrtable
  INITLOAD(loader)                880d  1045d
ptrtableptr
  INITLOAD(loader)                880d   913d
purge
  INIT(fs)                 ***    744d
purge_old_link
  SRM_DRV                  ***   1817c
  SRM_DRV(srm)             ***    522d
purgef
  LIFDAM(lifmodule)               662d   671c   998c  1131c
purgefile
  INIT(misc)               ***    434c
  INITLOAD(sysglobals)     ***    135d
  LIFDAM(lifmodule)        ***   1131c
  SRMDAM(srmdammodule)            1664c  1783c
  UCSD_DAM(ucsdmodule)     ***    638c
purgeit
  UCSD_DAM(ucsdmodule)            397d   638c
purgename
  INITLOAD(sysglobals)     ***    134d
  LIFDAM(lifmodule)        ***   1148c
  SRMDAM(srmdammodule)            1661c  1768c
  UCSD_DAM(ucsdmodule)     ***    639c
purgeold
  SRM_DRV                         1791d  1817c
  SRM_DRV(srm)             ***   1066d
  UCSD_DAM(ucsdmodule)             428d   639c
purgepack
  SRMDAM(srmdammodule)            1143c  1257c  1302c  1367c
  SRM_DRV                  ***   1965d
  SRM_DRV(srm)             ***   1089d
putbuffer
  UCSD_AM(ucsd_am)                 68d   108c   109c   119c   143c   205c
putchar
  UCSD_AM(ucsd_am)                113d   127c   129c
putenviron
  UCSD_AM(ucsd_am)                104d   174c   185c
putmenu
  ETU                      ***   1148d
putref
  LIBRARIAN                       2308d  2589c  2703c
```

```
putsize
  HEAPT(hpm)                          110d   181c   204c
putstatus
  CRT(crt)                  ***       535c
  GCRT(crtb)                ***       406c
  SYSDEVS(sysdevs)                     51d   582c
pvname
  INIT(fs)                  ***       753d
pword
  INIT(misc)                ***       203d
  SRMDAM(srmdammodule)                 54d    55d    56d    57d    58d    59d    60d    61d   484c   541c
q
  MIUI                                 15d   183c   191c   202c   335c   352c   476c
qcertify
  TAPEBKUP                           1247c  1483c
  TAPEBKUP(cs80tbdvr)                 329d   411d
qconfigure
  TAPEBKUP                  ***      1232c
  TAPEBKUP(cs80tbdvr)                 323d   387d
qcontrollercopy
  TAPEBKUP                  ***      1266c
  TAPEBKUP(cs80tbdvr)                 335d   693d
qcopy_start_address
  TAPEBKUP                           1239c  1397c
  TAPEBKUP(cs80tbdvr)                 339d   929d
qdescribe
  TAPEBKUP                  ***      1248c
  TAPEBKUP(cs80tbdvr)                 313d   429d   472c
qdevicename
  MINIT(qminit)                       881d   932d  1029c
  MIUI                      ***       192c
  TAPEBKUP                  ***      1091c
  TAPEBKUP(cs80tbdvr)                 315d   452d   504c
qgetinitparms
  MINIT(qminit)                       884d  1056d
  MIUI                      ***       336c
qintdata
  MINIT(qminit)                       882d   906d   910c   912c   913c   914c
  MIUI                      ***       337c
qm
  KEYS(keys)                           66d    67c
qminit
  MINIT(qminit)             ***       875d
  MIUI                      ***        12d
qminitialize
  MINIT(qminit)                       885d  1098d
  MIUI                      ***       352c
qphydata
  MINIT(qminit)                       883d   919d   923c   924c   926c   927c
  MIUI                      ***       476c
qs
  TAPEBKUP(cs80tbr)                   272d   292c   293c   295c
qstat
  CS80(cs80)                          356d   365d   377c   397c   441c   453c   469c   481c   496c   518c   534c   552c
  CS80(cs80dvr)             ***      1126c
  MINIT(cs80ir)                       621c   645c   672c   697c   724c   766c   814c   843c   868c
  TAPEBKUP(cs80tbr)                    69c    96c   129c   152c   175c   205c   251c   292c
qstat_byte
  CS80(cs80)                          371d   376c   377c
qtapesetup
  TAPEBKUP                  ***      1256c
  TAPEBKUP(cs80tbdvr)                 322d   364d
quick
  LIBRARIAN                           990d  1018c  1050c
```

```
quit
  LIBRARIAN                           264d  1458c  3647c
quittype
  LIBRARIAN                           298d  1455c  1457c  1469c
qunload
  TAPEBKUP                           1369c  1372c  1452c  1486c
  TAPEBKUP(cs80tbdvr)                 325d   911d
qverify
  TAPEBKUP                           1286c  1417c  1449c
  TAPEBKUP(cs80tbdvr)                 320d   509d
qwritefilemark
  TAPEBKUP                  ***      1269c
  TAPEBKUP(cs80tbdvr)                 324d   393d
r
  AMIGO(csamigo)                      107d   321c
  DGL_RAS(dgl_raster)               17067d 17105c
  GEN(dgl_gen)                        3048d  3049d  3166d  3181c  3182c  3183c  3184c  3185c  3186c  3190d  3219c  3220c  3227c  3231c
                                      3244c
  GLE_KNOB(gle_knob_in)             18047d 18053c 18054c
  GLE_RGL(gle_ras_out)                8155d  8185c  8189c  8193c  8197c  8201c
  IOLIB(serial_3)                     2532d  2536c  2537c  2538c  2539c  2540c  2541c  2542c  2543c  2544c  2545c  2546c  2547c  2548c
                                      2549c  2550c
  LIB(dgl_lib)                      20793d 20803c 20804c 20805c
  LIBRARIAN                           2358d  2410c  2479c  2480c  2481c  2482c  2491c  2492c  2493c  2494c  2502c
  MOREFSYS(mfs)                         22d   559d   661c
  UNITIO(uio)                           56d    62c    63c    65c    77d    83c    84c    86c
r3
  F9885(f9885dvr)                     121c   183c
r7
  F9885(f9885dvr)                     122c   193c   197c
rad_angle
  DGL_POLY(dgl_poly)                20879d 20899c 20900c 20901c 20929d 20993c 20994c 20995c
ram
  CTABLE(ctr)                         258d   639c   838c
ramaddress
  SRMAM(srmammodule)                   53d    67c    69c    75c    77c    84c   100d   114c   116c   122c   124c   131c
ramdriver
  M68KSYS(ci)                         731d   812c
random_access
  SRMAM(srmammodule)                   62c   109c
  SRM_DRV(srm)              ***       168d
range
  A804XDVR(a804xdvr)                  317c   319c   320c
  SYSDEVS(sysdevs)          ***       219d
rangetype
  LIBRARIAN                           306d  1199c  1200c  1201c  1250c  1251c  1252c  1460c  1461c  1462c  1463c
rareyoualive
  SRM_DRV(srm)                        877d  1543c
raster_byte
  GLE_RGL(gle_ras_out)                8012d  8013d  8014d  8098d
raster_code_space
  GLE_RGL(gle_ras_out)                8013d  8063d
raster_color
  DGL_RAS(dgl_raster)               17168d 17605c
raster_color_table
  DGL_RAS(dgl_raster)               17197d 17606c 17629c 17630c 17634c 17638c 17643c
raster_device_rec
  DGL_C_OUT(dgl_confg_out)  ***     11032d
  GLE_RGL(gle_ras_out)                8024d  8025d
raster_device_rec_ptr
  DGL_C_OUT(dgl_confg_out)          11050d 11059c 11157d
  DGL_INQ(dgl_inq)                   6198c  6214c
  DGL_RAS(dgl_raster)               17186c 17441c 17592c
  GLE_RGL(gle_ras_out)                8024d  8181c  8231c  8296c  8326c  8381c
```

```
raster_device_rec_space
  DGL_C_OUT(dgl_config_out)          11032d 11280c
raster_input_esc
  DGL_RAS(dgl_raster)                17347d 17603c
raster_linestyle
  DGL_RAS(dgl_raster)                17045d 17604c
raster_output_esc
  DGL_RAS(dgl_raster)                17426d 17602c
raster_patterns
  DGL_RAS(dgl_raster)          ***   17060c
  DGL_VARS(dgl_vars)           ***    1306d
rate
  GLE_KNOB(gle_knob_in)              18113d 18135c 18141c 18142c 18143c 18144c 18145c 18146c 18147c 18148c 18149c 18170c 18171c
  IOLIB(serial_3)                     2504d  2530d  2563c  2575c
  SYSDEVS(sysdevs)                     369d   421d   424c
rawait_blanking
  GLE_RGL(gle_ras_out)                8344c  8410c
rawmode
  HPHIL(hphil)                         136c   151c   163c   174c   361c
  SYSDEVS(sysdevs)             ***     345d
rawshift
  HPHIL(hphil)                         129d   238c   294c
rawshiftop
  HPHIL(hphil)                 ***     294c
  SYSDEVS(sysdevs)             ***     306d
rb
  LIBRARIAN                            442d   502c   506c
rba
  CS80(cs80)                   ***     225d
rblock
  INIT(fs)                     ***     790d
rbs
  CS80(cs80)                   ***     103d
rcat
  SRMDAM(srmdammodule)         ***     582c
  SRM_DRV(srm)                 ***     879d
rcatpass
  SRMDAM(srmdammodule)         ***     519c
  SRM_DRV(srm)                 ***     881d
rchangeprotect
  SRM_DRV(srm)                 ***     883d
rchangevolume
  SRM_DRV(srm)                 ***     885d
rclear
  GLE_RGL(gle_ras_out)         ***    8398c
rclose
  SRM_DRV(srm)                 ***     887d
rcopy
  SRM_DRV(srm)                 ***     889d
rcount
  LIBRARIAN                            151d   163c   180c   188c   189c   208c   209c   210c   211c
rcreatefile
  SRM_DRV(srm)                 ***     891d
rcreatelink
  SRM_DRV(srm)                 ***     893d
rcursor
  GLE_RGL(gle_ras_out)         ***    8417c
rdefine_color_map
  GLE_RGL(gle_ras_out)                8309d  8420c
rdefine_drawing_mode
  GLE_RGL(gle_ras_out)         ***    8418c
rdraw
  GLE_RGL(gle_ras_out)         ***    8397c

rdummy_proc
  GLE_RGL(gle_ras_out)                8290d  8411c  8421c  8422c  8423c  8424c  8478c  8482c
read_de_log_micro_op
  MINIT(cs80ir)                        739d   752c
read_de_log_micro_op_type
  MINIT(cs80ir)                        736d   739d
read_intr_mask
  HPHIL(hphil)                         225d   233c
read_operation
  AMIGO(amigodvr)                      621c   724c   731c   896c
  CS80(cs80dvr)                       1106c  1107c
  DISCHPIB(bkgnd)              ***      44d
  DISCHPIB(dischpib)           ***     450c
readblock
  F9885(f9885dvr)                       51d    57d   169c
readblocks
  INITLOAD(loader)                    1118d  1145c  1154c  1263c  1288c  1325c  1329c  1538c  1543c
  LIBRARIAN                            137d  1296c  1299c  1393c  2258c  2299c  2771c  2817c  2867c
readbuffer
  IOLIB(general_4)                    1235d  1557d  1631c
readbuffer_string
  IOLIB(general_4)                    1239d  1611d
readbusy
  HPHIL(hphil)                          40d   149c
readbytes
  AMIGO(amigodvr)                      592c   621c
  ASCII(asciimodule)                    50c   268c   278c   286c
  BUBBLES(bubble)                       71c    81c
  CRT(crt)                             435c   439c
  CS80(cs80dvr)                       1106c  1193c  1262c  1267c  1278c  1282c  1385c
  EPROMS(eproms)               ***      50c
  ETU                                 458c  1052c
  F9885(f9885dvr)                     167c   277c
  GCRT(crtb)                          305c   309c
  GLE_KNOB(gle_knob_in)        ***   18053c
  INIT(misc)                          269c   458c   489c   493c   523c   564c   608c   612c   629c   645c   656c   674c   682c
  INITLOAD(bootdammodule)      ***     755c
  INITLOAD(loader)             ***    1120c
  INITLOAD(mini)                      425c   438c   505c
  INITLOAD(sysglobals)         ***      66d
  KEYS(keys)                   ***      98c
  LIBRARIAN                           139c  2242d  2246c  2247c  2252c  2253c  2254c  2256c  2258c  2259c  2260c  2261c  2281d  2289c
                                     2290c  2291c  2292c  2293c  2294c  2295c  2297c  2298c  2300c  2301c  2302c  2304c
  LIFDAM(lifmodule)                   251c   444c
  M68KSYS(ci)                         346c   739c
  SRMAM(srmammodule)           ***    192c
  SWVOL                        ***     24c
  UCSD_AM(ucsd_am)                    165c   212c
  UCSD_DAM(ucsdmodule)         ***     604c
  UNITIO(uio)                  ***      83c
readchar
  IOLIB(general_1)                    238d   282d
  IOLIB(hpib_3)                ***   2101c
readchars
  ASCII(asciimodule)                  175d   278c   286c
readcheck
  ETU                                 137d   202c   298c
readctrl
  HPHIL(hphil)                         37d   140c   167c
readdrivetables
  TAPEBKUP(cs80tbr)                   268d   276c
readint
  LIBRARIAN                           283d   291c  1426c  1429c  3181c  3197c  3203c  3217c
```

```
readlogs
  MINIT(cs80ir)                      746d   752c
readlstat
  HPHIL(hphil)                       206d   216c
readmonthabbrev
  M68KSYS(ci)                        230d   247c   250c   265c
readnumber
  ETU                                147d   778c   801c   979c   987c
  IOLIB(general_2)                   797d   832d  1017c
readnumberln
  IOLIB(general_2)                   810d  1011d
readnumericfield
  M68KSYS(ci)                        208d   225c   227c   261c   266c   272c   276c   278c
readonly
  INIT(fs)                    ***    745d
readsize
  LIBRARIAN                         1370d  1391c  1392c  1393c
readstring
  ASCII(asciimodule)                 212d   279c   287c
  IOLIB(general_2)                   801d   935d
readstring_until
  IOLIB(general_2)                   804d   964d
readt
  A804XDVR(a804xdvr)          ***    287c
  SYSDEVS(sysdevs)                   214d   478c
readtoeol
  ASCII(asciimodule)                  27c   268c   270c
  CRT(crt)                           417c   423c
  ETU                                436c   479c
  GCRT(crtb)                         287c   293c
  INIT(misc)                  ***    584c
  INITLOAD(sysglobals)        ***     66d
  KEYS(keys)                          97c   101c   122c   129c
  M68KSYS(ci)                 ***    363c
  UCSD_AM(ucsd_am)                   163c   211c   223c   252c   259c
readuntil
  IOLIB(general_2)                   819d  1019c  1050d
readword
  IOLIB(general_1)                   242d   304d
readwrite
  INIT(fs)                    ***    745d
ready
  A804XDVR(a804xdvr)                  55d   120c   121c
  BAT(bat)                            43d    64c
  GLE_HPGLI(gle_hpgl_in)      ***  18239d
real_denom
  DGL_POLY(dgl_poly)                20282d 20307c 20309c
real_factor
  DGL_POLY(dgl_poly)                20283d 20309c 20310c 20311c
real_format
  DGL_POLY(dgl_poly)                20908d 20972c
real_num
  DGL_POLY(dgl_poly)                20281d 20308c 20309c
real_proc
  DC_DRV(intdc)               ***    539c
  DISCHPIB(dischpib)          ***    481c
  DMA_DRV(init_dma)                  192c   209c
  KERNEL(iodeclarations)      ***    466d
rec
  GLE_HPIB(gle_hpib_io)       ***  10061d
  IOLIB(hpib_1)               ***    430d
recalibrate
  AMIGO(amigodvr)             ***    824c
  AMIGO(csamigo)                     147d   296d
```

```
recalibrate_cmd
  AMIGO(csamigo)                      52d   300c
recalibrate_cmd_buf
  AMIGO(csamigo)                     298d   300c
recbytes
  INIT(fs)                    ***    763d
receive_data_sec
  MINIT(iramigo)                     213d   218c
recombine
  HEAPT(hpm)                         132d   266c   282c
reconfiguration_needed
  CS80(cs80dsr)                      642d   655c   704c   735c   760c
  TAPEBKUP(cs80tbdvr)                522d   535c   586c   617c   650c   702d   763c   818c   849c   883c
record_addr
  AMIGO(csamigo)                     150d   151d   339d   347c   348c   379d   387c   392d   400c
  F9885(f9885dvr)                    105d   149c   150c   211c
record_mode
  SRM_DRV(srm)                       279d   496d   660d  1763c
recordnum
  LIBRARIAN                         2738d  2764c  2767c  2787c
records_per_medium
  AMIGO(amigodvr)             ***    921c
  AMIGO(csamigo)                     141d   219d   225c
recoverable_data
  CS80(cs80)                         193d   318d
  CS80(cs80dsr)               ***    730c
  TAPEBKUP(cs80tbdvr)                612c   844c
recoverable_data_overflow
  CS80(cs80)                  ***    191d
  CS80(cs80dsr)               ***    728c
  TAPEBKUP(cs80tbdvr)                610c   842c
recoverblock
  INITLOAD(sysglobals)        ***    241d
red
  DGL_INQ(dgl_inq)                  6051c  6056c
  DGL_POLY(dgl_poly)               20079c 20086c
  DGL_RAS(dgl_raster)              17026d 17027d 17028d 17029d 17030d 17031d 17032d 17033d 17034d 17035d 17036d 17037d 17038d 17039d
                                   17040d 17041d 17111c 17181c 17187c 17191c 17223c 17235c 17240c 17273c 17629c
  DGL_VARS(dgl_vars)          ***   1024d
red_intensity
  GLE_RGL(gle_ras_out)               8043d  8197c
redef_background
  DGL_INQ(dgl_inq)            ***   6414c
  GLE_HPGL(gle_hpgl_out)      ***   7701c
  GLE_TYPES(gle_types)        ***   1097d
reduced_screen
  DGL_C_OUT(dgl_confg_out)          11038d 11057c 11061c 11179c 11273c
refbufblocks
  LIBRARIAN                         2162d  2516c  2517c  2519c  2524c
refbuffer
  INITLOAD(loader)                  1502d  1542c  1544c  1546c  1553c
  LIBRARIAN                         2139d  2178c  2209c  2211c  2212c  2269c  2271c  2274c  2275c  2288c  2290c  2292c  2524c
refbuftop
  LIBRARIAN                         2140d  2283c  2525c  2637c  2639c
referenceptr
  INITLOAD(loader)                   861d   876d   885d
  LIBRARIAN                          149d   199c  1501c  1506c  1513c  1517c  1528c  1533c  1535c  1581c  1613c  1624c  1838c  1839c
                                    2382c  2385c  2405c  2410c  2794c  2795c
refgvr
  LIBRARIAN                          313d   407c   408c   409c   411c   521c   524c   535c  1301c
refinblock
  LIBRARIAN                         2163d  2300c  2301c  2632c
refinbytes
  LIBRARIAN                         2281d  2283c  2284c  2297c  2299c
```

```
refindex
  INITLOAD(loader)          ***   1107d
refinsize
  LIBFARIAN                       2164d  2286c  2291c  2302c  2630c  2639c
reflink
  LIBFARIAN                        313d   407c   521c  1302c
refloc
  LIBFARIAN                        313d   409c   411c   521c   533c  1303c
refoutblock
  LIBFARIAN                       2165d  2175c  2212c  2271c  2272c
refoutsize
  LIBFARIAN                       2166d  2178c  2214c  2215c  2330c
refptr
  LIBFARIAN                        314d  1298c  1299c  1301c  1302c
refptrptr
  INITLOAD(loader)                 876d   906d
refsize
  INITLOAD(loader)                 992d  1540c  1542c  1544c
  LIBFARIAN                       1290c  1298c  1299c  1302c  1476d  1609c  1610c  1613c  1620c  1622c  2214c  2630c  2768c  2770c
                                  2771c  2772c
refstart
  INITLOAD(loader)                 991d  1544c
  LIBFARIAN                       1289c  1299c  2214c  2632c  2771c
reftop
  INITLOAD(loader)          ***   1107d
reg
  CRT(crt)                         263d   265c
  DC_DRV(extdc)                    205d   207d   209d
  DC_DRV(intdc)                    250d   252d   348d   351c   356d   369c   370c   374c   379c   382c   384c
  DI_DRV(extdi)                    161d   163d
  G_DRV(extg)                      178d   180d
  H_DRV(exth)                      176d   178d
  IOLIB(serial_0)                 2362d  2374c  2379c  2384c  2389c  2394c  2399c  2404c  2411c  2422c  2427c  2432c  2437c  2442c
                                  2447c  2452c  2459c
  KERNEL(iodeclarations)           428d   431d
  LIBFARIAN                        451d   571d   572c   576c   580d   582c   586d   588c   591d   592c   596d   598c   608c   689c
                                   692c   977c   982c
  RS_DRV(rs)                       178d   180d
regi
  LIBFARIAN                        433d   665c   668c   674c   677c   683c   684c   724c   813c   853c   857c   860c   992c   993c
                                  1006c  1015c  1069c  1090c  1098c  1108c  1113c  1115c  1126c  1128c  1130c  1134c  1136c
regclass
  LIBFARIAN                        450d   571d   572c   574c   608c   689c   692c   977c   982c
register
  KERNEL(general_0)                748d   751d   754d   757d   760d   763d   853d   864c   872d   883c   891d   903c   910d   922c
                                   930d   937c   945d   953c
regmasksym
  LIBFARIAN                        745d   762c   773c
regmasksymtype
  LIBFARIAN                        743d   745d
regrange
  LIBFARIAN                        416d   433d   436d   451d   571d   580d   586d   591d   596d
regstring
  LIBFARIAN                        737d   740d   762c   766c   769c   773c   774c   781c   785c   806c   807c   900c   901c   914c
                                   916c
regsym
  LIBFARIAN                        556d   574c
regsymtype
  LIBFARIAN                        547d   556d
regtype
  LIBFARIAN                        415d   450d   547d   571d
release_heap
  SEGMENTER(asm)                   222d   245c   307c
```

```
releaseuser
  INIT(ldr)                       2429c  2506c
  INITLOAD(loader)                1080d  1130d  1597c  1648c
  M68KSYS(ci)                      805c   928c  1124c
  TAIL                      ***    77c
relocatable
  INITLOAD(loader)          ***   858d
  LIBFARIAN                        180c  1565c  1616c  1925c  2182c  2377c  2451c
relocatablebase
  INITLOAD(loader)                 951d  1410c
  LIBFARIAN                        368c  1683c  1969c
relocatablesize
  INITLOAD(loader)                 950d  1378c  1411c
  LIBFARIAN                        369c  1684c  1968c  2616c
relocate
  INITLOAD(loader)                1107d  1546c
relocbase
  INITLOAD(loader)                1024d  1410c
  LIBFARIAN                       1683c  2616c
relocdelta
  INITLOAD(loader)                1026d  1410c
  LIBFARIAN                       1492c  1493c  1683c
reloctype
  INITLOAD(loader)                 858d   921d
remaining_surf_bytes
  AMIGO(amigodvr)                  717d   739c   740c   741c
remote
  IOLIB(hpib_2)                   1793d  1951d
remoted
  IOLIB(hpib_3)                   2043d  2154d  2157c
remove
  KEYS(keys)                       406d   478c   479c   485c   507c
remove_extraneous_volumes
  CTABLE                    ***   1822c
  CTABLE(ctr)                      308d   887d
ren_line
  IOLIB(hpib_2)                   1819c  1874c  1959c  1966c
  KERNEL(iodeclarations)    ***    388d
rendfile
  ASCII(asciimodule)               25d   157c   170c   200c   221c   283c
repeatcount
  GLE_RGL(gle_ras_out)      ***   8061d
repeating_timeout
  PRINTER(prtdvr)                  53d   251c
repeatrate
  GLE_RGL(gle_ras_out)      ***   8060d
replregcopy
  GLE_RGL(gle_ras_out)      ***   8123c
  SYSDEVS(sysdevs)          ***    133d
report
  M68KSYS(ci)                      387d   553c
  MINIT(cs80ir)                    785d   797c
reporting_sec
  CS80(cs80)                       278d   376c
req
  SRM_DRV                         1998d  2014c  2084d  2091c  2100c  2104c
  SRM_DRV(srm)                    1040d  1098d  1111d  1707d  1720c
req_are_you_alive
  SRM_DRV(srm)                     154d  1384c  1414c  1466c  1535c  1541c
req_catalog
  SRM_DRV(srm)                     136d  1562c  1581c
req_catprotect
  SRM_DRV(srm)                     140d  1600c  1619c
```

```
req_changeprotect
  SRM_DRV(srm)                    139d  1639c  1654c
req_close
  SRM_DRV(srm)                    131d  1688c  1698c
req_copy
  SRM_DRV(srm)                    150d  1418c  1713c  1724c
req_create
  SRM_DRV                  ***   1776c
  SRM_DRV(srm)                    137d  1419c  1751c
req_createlink
  SRM_DRV                         1800c  1821c
  SRM_DRV(srm)             ***    138d
req_flock
  SRM_DRV                         1890c  1898c
  SRM_DRV(srm)                    128d  1416c
req_funlock
  SRM_DRV                         2052c  2060c
  SRM_DRV(srm)             ***    129d
req_gang_cleanup
  SRM_DRV                  ***   1871c
  SRM_DRV(srm)             ***    153d
req_info
  SRM_DRV                         1852c  1860c
  SRM_DRV(srm)             ***    130d
req_init_vol
  SRM_DRV(srm)             ***    145d
req_label
  SRM_DRV(srm)                    146d  1667c  1677c
req_log_addr
  AMIGO(csamigo)                   49d   412c
req_old_read
  SRM_DRV(srm)             ***    121d
req_open
  SRM_DRV                         1917c  1937c
  SRM_DRV(srm)             ***    134d
req_position
  SRM_DRV                         1950c  1961c
  SRM_DRV(srm)             ***    123d
req_purgelink
  SRM_DRV                         1978c  1991c
  SRM_DRV(srm)             ***    135d
req_read
  SRMAM(srmammodule)               78c    85c    89c
  SRM_DRV                  ***   2006c
  SRM_DRV(srm)                    124d  1433c  1443c
req_set_eof
  SRM_DRV                         2031c  2041c
  SRM_DRV(srm)             ***    125d
req_status
  AMIGO(csamigo)                   47d   310c
req_syndrome
  AMIGO(csamigo)                   48d   334c
req_volstatus
  SRM_DRV                         2070c  2077c
  SRM_DRV(srm)             ***    144d
req_write
  SRMAM(srmammodule)              125c   132c   136c
  SRM_DRV                  ***   2092c
  SRM_DRV(srm)             ***    122d
req_xchg_data
  SRM_DRV(srm)             ***    141d
req_xchg_open
  SRM_DRV                         1833c  1841c
  SRM_DRV(srm)             ***    149d
```

```
reqsize
  LIFDAM(lifmodule)               1017d  1022c  1023c  1048c  1052c
request
  AMIGO(amigodvr)                  430d   564c   573c   621c
  ASCII(asciimodule)                 6d    13d    27c   267c   270c   278c   286c
  BUBBLES(bubble)                   29d    53d    69c    81c   100c
  CRT(crt)                         402d   410c   415c   448c   496c
  CS80(cs80dvr)                    987d  1094d  1106c  1168d  1182c  1185c  1214d  1250c  1260c  1339c  1362c  1395c
  EPROMS(eproms)                     8d    12d    47c    49c
  F9885(f9885dvr)                   40d   166c   223c   267c   298c
  GCRT(crtb)                       272d   280c   285c   316c   364c
  INIT(initunits)                 2139d  2141d  2148d  2152d  2156d  2160d  2165d  2228d  2232d
  INIT(misc)                       213d   426d   432c   454c   483d   488c   493c   503c   505c   515d   521c   523c   534d   575c
                                   612c   629c   645c   656c   674c   682c
  INITLOAD(bootdammodule)          543d   544d   607d   691c   747d   755c   773c
  INITLOAD(mini)                   309d   384d   424c   437c   495c   526c
  INITLOAD(sysglobals)              68d   140d
  KEYS(keys)                        72d    81c    94s   101c   122c   129c   136c
  LIFDAM(lifmodule)                 28d   195d  1126c  1159c
  M68KSYS(ci)                      306d   335c   345c   383c   731d   735c
  PRINTER(prtdvr)                   40d    47d   294c
  SRMAM(srmammodule)                40d   178d   188c   191c
  SRMDAM(srmdammodule)              45d  1626d  1648c  1687c  1690c  1736c
  SYSDEVS(sysdevs)                 372d   376d   407d   432d   434c   484d   486c
  UCSD_AM(ucsd_am)                  40d   163c   172c   223c   252c   259c
  UCSD_DAM(ucsdmodule)              32d    66d   634c   650c
requestltype
  SYSDEVS(sysdevs)                  40d   153d   180d
request_service
  IOLIB(hpib_3)                   2036d  2116d
request_status
  F9885(f9885dvr)                  115d   125c
request_vol
  MIUI                              99d   546c
  TAPEBKUP                        1048d  1294c  1307c  1433c  1467c
requested
  IOLIB(hpib_3)                   2028d  2062d  2068c
  SRM_DRV                         2014c  2100c
  SRM_DRV(srm)                     478d   715d   805d  1720c
requesting_unit
  CS80(cs80)               ***    209d
requeue
  SRM_DRV(srm)             ***    296d
res_adjust
  GLE_STEXT(gle_stext)            4022d  4037c  4041c
reserved
  SYSDEVS(sysdevs)         ***    318d
reserved1
  INITLOAD(sysglobals)     ***    284d
reserved2
  INITLOAD(sysglobals)     ***    284d
reset_card_and_confirm_timeout
  PRINTER(prtdvr)                   85d   126c   131c   247c
resetcard
  SRMAM(srmammodule)                92c   139c   154c
  SRM_DRV(srm)                    1000d  1228d  1367c
resetdevice
  HPHIL(hphil)             ***    261c
  MOUSE(mouse)             ***    175c
  SYSDEVS(sysdevs)         ***    302d
resetmask
  A804XDVR(a804xdvr)       ***    385c
  SYSDEVS(sysdevs)         ***    166d
  TAIL                              87c    92c
```

```
resetTheLoop
  HPHIL(hphil)                          277d  370c
resolution
  A804XDVR(a804xdvr)          ***       316c
  SYSDEVS(sysdevs)                      219d  480c
resolve
  INITLOAD(loader)                     1397d 1519c
resolved
  INIT(ldr)                   ***      2391c
  INITLOAD(loader)                     1016d 1170d 1183c 1186c 1191c 1306c 1308c 1311c 1341c 1346c 1436c
  LIBRARIAN                   ***      1733c
resolveexts
  INITLOAD(loader)                     1082d 1206d 1350c
response
  IOLIB(hpib_3)                        2038d 2118d 2123c
  MINIT(qminit)                        1036d 1042c 1043c 1046c 1047c
  MIUI                                  124d  137c  138c  139c  144c  149c  152c  290d  303c  304c  309c  310c
  TAPEBKUP                             1054d 1067c 1068c 1069c 1074c 1079c 1082c
restore_heap
  SEGMENTER(asm)                         62d  145c  161c  226c
restore_line
  PRINTER(prtdvr)                       152d  198c  240c  263c
result
  C_HOOK                                 82d   99c  105c  107c
  DGL_RAS(dgl_raster)                 17383d 17407c 17413c 17415c
  GCRT(crtb)                            155d  175c  181c  183c
  KEYS(keys)                            194c  196c  199c  203c  204c  219c  221c  433c  436c  565c
  LIFDAM(lifmodule)                      86d   87c
  NONUSKBD1(non_us_kbd1)                 42c   44c   48c   49c   52c   56c   65c
  SRMDAM(srmdammodule)                   79d   82c
  SYSDEVS(sysdevs)            ***       265d
retaddmode
  TAPEBKUP(cs80tbr)                     114d  122c
retain
  CTABLE                                124d 1253c 1255c 1257c 1259c 1261c
retransmit
  CS80(cs80)                  ***       163d
  CS80(cs80dsr)               ***       712c
  TAPEBKUP(cs80tbdvr)                   594c  826c
retroactive_color_support
  DGL_HPGL(dgl_hpgl)          ***     17265c
  DGL_INQ(dgl_inq)            ***      6408c
  DGL_RAS(dgl_raster)         ***     17597c
  DGL_VARS(dgl_vars)          ***      1105d
retroactive_polygon_support
  DGL_HPGL(dgl_hpgl)          ***     17266c
  DGL_INQ(dgl_inq)            ***      6396c
  DGL_RAS(dgl_raster)         ***     17596c
  DGL_VARS(dgl_vars)          ***      1104d
retry_required
  CS80(cs80dsr)                         604d  635d  705c  713c  776d  881c  883c  884c  887c  889c  890c  896c  898c  899c
                                        902c  904c  905c  908c  910c  911c  943c  945c  946c  949c  951c  952c
  CS80(cs80dvr)                        1029d 1046c 1048c 1049c 1052c 1054c 1055c 1098d 1125c 1127c 1135c 1217d 1225c 1334c
                                       1337c
  MINIT(qminit)                         935d  961c  963c  964c  967c  969c  970c  977c  979c  980c 1101d 1128c 1130c 1131c
                                       1142c 1144c 1149c 1151c 1152c 1157c 1159c 1160c 1166c 1168c 1169c 1172c 1174c 1175c
                                       1178c 1180c 1181c 1185c 1188c 1189c 1196c 1198c 1199c 1208c 1210c 1211c 1214c 1218c
                                       1219c 1227c 1229c 1230c 1233c 1235c 1236c 1239c 1241c 1242c 1245c 1249c 1250c 1266c
                                       1268c 1269c
  TAPEBKUP(cs80tbdvr)                   368d  371c  373c  374c  376c  378c  379c  397d  400c  402c  403c  415d  418c  420c
                                        421c  433d  436c  438c  439c  441c  443c  444c  515d  587c  595c  641c  664d  669c
                                        671c  672c  678c  680c  681c  695d  714d  723c  725c  726c  729c  731c  732c  819c
                                        827c  896d  899c  902c  903c  915d  918c  920c  921c  933d  936c  938c  939c
```

```
return
  PRINTER(prtdvr)                        62d  301c
return_are_you_alive
  SRM_DRV(srm)                          362d  825d
return_bytes
  INIT(misc)                            259d  272c  287c  295c
return_cat_passwords
  SRM_DRV(srm)                          405d  829d
return_catalogue
  SRM_DRV(srm)                          382d  827d
return_change_protect_codes
  SRM_DRV(srm)                          428d  831d
return_change_volume_label
  SRM_DRV(srm)                          445d  833d
return_closeafile
  SRM_DRV(srm)                          463d  835d
return_closest_color
  DGL_RAS(dgl_raster)                 17067d 17164c 17187c 17191c 17240c
return_copy
  SRM_DRV(srm)                          481d  837d
return_createafile
  SRM_DRV(srm)                          508d  839d
return_createlink
  SRM_DRV(srm)                          526d  841d
return_fileinfo
  SRM_DRV(srm)                          575d  845d
return_jang_cleanup
  SRM_DRV(srm)                          592d  847d
return_header_type
  SRM_DRV(srm)                          320d  363d  383d  406d  429d  446d  464d  482d  509d  527d  544d  560d  576d  593d
                                        620d  635d  658d  685d  701d  720d  742d  759d  774d  788d  813d  872d
return_initialize
  SRM_DRV(srm)                          619d  849d
return_lockafile
  SRM_DRV(srm)                          634d  851d
return_machine_type
  DGL_C_IN(dgl_config_in)     ***     12055c
  DGL_C_OUT(dgl_config_out)   ***     11234c
  DGL_TOOLS(dgl_tools)                20007d 20013d 20048c 20051c 20053c 20057c 20063c 20067c
return_mess_header
  SRM_DRV(srm)                          363d  383d  406d  429d  446d  464d  482d  509d  527d  544d  560d  576d  593d  620d
                                        635d  658d  685d  701d  720d  742d  759d  774d  788d  813d 1543c
return_openafile
  SRM_DRV(srm)                          657d  853d
return_positiontoarecord
  SRM_DRV(srm)                          684d  855d
return_purgelink
  SRM_DRV(srm)                          700d  857d
return_readarecord
  SRM_DRV(srm)                          719d  859d
return_request_type
  SRM_DRV(srm)                          323d 1443c 1464c
return_set_date
  SRM_DRV(srm)                          741d  861d
return_set_eof
  SRM_DRV(srm)                          758d  863d
return_unlockafile
  SRM_DRV(srm)                          773d  865d
return_volume_status
  SRM_DRV(srm)                          787d  867d
return_writearecord
  SRM_DRV(srm)                          812d  869d
return_xchg_data
  SRM_DRV(srm)                ***       543d
```

```
return_xchg_open
  SRM_DRV(srm)                          559d   843d
reversepointers
  INITLOAD(loader)                     1357d  1371c  1425c
revision
  INITLOAD(loader)            ***       942d
  LIBRARIAN                             349c  1961c
rexchange
  SRM_DRV(srm)                ***       895d
rfileInfo
  SRMDAM(srmdammodule)                  422c  1103c  1445c  1485c
  SRM_DRV(srm)                ***       897d
rfill_index_color
  GLE_RGL(gle_ras_out)        ***      8415c
rgangclean
  SRM_DRV(srm)                ***       899d
rgcbinit
  GLE_RGL(gle_ras_out)        ***      8394c
rget_pip2
  GLE_RGL(gle_ras_out)        ***      8412c
rget_polygon_info
  GLE_RGL(gle_ras_out)                 8301d  8413c
rget_raster
  GLE_RGL(gle_ras_out)        ***      8425c
rgltemp1
  GLE_RGL(gle_ras_out)        ***      8055d
rgltemp2
  GLE_RGL(gle_ras_out)        ***      8056d
rgltemp3
  GLE_RGL(gle_ras_out)        ***      8057d
rgltemp4
  GLE_RGL(gle_ras_out)        ***      8058d
rgltemp5
  GLE_RGL(gle_ras_out)        ***      8059d
rgraphics_on_off
  GLE_RGL(gle_ras_out)        ***      8416c
rhead
  SRMAM(srmammodule)                     60c   107c
  SRMDAM(srmdammodule)                  338c   397c
  SRM_DRV(srm)                          923d  1423c
ri_line
  IOLIB(serial_0)                      2403c  2451c
  KERNEL(iodeclarations)      ***       412d
right
  CRT(crt)                               95d   141d
  GCRT(crtb)                  ***        74d
  SYSDEVS(sysdevs)            ***        93d
rightchar
  CRT(crt)                    ***       463c
  GCRT(crtb)                  ***       334c
  INIT(misc)                  ***       171d
  KEYS(keys)                            162c   307c
rinit
  SRM_DRV(srm)                ***       901d
rleng
  INIT(fs)                              801d   805d   806d   807d   809d   812d   814d   819d   822d   827d   829d   835d   838d
rlf
  CRT(crt)                    ***       121d
  GCRT(crtb)                  ***        54d
  SYSDEVS(sysdevs)            ***        77d
rlinestyle
  GLE_RGL(gle_ras_out)        ***      8409c
```

```
rlist
  DGL_HPGL(dgl_hpgl)                  17020d 17065d
  DGL_INQ(dgl_inq)                     6014d  6090d  6185c  6186c  6191c  6192c  6200c  6201c  6206c  6207c  6216c  6217c  6222c  6223c
                                       6228c  6229c  6236c  6237c  6241c  6242c  6249c  6250c  6254c  6255c  6260c  6261c  6267c  6268c
                                       6275c  6276c  6280c  6281c  6288c  6289c  6290c  6291c  6297c  6298c  6299c  6300c
  DGL_RAS(dgl_raster)                 17351d 17430d 17467c 17469c 17470c 17537c
  DGL_VARS(dgl_vars)                   1125d  1130d
  LIB(dgl_lib)                        20060d 20067d 20754d 20764c 20772d 20782c
rlock
  SRMDAM(srmdammodule)        ***      1475c
  SRM_DRV(srm)                ***       903d
rmove
  GLE_RGL(gle_ras_out)        ***      8396c
rnextByte
  ASCII(asciimodule)                    153d   159c   168c   169c   171c   187c
rom_id
  CTABLE(brstuff)                       967d   979c   996c
roman8_kbd
  KEYS(keys)                  ***       316c
  SYSDEVS(sysdevs)            ***       150d
romanalphabet
  KEYS(keys)                            247d   324c   325c
romanspecials
  KEYS(keys)                            243d   329c   330c
romexset
  CRT(crt)                              359d   362c
romexsettype
  CRT(crt)                              358d   359d
romheader
  EPROMS(eproms)                         18d    86c
romtokanamap
  CRT(crt)                               72d   193c
romtokanatype
  CRT(crt)                               48d    72d
root_password
  SRM_DRV(srm)                          271d  1524c
ropen
  SRMDAM(srmdammodule)                  755c   917c  1101c
  SRM_DRV(srm)                ***       905d
rotate
  MINIT(xminit)                        1461d  1467c  1468c
row
  CRT(crt)                              182d   188c   192c
  C_HOOK                                 76d    86c   105c
  DGL_RAS(dgl_raster)                 17377d 17391c 17413c
  GCRT(crtb)                            149d   159c   181c
  MOUSE(mouse)                ***        53d
row_def
  C_HOOK                                 73d    76d
  DGL_RAS(dgl_raster)                 17374d 17377d
  GCRT(crtb)                            146d   149d
rows
  CRT(crt)                              231d   242c   248c   251c
rp
  INITLOAD(loader)            ***       885d
  LIBRARIAN                            1574c  1691c  1692c
  SRM_DRV(srm)                         1516d  1524c
rp1
  GEN(dgl_gen)                         3117d  3125c  3131c
rp2
  GEN(dgl_gen)                         3117d  3126c  3130c
rpg_disable
  SYSDEVS(sysdevs)            ***       177d
```

```
rpg_enable
  SYSDEVS(sysdevs)              ***    177d
rpghandler
  KEYS(keys)                           355d   581c
rpgishook
  A804XDVR(a804xdvr)            ***    108c
  KEYS(keys)                    ***    581c
  MOUSE(mouse)                         163c   170c
  SYSDEVS(sysdevs)                     161d   604c
rpgreshook
  A804XDVR(a804xdvr)            ***    390c
  SYSDEVS(sysdevs)                     180d   424c   603c
rpolygon
  GLE_RGL(gle_ras_out)          ***   8419c
rpos
  SRM_DRV(srm)                  ***    907d
rpp
  INITLOAD(loader)              ***    906d
  LIBRARIAN                            149d   185c  1502c  1518c  1521c  1531c  1570c  1613c  1625c  1834c  1835c  1836c  1841c  2383c
                                      2386c  2394c  2793c
rptr
  LIBRARIAN                           2388d  2382c  2383c  2385c  2386c  2393c  2394c  2398c  2403c  2411c
rpurge
  SRM_DRV(srm)                  ***    909d
rread
  SRM_DRV(srm)                         911c  1423c
rs
  RS_DRV(init_rs)               ***    215d
  RS_DRV(rs)                    ***    156d
rs232
  CTABLE(options)               ***     79d
rs_drivers
  RS_DRV(init_rs)                      203d   228c   229c   250c
rs_init
  RS_DRV(init_rs)               ***    230c
  RS_DRV(rs)                    ***    172d
rs_initialize
  RS_DRV                        ***    147d
rs_ist
  RS_DRV(init_rs)               ***    231c
  RS_DRV(rs)                    ***    173d
rs_rdi
  RS_DRV(init_rs)               ***    232c
  RS_DRV(rs)                    ***    174d
rs_rdo
  RS_DRV(init_rs)               ***    236c
  RS_DRV(rs)                    ***    178d
rs_rdv
  RS_DRV(init_rs)               ***    234c
  RS_DRV(rs)                    ***    176d
rs_tfr
  RS_DRV(init_rs)               ***    238c
  RS_DRV(rs)                    ***    182d
rs_wti
  RS_DRV(init_rs)               ***    233c
  RS_DRV(rs)                    ***    175d
rs_wto
  RS_DRV(init_rs)               ***    237c
  RS_DRV(rs)                    ***    180d
rs_wtu
  RS_DRV(init_rs)               ***    235c
  RS_DRV(rs)                    ***    177d
rset_color
  GLE_RGL(gle_ras_out)                 8408c  8517c
```

```
rsetdate
  SRM_DRV(srm)                  ***    913d
rsetecf
  SRM_DRV(srm)                  ***    915d
rsize
  DGL_HPGL(dgl_hpgl)                  17018d 17063d
  DGL_INQ(dgl_inq)                     6011d  6087d  6171c
  DGL_RAS(dgl_raster)                 17349d 17428d 17534c
  DGL_VARS(dgl_vars)                   1123d  1128d
  LIB(dgl_lib)                        20058d 20065d 20752d 20763c 20764c 20770d 20781c 20782c
rsolve
  LIBRARIAN                            1651d  2984c
rtctime
  A804XDVR(a804xdvr)            ***    239d
  CLOCK(clock)                          51d    85d    91d   101d   115d   127d
  SYSDEVS(sysdevs)                     195d   206d   455d   458d
rtpass
  SRM_DRV                              1907d  1926c  1970d  1987c
  SRM_DRV(srm)                         1010d  1018d  1026d  1046d  1082d  1094d  1552d  1577c  1590d  1615c  1628d  1648c  1733d  1760c
rts_line
  IOLIB(serial_0)                      2244c  2264c  2307c  2327c  2373c  2421c
  KERNEL(iodeclarations)        ***    407d
rule
  GLE_RGL(gle_ras_out)                 8120d  8123c  8125c
rule_reg
  GLE_RGL(gle_ras_out)                 8125c  8387c
  GLE_TYPES(gle_types)          ***   1268d
runlight
  KEYS(keys)                    ***    108c
  SYSDEVS(sysdevs)                     392d   584d   585c
runlist
  LIBRARIAN                            1482d  1579c  1606c
runlock
  SRM_DRV(srm)                  ***    917d
runstate
  BUBBLES(bubble)                       71c    82c    83c    88c   101c   102c   115c
runtime
  MINIT(cs80ir)                 ***    528d
  MINIT(qminit)                 ***   1191c
runworkfile
  M68KSYS(ci)                          1016d  1114c  1134c
rvb
  CS80(cs80)                    ***    110d
  CS80(cs80dvr)                 ***   1066c
rvol
  SRMDAM(srmdammodule)          ***    397c
  SRM_DRV(srm)                  ***    919d
rwrite
  SRMAM(srmammodule)            ***    107c
  SRM_DRV(srm)                  ***    921d
rx
  DGL_HPGLI(dgl_hpgli)                18018d 18030c 18038d 18102c 18103c
  DGL_KNOB(dgl_knob)                  18025d 18030c 18038d 18145c 18146c 18148c
  GEN(igl_gen)                         3044d  3252d  3259c  3261c  3263c
  LIB(igl_lib)                        20078d 20081d 20935d 20943c 20953d 20996c
rxvec
  DGL_POLY(dgl_poly)                  20911d 20973c
ry
  DGL_HPGLI(dgl_hpgli)                18018d 18030c 18038d 18102c 18103c
  DGL_KNOB(dgl_knob)                  18025d 18031c 18038d 18145c 18146c 18149c
  GEN(igl_gen)                         3044d  3252d  3258c  3261c  3262c
  LIB(igl_lib)                        20078d 20081d 20935d 20943c 20953d 20996c
ryvec
  DGL_POLY(dgl_poly)                  20911c 20973c
```

```
s
  AMIGO(csamigo)                ***     98d
  CONVERT(convert_text)                 46d    72c    73c    74c
  CRT(crt)                              405d   423c   424c   428c
  DGL_HPGL(dgl_hpgl)                  17027d 17049c 17051c 17052c 17071d 17077c 17083c 17085c 17091c 17092c 17098c 17100c 17106c 17108c
                                      17114c 17116c 17122c 17177c 17178c
  DGL_POLY(dgl_poly)                  20059d 20108d
  DGL_RAS(dgl_raster)           ***   17172d
  GCRT(crtb)                            275d   293c   294c   298c
  GLE_HPGL(gle_hpgl_out)                7067d  7075c
  GLE_HPGLI(gle_hpgl_in)              18040d 18048c
  GLE_SMARK(gle_smark)                  5095d  5135c  5139c
  GLE_UTLS(gle_utls)                  21008d 21009d 21010d 21039d 21066c 21069d 21082c 21085c 21094c 21096c 21106d 21115c
  INIT(fs)                              749d   753d   755d   807d   808d   817d   818d   820d   822d   825d   826d   828d   830d   832d
                                        834d   836d   837d
  INIT(ldr)                             2412d  2414c  2416c
  INIT(misc)                            209d   211d   246d   250c   252c   253c   300d   304c   305c   306c   307c   308c   309c   310c
                                        311c   312c   313c   314c   315c   316c   317c   318c   319c   320c   321c   322c   323c   324c
                                        325c   326c   327c   328c   329c   330c   331c   332c   333c   334c   335c   336c   337c   338c
                                        339c   340c   341c   342c   343c   344c   345c   346c   347c   348c   349c   350c   351c   352c
                                        353c   354c   355c   358c   359c   367d   378c   379c   380c   381c   382c   383c   384c   385c
                                        386c   387c   389c   391c   392c   393c   394c   395c   396c   397c   398c   399c   400c   401c
                                        404c   405c   406c   407c   408c   409c   412c   416c
  INITLOAD                              1727d  1731c
  KERNEL(general_0)                     959d   961c
  KERNEL(lodeclarations)        ***     377d
  LIB(dgl_lib)                        20072d 20709d 20725c 20726c 21252d 21263c 21264c 21265c 21266c 21301d 21316c 21317c 21318c 21319c
  LIBRARIAN                             98d    100c   284d   287c   288c   290c   2902d  2905c  2908c  2909c  2912c  2913c  3232d  3235c
                                      3236c  3239c  3240c  3241c  3246d  3251c  3252c  3254c  3259c  3264c  3270d  3273c  3274c  3276c
                                      3278c  3283c
  LIFDAM(lifmodule)                     90d    93c    94c    98d    103c   104c   105c   109d   110c   131d   139c   142c   144c   148c
                                        151c   152c   160d   167c   168c   172c   176c   178c   184c   186c
  M68KSYS(ci)                           308d   365c   368c   369c   374c   375c   378c   379c   408d   443c   451c   454c   460c   502c
                                        503c   662d   664c   665c   666c   672d   674c   675c   676c   678c   679c   682c   683c   747d
                                        750c   751c   752c   754c   755c   757c
  MOREFSYS(mfs)                         21d    40d    533d   541c   544c   554d   556d   563d   568c   573c   578c   596c   597c   598c
                                        599c   600c   605c   608c   609c   611c   615c   618c   620c   622c   625c   626c   632c   633c
                                        636c   637c   646c   647c   648c   649c   651c   652c   655c   657c   658c   660c   661c   669d
                                        672c   673c   675c
  SRMDAM(srmdammodule)                  1521d  1536c  1580c  1586c  1599c  1601c  1608c  1610c
s1
  AMIGO(amigodvr)                       804c   817c   821c   834c   872c
  AMIGO(csamigo)                ***     101d
  GLE_UTLS(gle_utls)                  21011d 21118d 21130c
  MINIT(hminit)                 ***     317c
s1_11
  AMIGO(csamigo)                ***     74d
s1_20
  AMIGO(csamigo)                ***     79d
s1_21
  AMIGO(csamigo)                ***     79d
s1_24
  AMIGO(csamigo)                ***     81d
s1_25
  AMIGO(csamigo)                ***     81d
s1_26
  AMIGO(csamigo)                ***     82d
s1_27
  AMIGO(csamigo)                ***     82d
s1_28
  AMIGO(csamigo)                ***     83d
s1_29
  AMIGO(csamigo)                ***     83d

s1_30
  AMIGO(csamigo)                ***     84d
s1_4
  AMIGO(csamigo)                ***     71d
s1_5
  AMIGO(csamigo)                ***     71d
s1_6
  AMIGO(csamigo)                ***     72d
s1_type
  AMIGO(csamigo)                        67d    101d   120d
s2
  GLE_UTLS(gle_utls)                  21012d 21119d 21130c
  INITLOAD                              1758d  1761c  1762c  1764c
  LIFDAM(lifmodule)                     109d   110c
s3
  M68KSYS(ci)                           232d   234c   241c   247c   249c
s3va
  MINIT(cs80ir)                         827d   835c   836c   837c   838c   839c   840c   841c
s_address
  TAPEBKUP(cs80tbr)                     228d   242c   243c
s_setadd
  TAPEBKUP(cs80tbr)                     227d   241c
s_volunit
  TAPEBKUP(cs80tbr)                     226d   239c   240c
sample
  GLE_GENI(gle_geni)            ***     3070c
  GLE_HPGLI(gle_hpgl_in)        ***     18370c
  GLE_KNOB(gle_knob_in)         ***     18221c
  GLE_TYPES(gle_types)          ***     1209d
sample_locator
  LIB(dgl_lib)                        20080d 20334d
sappend
  LIBRARIAN                             158c   159c   181c   191c   195c   196c   209c   465c   468c   501c   504c   540c   568c   572c
                                        574c   581c   583c   593c   597c   603c   609c   610c   616c   622c   699c   700c   720c   721c
                                        762c   766c   769c   773c   774c   825c   826c   831c   832c   872c   878c   893c   899c   901c
                                        913c   916c   949c   961c   963c   1034c  1035c  1042c  1054c  1055c  1056c  1061c  1064c  1080c
                                      1083c  1087c  1096c  1097c  1107c  1120c  2936c
sareyoualive
  SRM_DRV(srm)                          876d   1531c
sarm
  TAPEBKUP(cs80tbr)                     110d   120c   121c   122c   123c   124c   125c   126c   127c
sat
  GEN(dgl_gen)                          3048d  3049d  3166d  3176c  3177c  3178c  3190d  3217c  3223c  3224c
save
  DGL_C_OUT(dgl_confg_out)            11411d 11505d
  UCSD_AM(ucsd_am)                      72d    83c    98c
save_crthook
  DGL_C_OUT(dgl_confg_out)            11023d 11068c 11185c
save_joresult
  DGL_C_OUT(dgl_confg_out)            11374d 11381c 11383c 11410d 11452c 11454c 11463c 11465c
save_state
  SEGMENTER(asm)                        141d   250c   312c   328c
savearea
  CRT(crt)                              633c   643c   645c
  GCRT(crtb)                            126d   471c   480c
  SYSDEVS(sysdevs)              ***     56d
savecod
  SEGMENTER(asm)                        65d    148c   163c
saved_char_height
  GLE_SMARK(gle_smark)                  5103d  5126c  5146c
saved_char_width
  GLE_SMARK(gle_smark)                  5102d  5125c  5145c
saved_color
  DGL_POLY(dgl_poly)                  20203d 20210c 20245c 20426d 20598c 20840c
```

```
saved_drawing_mode
  LIB(dgl_lib)                    20965d 20979c 21011c
saved_echo
  PRINTER(prtdvr)                   81d   157c   181c   182c
saved_escapecode
  INITLOAD(mini)                   413d   457c   462c
  MINIT(mminit)                    150d   169c   174c
  PRINTER(prtdvr)                   87d    91c   102c   103c
  TAPEBKUP                        1164d  1167c  1170c
saved_ioe_result
  PRINTER(prtdvr)                   89d    93c   101c
saved_ioe_sc
  PRINTER(prtdvr)                   88d    92c   100c
saved_ioresult
  DISCHPIB(bkgnd)                  127d   137c   140c
  INIT(misc)                       260d   286c   293c
  MINIT(qminit)                    881d   932d   997c   998c
  MIUI                             130d   143c   192c   209c   210c   211c   214c
  TAPEBKUP                        1060d  1073c  1091c  1098c  1101c  1165d  1168c  1171c
  TAPEBKUP(cs80tbdvr)              315d   452d   477c   478c
saved_length
  GLE_SMARK(gle_smark)            5108d  5115c  5168c
  LIB(dgl_lib)                   20963d 20977c 21003c
saved_line
  PRINTER(prtdvr)                   78d   176c   177c   178c   186c
saved_line_pattern
  GLE_SMARK(gle_smark)            5106d  5117c  5170c
saved_linestyle
  DGL_POLY(dgl_poly)             20049d 20204d 20213c 20246c 20944c 21018c 21019c
  GLE_SMARK(gle_smark)            5107d  5116c  5167c
  LIB(dgl_lib)                   20962d 20976c 21002c
saved_linewidth
  DGL_POLY(dgl_poly)             20050d 20205d 20214c 20247c 20945c 21020c 21021c
  LIB(dgl_lib)                   20961d 20975c 21008c
saved_mode
  GLE_SMARK(gle_smark)            5109d  5114c  5169c
  LIB(dgl_lib)                   20964d 20978c 21004c
saved_pattern
  LIB(dgl_lib)                   20960d 20974c 21005c
saved_reportbit
  INIT(misc)                       261d   285c   294c
saved_text_cos_dir
  GLE_SMARK(gle_smark)            5104d  5127c  5148c
saved_text_sin_dir
  GLE_SMARK(gle_smark)            5105d  5128c  5149c
saved_timeout
  GLE_HPGL(gle_hpgl_out)          7513d  7566c  7610c
  GLE_HPGLI(gle_hpgl_in)         18328d 18339c 18398c
saved_ureportchange
  CS80(cs80dsr)                    775d   874c   958c
  CS80(cs80dvr)                   1147d  1153c  1161c
savedef
  SEGMENTER(asm)                    67d   149c   164c
saveec
  INIT(misc)                       228d   234c   241c
saveentry
  SEGMENTER(asm)                    66d   150c   165c
saveesc
  ETU                               81d   821c   823c   827c   863d  1137c  1142c  1265c  1268c  1272c
saveffpw
  SRMDAM(srmdammodule)            1632d  1680c  1807c
savefileid
  SRMDAM(srmdammodule)            1338d  1411c  1426c  1429c  1630d  1677c  1804c

savefiles
  SEGMENTER(asm)                    68d   151c   166c
savefsavepathid
  SRMDAM(srmdammodule)            1635d  1678c  1805c
saveftid
  SRMDAM(srmdammodule)            1631d  1679c  1806c
saveftitle
  SRMDAM(srmdammodule)            1634d  1683c  1685c  1809c  1810c
savefvid
  SRMDAM(srmdammodule)            1633d  1681c  1808c
saveglob
  SEGMENTER(asm)                    64d   147c   162c
saveheap
  SEGMENTER(asm)                    63d   145c   146c   161c   225c
saveid
  SRMDAM(srmdammodule)             657d   819c   824c   848c   851c
saveintlevel
  INIT(isr)                        139d   142c   145c
saveio
  ETU                               80d   141c   820c   822c   862d  1136c  1139c  1264c  1267c
  INIT(misc)                       227d   233c   240c
savelecf
  SRMDAM(srmdammodule)            1337d  1412c  1413c  1426c
savelist
  SEGMENTER(asm)                    61d   144c   160c
savepathid
  SRMDAM(srmdammodule)             349d   351c  1629d  1676c  1803c
savepc
  LIBRARIAN                       1150d  1154c  1156c
savesize
  CRT(crt)                         561c   632c
  GCRT(crtb)                       427c   470c
  SYSDEVS(sysdevs)          ***     57d
savewd
  SRM_DRV                         1865d  1874c
  SRM_DRV(srm)              ***   1073d
savpathid
  SRMDAM(srmdammodule)             878d   889c   891c
sb
  INITLOAD(loader)                 844d   891d
  LIBRARIAN                       2663c  2664c  2665c
  MINIT(cs80ir)                    856d   863c   864c   865c   866c
sb_correction_bytes
  AMIGO(amigodvr)           ***     848c
  AMIGO(csamigo)            ***     124d
sb_offset
  AMIGO(amigodvr)                  840c   841c   843c
  AMIGO(csamigo)            ***     123d
sb_pad1
  AMIGO(csamigo)            ***     119d
sb_pad2
  AMIGO(csamigo)            ***     121d
sb_s1
  AMIGO(amigodvr)           ***     838c
  AMIGO(csamigo)            ***     120d
sb_tva
  AMIGO(amigodvr)           ***     839c
  AMIGO(csamigo)            ***     122d
sbyte
  INITLOAD(loader)                 857d   932d
  LIBRARIAN                 ***   2660c
sbytere
  INITLOAD(loader)                 844d   891d
  LIBRARIAN                 ***   2662c
```

```
sc
  BUBBLES(bubble)                    63c   64c   67c   143d  149c  150c
  CS80(cs80dsr)              ***     620c
  CTABLE                           1191d 1243d 1245c 1274c 1278c 1467c 1505c 1508c 1531c 1540c 1554c 1574c 1596c 1600c
                                   1623c 1781s 1806c
  CTABLE(brstuff)                   934d  958d 1007c
  CTABLE(ctr)                       325d  328d  330d  331d  332d  333d  334d  335d  336d  337d  338d  413c  473c  497c
                                    668d  670c  673c  677d  679c  682c  686d  690c  694c  698d  702c  707c  711d  713c
                                    716c  720d  724c  733c  737d  741c  752c  756d  764c  768c  772d  783c  787c  791d
                                    795c  796c  797c  798c  805d  808c  809c  881c
  CTABLE(options)                   114d  118d  120d  123d  125d  127d
  CTABLE(scanstuff)          ***   1114c
  DISCHPIB(dischpib)                221c  223c  232c  239c  246c  261c  292c  315c  351c  363c  394c  426c  446c  508c
                                    526c
  ETU                               637d  641c  698d  700c  701c  703c  704c
  F9885(f9885dvr)            ***    260c
  INIT(initunits)            ***   2206c
  INITLOAD                   ***   1679c
  INITLOAD(bootdammodule)           546d  778d  786c  787c  788c
  INITLOAD(sysglobals)       ***    146d
  MIUI                               35c  199c  200c  350c  351c
  PRINTER(prtdvr)            ***    273c
  SRMAM(srmammodule)                149c  167c
  SRMDAM(srmdammodule)              163c  417c
  SRM_DRV(srm)                     1230c 1253d 1256c 1297d 1304c 1315c 1328c 1330c 1336c 1338c 1378c 1393c 1415c 1417c
                                   1420c 1422c 1435c 1440c 1445c 1450c 1460c
  TAPEBKUP                         1009c 1017c 1092c
sc_list
  CTABLE                     ***   1467c
  CTABLE(options)            ***    217d
sc_list_length
  CTABLE                     ***   1464c
  CTABLE(options)                   211d  215d
sc_list_type
  CTABLE(options)                   214d  218d
sc_table_entry_ptr
  PRINTER(prtdvr)                    67d   95c  111c  120c  208c  274c  288c
scalef
  DGL_VARS(dgl_vars)         ***   1293d
  GEN(dgl_gen)                     3290c 3440c
scan_dav
  CTABLE(scanstuff)                1040d 1083d 1114c 1115c 1116c 1117c
scanneddevice_letter
  CTABLE                           1347c 1364c 1471c
  CTABLE(scanstuff)                1040d 1083d 1106c 1111c 1126c 1137c
scanner_msus
  CTABLE                           1206d 1463c 1472c
scanner_uep
  CS80(cs80dsr)                     614d  618c  619c  620c  621c  622c  626c  627c  628c
scanptr
  EPROMS(eproms)                     39d   60c   61c   66c   68c   71c   80c   86c   88c   89c   95c   98c  100c  102c
                                    106c  107c
scanrec
  EPROMS(eproms)                     31d   39d
scanstuff
  CTABLE                     ***   1186d
  CTABLE(scanstuff)          ***   1025d
scantitle
  ETU                        ***    316c
  INIT(fs)                   ***    757c
  LIBRARIAN                  ***   2931c
  M68KSYS(ci)                       540c  918c
  MIUI                       ***    139c
  TAPEBKUP                   ***   1069c

scantitle_ok
  MIUI                              129d  139c  140c
  TAPEBKUP                         1059d 1069c 1070c
scat
  SRM_DRV(srm)                      878d 1556c
scatpass
  SRM_DRV(srm)                      880d 1594c
scg_base
  DISCHPIB(dischpib)                213d  282c  341c
schangeprotect
  SRM_DRV(srm)                      882d 1632c
schangevolume
  SRM_DRV(srm)                      884d 1663c
sclear
  A804XDVR(a804xdvr)                333d  350c  352c
sclose
  SRM_DRV(srm)                      886d 1684c
scode
  EDRIVER(edriver)                   36d   42d   45d   47d   74d  131c  132c  147c  306d  322c  325d  331c  334d  340c
                                    343d  348c
  ETU                                69d  572c  578c  591c  593c  648c  660c  704c  720c  760c  763c  777c  778c  779c
                                    780c  781c  783c 1071c 1231c
scodes
  ETU                               696d  703c
scopy
  SRM_DRV(srm)                      888d 1709c
screatefile
  SRM_DRV(srm)                      890d 1743c
screatelink
  SRM_DRV                    ***   1793c
  SRM_DRV(srm)               ***    892d
screen
  CRT(crt)                          164d  192c  272c  284c  321c  331c  332c  335c  341c  342c  344c  483c  585c  588c
                                    605c  607c  608c  615c  619c  624c  628c  642c  644c  668c  675c  678c
screenheight
  CRT(crt)                          160d  303c  670c
  GCRT(crtb)                        109d  212c  381c  386c  392c  395c  401c  403c  406c  504c  506c  507c
screensize
  CRT(crt)                          163d  333c  343c  471c  473c  673c  675c
  GCRT(crtb)                        112d  342c  344c  507c
screenwidth
  CRT(crt)                          159d  272c  300c  321c  331c  333c  335c  342c  343c  471c  472c  483c  576c  580c
                                    582c  594c  603c  614c  619c  624c  628c  642c  644c  669c
  GCRT(crtb)                        108d  209c  342c  343c  503c  505c  507c
scrolldown
  CRT(crt)                          338d  465c
  GCRT(crtb)                        245d  336c
scrollup
  CRT(crt)                          328d  449c  468c  488c
  GCRT(crtb)                        240d  319c  339c  356c
scrptr
  CRT(crt)                           58d  158d  164d
  GCRT(crtb)                 ***     94d
scrtype
  GCRT(crtb)                         93d   94d
scsize
  INITLOAD(bootdammodule)           779d  786c  787c  788c
sctype
  INITLOAD(sysglobals)       ***    199d
sd
  ETU                               219d  240c  242c
sdate
  LIFDAM(lifmodule)                  56d  329c  331c  332c  333c  346c  347c  348c  349c  516c  573c
```

```
sdc
  DISCHPIB(dischpib)                    210d   321c
  PRINTER(prtdvr)                        56d   116c
sdc_message
  IOLIB(hpib_2)                  ***   1844c
  KERNEL(iodeclarations)         ***    301d
sdp
  INITLOAD(loader)               ***    907d
  LIBRARIAN                            1737c  2023c
search
  MINIT(xminit)                        1439d  1789c
search_for_other_units
  CTABLE                               1200d  1365c  1410c  1458c  1472c  1479c
search_higher_numbered_cs80_units
  CTABLE                               1354d  1411c
sec
  AMIGO(csamigo)                        244d   251d   252d   253d   254d   255d   256d   257d   258d   259d   260d   261d   262d   263d
                                        269c
  DISCHPIB(dischpib)                    196d   198d   202d   270d   282c   287d   293c   329d   341c   357d   364c   440d   452c   457c
sec_ext_size
  SRM_DRV(srm)                   ***    666d
sec_per_trk
  AMIGO(csamigo)                        131d   193d   194d   196d   197d   198d   199d   200d   201d   202d   203d   225c   347c   348c
                                        374c
  MINIT(hminit)                         293c   364c   366c   373c   391c   403c   437c   439c   446c
secbuf
  LIFDAM(lifmodule)                     486d   529c   530c
second
  M68KSYS(ci)                           112d   114c   125c   127c   130c
secondary
  DGL_C_OUT(dgl_config_out)           11039d 11271c 11295c 11333c 11336c 11344c
  IOLIB(hpib_2)                         1794d  1972d
secondary_dam
  CTABLE                                1607c  1629c  1631c
  CTABLE(ctr)                            293d   513c
secondary_extent
  SRM_DRV(srm)                           500d  1768c
seconds
  MIUI                                    55d    84c    85c    87c
seconds_since_midnight
  SRMDAM(srmdammodule)           ***    372c
  SRM_DRV(srm)                   ***    256d
sect
  AMIGO(csamigo)                         92d   347c   374c
  MINIT(cs80ir)                         524d   564d   818d   840c
sect_per_surf
  AMIGO(amigodvr)                       712d   739c
section
  MINIT(cs80ir)                         560d   625d   636d   642c
sector
  F9885(f9885dvr)                        60d   116d   150c
  INITLOAD(bootdammodule)        ***    591d
  INITLOAD(mini)                        406d   426c   428c   430c   440c   446c
  LIBRARIAN                            2740d  2742c
  MINIT(bminit)                          67d    79c    83c    86c
  MIUI                                  456d   469c   481c
sector_count
  MINIT(iramigo)                        201d   204c
sector_ert
  MINIT(cs80ir)                  ***    551d
  MINIT(qminit)                  ***   1216c
sector_ert_passes
  MINIT(qminit)                         1113d  1216c
```

```
sector_type
  MIUI                                  419d   456d
sectorcount
  MINIT(xminit)                        1405d  1570c  1574c  1586c  1687c  1690c
sectors
  F9885(f9885dvr)                       112d   161c   165c   211c
  MINIT(asmr)                          1332d  1336d  1343d  1347d
sectors_per_block
  SRM_DRV(srm)                   ***    609d
sectorsize
  ASCII(asciimodule)                     11d    49c    70c    72c    74c
  LIBRARIAN                            2735d  2742c  2820c
sectorspertrack
  MINIT(xminit)                        1403d  1524c  1535c  1570c  1629c  1688c  1723c  1734c  1758c  1799c
seek_cmd
  AMIGO(csamigo)                         50d   388c
seek_cmd_buf
  AMIGO(csamigo)                        381d   387c   388c
seekcomplete
  F9885(f9885dvr)                        62d   208c
seekeof
  INITLOAD(sysglobals)           ***     67d
seekposit
  ASCII(asciimodule)                     55d   201c   221c   284c
segment_list
  SEGMENTER(asm)                         73d   144c   157c   158c   160c   172c   223c
segment_proc
  SEGMENTER(asm)                        104d   109d   279d
  SEGMENTER(segmenter)                   21d    35d    36d
segment_space
  SEGMENTER(asm)                 ***     96d
  SEGMENTER(segmenter)           ***     38d
segmenter
  SEGMENTER(segmenter)           ***     16d
segs
  ETU                                   310d   313c   316c   322c
  M68KSYS(ci)                           399d   540c   914d   918c
select_code
  DGL_C_OUT(dgl_config_out)           11026d 11081d 11118c 11269c 11314c
  GLE_HPIB(gle_hpib_io)               10015d 10064d 10066c 10072d 10075c 10081d 10087c 10092c 10095c 10108d 10111c 10119d 10122c 10124c
                                      10131c 10136c 10143d 10146c 10148c 10472c 10480c 10494c 10495c 10496c 10498c
  IOLIB(general_1)                      237d   238d   240d   242d   244d   246d   274d   276c   282d   285c   293d   296c   304d   308c
                                        317d   322c   330d   336c   341c   344c
  IOLIB(hpib_0)                        1709d  1711d  1713d  1722d  1725c  1733d  1736c  1744d  1749c
  IOLIB(hpib_1)                         393d   395d   398d   401d   413d   432d   435c   443d   446c   448c   455c   460c   467d   470c
                                        472c   482d   485c   487c   498d   500c
  IOLIB(hpib_2)                        1779d  1781d  1785d  1795d  1797d  1800d  1801d  1811d  1816c  1818c  1819c  1820c  1821c  1824c
                                       1826c  1827c  1828c  1829c  1853d  1856c  1889d  1891c  1973d  1976c  1981d  1984c  1995d  1997c
                                       2001d  2003c
  IOLIB(hpib_3)                        2029d  2031d  2037d  2039d  2041d  2043d  2046d  2063d  2066c  2068c  2072c  2080d  2084c  2117d
                                       2120c  2122c  2123c  2124c  2128c  2135d  2138c  2145d  2148c  2154d  2157c  2164d  2167c
  IOLIB(serial_0)                      2215d  2217d  2219d  2231d  2236c  2238c  2240c  2250c  2253c  2255c  2270c  2274c  2276c  2294d
                                       2299c  2301c  2303c  2313c  2316c  2318c  2333c  2337c  2339c  2357d  2365c  2367c  2369c  2408c
                                       2411c  2456c  2459c
  IOLIB(serial_3)                      2503d  2506d  2509d  2512d  2515d  2518d  2529d  2556c  2558c  2560c  2564c  2565c  2566c  2571c
                                       2572c  2576c  2577c  2584c  2598d  2603c  2605c  2607c  2625c  2632c  2636c  2637c  2646c  2647c
                                       2656c  2663c  2665c  2669c  2682d  2687c  2689c  2698c  2703c  2706c  2711c  2712c  2715c  2717c
                                       2722c  2736d  2741c  2743c  2745c  2756c  2760c  2765c  2766c  2777c  2781c  2783c  2788c  2801d
                                       2806c  2808c  2810c  2813c  2820c  2825c  2836d  2841c  2843c  2845c  2848c  2855c  2860c
  KERNEL(general_0)                     747d   750d   753d   756d   759d   762d   852d   857c   861c   871d   876c   880c   890d   896c
                                        900c   909d   915c   919c   929d   934c   944d   950c
  KERNEL(iodeclarations)                656d   664d   666c
  PRINTER(prtdvr)                        66d    98c   102c   139c   273c   274c
```

```
selectcode
  MINIT(xminit)                          1364d  1515c  1516c  1520c  1586c  1628c  1722c  1733c  1757c  1798c
selfinit
  C_HOOK                          ***     44d
  DGL_TOOLS(dgl_tools)            ***   20022d
semantics
  KEYS(keys)                             209c   260c   434c   487c   555c
  NONUSKBD1(non_us_kbd1)                  61c   145c
  SYSDEVS(sysdevs)                ***    279d
send_addr_sec
  AMIGO(csamigo)                         410d   414c
send_are_you_alive
  SRM_DRV(srm)                           358d   824d
send_break
  IOLIB(serial_3)                       2514d  2800d
send_cat_passwords
  SRM_DRV(srm)                           394d   828d
send_catalogue
  SRM_DRV(srm)                           371d   826d
send_change_protect_codes
  SRM_DRV(srm)                           420d   830d
send_change_volume_label
  SRM_DRV(srm)                           437d   832d
send_closeafile
  SRM_DRV(srm)                           454d   834d
send_command
  GLE_HPIB(gle_hpib_io)                10108d 10213c 10214c 10215c 10301c 10302c 10303c
  IOLIB(hpib_1)                          393d   432d   535c   536c   537c   623c   624c   625c
  IOLIB(hpib_2)                         1826c  1828c  1844c  1847c  1856c  1857c  1880c  1891c  1903c  1904c  1910c  1927c  1928c  1941c
                                        1942c  1945c  1976c  1984c  1991c  1997c  2003c
  IOLIB(hpib_3)                         2100c  2102c  2103c
send_copy
  SRM_DRV(srm)                           472d   836d
send_createafile
  SRM_DRV(srm)                           491d   838d
send_createlink
  SRM_DRV(srm)                           517d   840d
send_fileinfo
  SRM_DRV(srm)                           569d   844d
send_gang_cleanup
  SRM_DRV(srm)                           587d   846d
send_header_type
  SRM_DRV(srm)                           328d   359d   372d   395d   421d   438d   455d   473d   492d   518d   536d   554d   570d   588d
                                         603d   629d   645d   676d   694d   710d   732d   751d   768d   783d   800d   871d  1477d
send_initialize
  SRM_DRV(srm)                           602d   848d
send_lockafile
  SRM_DRV(srm)                           628d   850d
send_mess_header
  SRM_DRV                               1798c  1831c  1850c  1869c  1888c  1915c  1948c  1976c  2004c  2029c  2050c  2068c  2090c
  SRM_DRV(srm)                           359d   372d   395d   421d   438d   455d   473d   492d   518d   536d   554d   570d   588d   603d
                                         629d   645d   676d   694d   710d   732d   751d   768d   783d   800d  1533c  1560c  1598c  1637c
                                        1665c  1686c  1711c  1749c
send_openafile
  SRM_DRV(srm)                           644d   852d
send_positiontoarecord
  SRM_DRV(srm)                           675d   854d
send_purgelink
  SRM_DRV(srm)                           693d   856d
send_readarecord
  SRM_DRV(srm)                           709d   858d
send_request_type
  SRM_DRV(srm)                           331d  1485c
```

```
send_set_date
  SRM_DRV(srm)                           731d   860d
send_set_eof
  SRM_DRV(srm)                           750d   862d
send_sts_sec
  AMIGO(csamigo)                         308d   312c
send_syn_sec
  AMIGO(csamigo)                         332d   335c
send_unlockafile
  SRM_DRV(srm)                           767d   864d
send_volume_status
  SRM_DRV(srm)                           782d   866d
send_wait
  A804XDVR(a804xdvr)                     232d   244c   246c   295c   301c   306c
send_writearecord
  SRM_DRV(srm)                           799d   868d
send_xchg_data
  SRM_DRV(srm)                    ***    535d
send_xchg_open
  SRM_DRV(srm)                           553d   842d
sendbyteslsf
  A804XDVR(a804xdvr)                     151d   250c   251c   269c   276c   283c
sendcmd
  A804XDVR(a804xdvr)                      40d   138d   147c   159c   166c   174c   175c   203c   208c   236c   244c   250c   251c   266c
                                         267c   273c   274c   281c   282c   295c   301c   306c
  HPHIL(hphil)                            83c    88c   148c   172c
senddata
  A804XDVR(a804xdvr)                      41d   142d   154c   160c   174c   203c
  HPHIL(hphil)                            84c    85c    86c   146c   172c
sendhphilcmd
  HPHIL(hphil)                           291d   371c
sendreadpack
  SRMAM(srmammodule)                      67c    75c    84c
  SRM_DRV                         ***   1995d
  SRM_DRV(srm)                    ***   1095d
sendreq
  SRM_DRV(srm)                          1003d  1400d  1414c  1416c  1418c  1419c  1433c  1464c  1466c
sendwritepack
  SRMAM(srmammodule)                     114c   122c   131c
  SRM_DRV                         ***   2081d
  SRM_DRV(srm)                    ***   1108d
sent
  SRMAM(srmammodule)                      55d    64c    68c    73c    76c    84c   102d   111c   115c   120c   123c   131c
sequential_access
  SRMAM(srmammodule)                      70c   117c
  SRM_DRV(srm)                    ***    169d
serial_0
  IOLIB(serial_0)                 ***   2196d
serial_3
  IOLIB(serial_3)                 ***   2486d
serial_card
  DC_DRV(init_dc)                 ***    636c
  IOLIB(serial_0)                       2238c  2301c  2367c
  IOLIB(serial_3)                       2558c  2605c  2686c  2743c  2808c  2843c
  KERNEL(general_0)                     1044c  1131c  1141c
  KERNEL(iodeclarations)          ***    328d
  PRINTER(prtdvr)                 ***    223c
serial_dma
  DC_DRV(intdc)                   ***    421c
  IOLIB(general_4)                ***   1511c
  KERNEL(iodeclarations)          ***    550d
```

```
serial_fastest
  IOLIB(general_4)              ***   1508c
  KERNEL(iodeclarations)        ***    552d
  SRM_DRV(srm)                  ***   1344c
serial_fhs
  DISCHPIB(dischpib)                   471c   510c
  IOLIB(general_4)              ***   1508c
  KERNEL(iodeclarations)        ***    551d
serial_line
  IOLIB(serial_0)                     2219d  2357d  2476c
serialtextam
  INIT(misc)                           515d   697c
serialtextamhook
  INIT(misc)                           470c   697c
  INITLOAD(sysglobals)          ***    279d
  M68KSYS(ci)                   ***    334c
set_3v_address
  MINIT(cs80ir)                        564d   818d   843c
  MINIT(qminit)                       1179c  1209c  1228c  1240c
set_6845s
  GLE_RGL(gle_ras_out)          ***   8258c
set_acceleration
  DGL_HPGL(dgl_hpgl)                 17111d 17169c
set_access_date
  SRM_DRV(srm)                  ***    735d
set_address_and_return_mode
  TAPEBKUP(cs80tbdvr)           ***    670c
  TAPEBKUP(cs80tbr)                     26d   100d   129c
set_all_color_table
  DGL_RAS(dgl_raster)                17247d 17537c
set_aspect
  LIB(dgl_lib)                       20018d 20348d
set_auto_delay
  KEYS(keys)                    ***    589c
  SYSDEVS(sysdevs)              ***    140d
set_auto_pen
  DGL_HPGL(dgl_hpgl)                 17074d 17154c
set_auto_repeat
  KEYS(keys)                    ***    588c
  SYSDEVS(sysdevs)              ***    140d
set_baud_rate
  IOLIB(serial_3)                     2502d  2528d
set_char_length
  IOLIB(serial_3)                     2508d  2681d
set_char_size
  LIB(dgl_lib)                       20071d 20719c 20816d 21152c
set_color
  DGL_POLY(dgl_poly)                 20215c 20245c 20599c 20840c
  LIB(dgl_lib)                       20036d 20447d 21171c
set_color_model
  LIB(dgl_lib)                       20034d 20399d
set_color_table
  LIB(dgl_lib)                       20086d 20408d
set_creation_date
  SRM_DRV(srm)                  ***    737d
set_display_lim
  LIB(dgl_lib)                       20052d 20488d
set_echo_pos
  LIB(dgl_lib)                       20074d 20846d
set_file_mask
  AMIGO(amigodvr)               ***    680c
  AMIGO(csamigo)                        146d   274d
set_force
  DGL_HPGL(dgl_hpgl)                 17103d 17166c
```

```
set_gator_vals
  DGL_C_OUT(dgl_config_out)          11247d 11290c 11298c 11308c 11315c
set_hpib
  IOLIB(hpib_0)                       1709d  1722d
  IOLIB(hpib_2)                       1818c  1819c  1959c  1966c
set_in_use
  DISCHPIB(bkgnd)                       40d    89c   114c   119c   132c
set_kbdlang
  A804XDVR(a804xdvr)            ***    397c
  SYSDEVS(sysdevs)              ***    142d
set_kbdtype
  A804XDVR(a804xdvr)            ***    396c
  SYSDEVS(sysdevs)              ***    142d
set_line_style
  DGL_POLY(dgl_poly)                 20216c 20246c 20948c 21019c
  LIB(dgl_lib)                       20035d 20469d 21172c
set_line_width
  DGL_POLY(dgl_poly)                 20217c 20247c 20949c 21021c
  LIB(dgl_lib)                       20037d 20431d 21173c
set_locator_lim
  LIB(dgl_lib)                       20083d 20880d
set_marker
  GLE_GEN(gle_gen)              ***   2154c
  GLE_HPGL(gle_hpgl_out)        ***   7539c
  GLE_RGL(gle_ras_out)          ***   8407c
  GLE_TYPES(gle_types)          ***   1056d
set_options
  CS80(cs80)                            346d   522d   534c
  CS80(cs80dsr)                 ***    944c
  TAPEBKUP(cs80tbdvr)           ***    377c
set_parity
  IOLIB(serial_3)                     2511d  2735d
set_pgn_color
  DGL_POLY(dgl_poly)                 20014d 20102d
set_pgn_ls
  DGL_POLY(dgl_poly)                 20013d 20126d
set_pgn_style
  DGL_POLY(dgl_poly)                 20020d 20142d 20187c
set_pgn_table
  DGL_POLY(dgl_poly)                 20015d 20165d
set_polygon_color
  DGL_POLY(dgl_poly)                 20053d 20950c
set_release
  CS80(cs80)                            345d   500d   518c
  CS80(cs80dsr)                 ***    882c
set_rpg_rate
  SYSDEVS(sysdevs)                      178d   424c
set_rule
  GLE_RGL(gle_ras_out)                8120d  8228c  8229c  8239c  8282c
set_scanneddevice_letter
  CTABLE(scanstuff)                   1099d  1123c  1124c
set_serial
  IOLIB(serial_0)                     2215d  2231d
set_specified_unitvol
  TAPEBKUP(cs80tbdvr)           ***    730c
  TAPEBKUP(cs80tbr)                     24d    55d    69c
set_status_mask
  CS80(cs80)                            347d   538d   552c
  CS80(cs80dsr)                         888c   950c
set_stop_bits
  IOLIB(serial_3)                     2505d  2597d
set_text_rot
  LIB(dgl_lib)                       20070d 20720c 20785d 21157c
```

set_timeout															
GLE_HPIB(gle_hpib_io)		10081d	10480c												
IOLIB(general_1)		246d	330d												
set_timing															
LIB(dgl_lib)		20030d	20386d	21169c											
set_to_listen															
IOLIB(hpib_1)		410d	743d	775c											
IOLIB(hpib_2)		1841c	1866c	1920c	1938c	1960c	1991c								
set_to_talk															
IOLIB(hpib_1)		408d	696d	725c											
IOLIB(hpib_2)	***	1908c													
IOLIB(hpib_3)	***	2099c													
set_unit															
CS80(cs80)		340d	433d	441c											
CS80(cs80dsr)	***	763c													
TAPEBKUP(cs80tbdvr)		653c	724c	886c											
set_unitvol															
CS80(cs80)		341d	445d	453c											
CS80(cs80dsr)	***	903c													
CS80(cs80dvr)	***	1047c													
MINIT(qminit)	***	962c													
TAPEBKUP(cs80tbdvr)		372c	437c												
set_velocity															
DGL_HPGL(dgl_hpgl)		17095d	17132c	17141c	17163c										
set_viewport															
LIB(dgl_lib)		20019d	20286d	20374c											
set_width															
GLE_RGL(gle_ras_out)		8136d	8230c												
set_window															
LIB(dgl_lib)		20020d	20321d												
setadd															
CS80(cs80)		571d	582c												
MINIT(cs80ir)		831d	837c												
TAPEBKUP(cs80tbr)		116d	124c												
setcopyright															
LIBRARIAN		3206d	3653c												
setcopytimeout															
SRM_DRV(srm)		1115d	1247d												
setcursor															
CRT(crt)	***	412c													
GCRT(crtb)	***	282c													
INITLOAD(sysglobals)	***	67d													
setdate															
LIFDAM(lifmodule)		113d	515c	940c	1008c	1093c									
setdefaulttimeout															
SRM_DRV(srm)		1114d	1235d												
seteofpack															
SRMDAM(srmdammodule)		1279c	1319c	1347c	1508c										
SRM_DRV	***	2022d													
SRM_DRV(srm)	***	1101d													
setformatoption															
MINIT(cs80ir)		604d	614c												
setglobal															
LIBRARIAN		3200d	3633c												
setintegertimeout															
SRM_DRV(srm)		1253d	1378c	1415c	1417c	1420c	1422c								
setintlevel															
CLOCK(clock)		144c	158c												
HPHIL(hphil)		139c	152c	166c	175c	188c	201c	215c	218c						
INIT(isr)		143c	145c												
KEYS(keys)		86c	88c	113c	114c	132c									
setioresult															
SRMDAM(srmdammodule)		79c	124c	145c	190c	233c	272c	297c	324c	401c	753c	835c	886c	920c	926c
		998c	1107c	1110c	1197c	1273c	1480c	1545c	1581c	1592c	1594c	1604c	1798c	1819c	
setlen															
CS80(cs80)		574d	586c												
TAPEBKUP(cs80tbr)		193d	200c	222d	235c										
setmaxmodules															
LIBRARIAN		3177d	3634c												
setname															
LIBRARIAN		3231d	3643c												
UCSD_DAM(ucsdmodule)		459d	663c												
setoptn															
CS80(cs80)		526d	530c												
setpasswords															
INITLOAD(sysglobals)	***	138d													
SRMDAM(srmdammodule)		1653c	1711c												
setpctl															
F9885(f9885dvr)	***	195c													
setrel															
CS80(cs80)		505d	512c												
setreloc															
LIBRARIAN		3194d	3648c												
setretadd															
TAPEBKUP(cs80tbr)		113d	121c												
setrpgrate															
KEYS(keys)	***	590c													
SYSDEVS(sysdevs)		369d	421d												
setrtctime															
CLOCK(clock)		85d	111c	123c	160c										
setrunlight															
KEYS(keys)		116c	575c												
M68KSYS(ci)		116c	1107c												
SYSDEVS(sysdevs)		393d	587d												
setsdate															
LIFDAM(lifmodule)		339d	1150c												
setstatus															
A804XDVR(a804xdvr)	***	399c													
KEYS(keys)	***	451c													
SYSDEVS(sysdevs)		391d	580d	588c											
setstsmsk															
CS80(cs80)		543d	548c												
setsysdate															
M68KSYS(ci)		163c	267c												
SYSDEVS(sysdevs)		386d	469d												
setsystime															
M68KSYS(ci)		166c	280c												
SYSDEVS(sysdevs)		387d	473d												
sett															
A804XDVR(a804xdvr)	***	261c													
SYSDEVS(sysdevs)	***	214d													
settracksector															
F9885(f9885dvr)		51d	59d	116d	147c										
setunit															
CS80(cs80)		389d	393c	423d	427c	504d	511c	568d	579c						
MINIT(cs80ir)		578d	585c	633d	639c										
TAPEBKUP(cs80tbr)		43d	50c	65c											
setunit15															
TAPEBKUP(cs80tbr)		221d	234c												
setunitprefix															
INITLOAD(sysglobals)	***	137d													
LIFDAM(lifmodule)	***	1162c													
SRMDAM(srmdammodule)		1665c	1785c												
UCSD_DAM(ucsdmodule)	***	653c													
setunitvol															
MINIT(cs80ir)		602d	612c	660d	666c	687d	692c	712d	718c	744d	750c	780d	790c	829d	835c
		858d	863c												
TAPEBKUP(cs80tbr)		84d	90c	116d	120c	144d	148c	167d	171c	191d	198c	266d	274c		

```
setunitvol_type
  MINIT(cs80ir)                          576d   583d   602d   660d   687d   712d   744d   780d   829d   858d
  TAPEBKUP(cs80tbr)                       41d    48d    63d    84d   112d   144d   167d   191d   266d
setup
  EDRIVER(edriver)                       110d   198c   206c   269c   282c
setup_display
  LIB(dgl_lib)                         21111d 21233c 21281c
setup_fnn
  SRM_DRV                                1805c  1811c  1922c  1983c
  SRM_DRV(srm)                           1512d  1573c  1611c  1644c  1756c
setup_fns
  SRMDAM(srmdammodule)                    119d   142c   478c   521c   975c  1079c  1187c  1226c  1230c  1253c  1360c
setup_fns3
  SRMDAM(srmdammodule)                    133d  1119c  1286c  1359c
setup_for_polygon
  DGL_POLY(dgl_poly)                   20908d 21040c 21086c 21130c 21161c
setup_gator
  GLE_RGL(gle_ras_out)                   8244d  8479c
setup_internal
  DGL_C_OUT(dgl_config_out)            11210d 11287c 11300c 11305c
setup_smh
  SRM_DRV                                1798c  1831c  1850c  1869c  1888c  1915c  1948c  1976c  2004c  2029c  2050c  2068c  2090c
  SRM_DRV(srm)                           1477d  1533c  1560c  1598c  1637c  1665c  1686c  1711c  1749c
setup_vnh
  SRM_DRV                                1803c  1920c  1981c  2073c
  SRM_DRV(srm)                           1491d  1569c  1607c  1642c  1670c  1754c
setupfib
  UCSD_DAM(ucsdmodule)                    159d   251c   293c   316c
setupfibforfile
  ETU                                     305d   361c   893c
setuphpgl
  DGL_C_IN(dgl_config_in)              12084d 12135c
  DGL_C_OUT(dgl_config_out)            11399d 11511c
setupkrob
  DGL_C_IN(dgl_config_in)              12039d 12134c
setupraster
  DGL_C_OUT(dgl_config_out)            11153d 11510c
setupread
  A804XDVR(a804xdvr)                      126d   147c   243c   245c   293c   300c   305c
setupsys
  INIT                          ***       30d
setvol
  CS80(cs80)                              569d   580c
  MINIT(cs80ir)                           579d   586c
  TAPEBKUP(cs80tbr)                        44d    51c    66c
setvolumedate
  INITLOAD(sysglobals)          ***      133d
  LIFDAM(lifmodule)             ***     1150c
  M68KSYS(ci)                   ***      287c
  SRMDAM(srmdammodule)          ***     1793c
  UCSD_DAM(ucsdmodule)          ***      655c
setvolumename
  INIT(misc)                    ***      447c
  INITLOAD(sysglobals)          ***      133d
  LIFDAM(lifmodule)             ***     1138c
  UCSD_DAM(ucsdmodule)          ***      663c
setwaitforlocktimeout
  SRM_DRV(srm)                           1116d  1241d
setxy
  CRT(crt)                                298d   310c   460c   461c   462c   463c   464c   466c   469c   473c   474c   489c   491c
  GCRT(crtb)                              207d   219c   331c   332c   333c   334c   335c   337c   340c   344c   345c   357c   359c
sexcharge
  SRM_DRV                       ***      1829c
  SRM_DRV(srm)                  ***       894d
```

```
sf
  M68KSYS(ci)                             581d   598c   620c
sfile
  M68KSYS(ci)                             387d   425c   472c   537c   540c   542c   546c   553c  1033d  1038c  1043c  1045c  1046c  1049c
                                         1052c  1056c
sfileinfo
  SRM_DRV                       ***      1848c
  SRM_DRV(srm)                  ***       896d
sfm_cmd
  AMIGO(csamigo)                          289d   291c   292c
sfm_cmd_array
  AMIGO(csamigo)                          285d   291c
sfm_cmd_array_type
  AMIGO(csamigo)                          281d   285d
sfm_cmd_type
  AMIGO(csamigo)                          276d   281d   286d   287d   289d
sfm_oc
  AMIGO(csamigo)                          284d   286d   287d
sfm_sec
  AMIGO(csamigo)                          283d   292c
sgangclean
  SRM_DRV                       ***      1867c
  SRM_DRV(srm)                  ***       898d
sgn8
  TAPEBKUP(cs80tbdvr)                     710d   717c
sh_exc
  ETU                                      34d   157c   197c   203c   773c  1009c  1247c
share
  SRM_DRV                                1908d  1930c
  SRM_DRV(srm)                  ***      1083d
share_code
  SRMDAM(srmdammodule)          ***       619c
  SRM_DRV                       ***      1930c
  SRM_DRV(srm)                            277d   650d
shareBits
  SRM_DRV(srm)                  ***       665d
sharecode
  SRMDAM(srmdammodule)                    906d   922c
shared_share_code
  SRMDAM(srmdammodule)                    290c   293c   621c   808c   840c
  SRM_DRV(srm)                  ***       163d
sharemode
  SRMDAM(srmdammodule)                    195d   239c   288c   290c   293c   968d   976c   980c  1028d  1060c  1081c  1096c  1159c
shead
  SRM_DRV(srm)                  ***       922d
shift
  KEYS(keys)                              215c   216c   429c   444c   445c   446c   449c   450c   466c   491c   496c   497c   500c   508c
                                          510c   512c   528c   540c   565c
  LIBRARIAN                               997d  1140c
  NONUSKBD1(non_us_kbd1)                   68c    69c
  SYSDEVS(sysdevs)              ***       266d
shifted_left
  CS80(cs80dvr)                          1192c  1240c
shifted_right
  CS80(cs80dvr)                          1244c  1271c  1298c  1310c
shiftor
  LIBRARIAN                              1001d  1006c  1011c
shiftortype
  LIBRARIAN                               999d  1002d
```

```
short
  CTABLE(ctr)                  ***    403c
  INITLOAD(loader)                    926d  1272c  1462c  1526c
  LIBRARIAN                           411c  1280c  1628c  1646c  1927c  1931c  1941c  2053c  2102c  2315c  2389c  2399c  2404c  2406c
                                     2412c  2414c  2531c  2581c  2582c  2621c  2651c  2715c  2762c  2780c  2810c
  SEGMENTER(asm)               ***    135c
short_defaults
  DGL_VARS(dgl_vars)           ***   1320d
  GEN(dgl_gen)                       3284c  3456c  3458c
short_flag
  DGL_POLY(dgl_poly)           *** 20975c
  DGL_VARS(dgl_vars)           ***   1317d
  GEN(dgl_gen)                       3411c  3462c
  LIB(dgl_lib)                      20624c 20655c
short_tc
  DISCHPIB(dischpib)                  209d   395c
short_vertex
  DGL_POLY(dgl_poly)                20387d 20539c 20542c
```

```
shortint
  A804XDVR(a804xdvr)                   65d    66d   126d   151d   152d   233d
  AMIGO(amigodvr)                     428d   439d   443d   449d   492d   507d   558d   795d
  AMIGO(csamigo)                       90d   123d   129d   130d   131d   244d
  CLOCK(clock)                 ***     47d
  CRT(crt)                             53d   163d   165d   167d   205d   206d   207d   298d   315d   316d   329d   339d   547d   550d
                                      551d   552d   553d   554d   713d   714d
  CS80(cs80dvr)                       982d   985d   995d  1018d  1027d  1218d
  CTABLE                             1212d  1216d  1268d  1292d  1374d
  CTABLE(brstuff)              ***    968d
  CTABLE(ctr)                         311d   313d   314d   325d   328d   329d   330d   331d   332d   333d   334d   335d   336d   337d
                                      338d   339d   424d   425d   426d   427d   578d   580d   604d   615d   655d   668d   677d   686d
                                      698d   711d   720d   737d   756d   772d   791d   805d   812d
  CTABLE(opticns)                     136d   215d
  CTABLE(scanstuff)                  1043d  1052d  1053d  1056d  1094d  1135d
  C_HOOK                               49d    55d
  DGL_C_OUT(dgl_confg_out)          11026d 11029d 11030d 11031d 11081d 11086d 11087d 11162d 11163d 11213d 11214d 11215d 11216d
  DGL_RAS(dgl_raster)          ***  17384d
  DISCHPIB(bkgnd)                      47d    79d    84d    96d
  DISCHPIB(dischpib)                  196d   198d   199d   200d   287d   357d   383d   388d   410d
  F9885(f9885dvr)                      55d    94d   108d   112d
  GCRT(crtb)                          100d   108d   109d   110d   111d   112d   113d   114d   117d   121d   122d   126d   128d   129d
                                      130d   131d   132d   207d   224d   225d   226d   251d   372d   417d   419d   498d   529d   530d
                                      531d
  HPHIL(hphil)                 ***    228d
  INIT(fs)                            748d   801d   803d   805d   806d   807d   809d   810d   811d   812d   814d   819d   820d   822d
                                      827d   829d   831d   833d   835d   838d
  INIT(ldr)                          2386d  2408d
  INIT(misc)                          181d   228d   247d   262d   265d   428d
  INITLOAD                     ***   1737d
  INITLOAD(bootcammodule)             546d   560d   578d   589d   609d   649d   778d
  INITLOAD(loader)                    827d   828d   833d   835d   892d  1022d  1029d  1038d  1043d  1082d  1174d  1206d  1210d  1215d
                                     1220d  1225d  1400d  1401d  1423d  1589d
  INITLOAD(mini)                      319d   334d   373d   413d
  INITLOAD(sysglobals)                 30d    77d    96d   104d   124d   154d   170d   239d   294s
  KEYS(keys)                           53d   253d   377d
  LIBRARIAN                            46d    64d   109d   151d   441d   448d   478d   755d  1273d  1476d  1486d  1655d  1656d  1718d
                                     1719d  1720d  1746c  1852d  1999d  2000d  2001d  2032c  2309d  2669c  2722d  2734d  2829d  2832d
                                     2834d  2903d  3015d  3016d  3027d  3055d  3247d  3270d  3289d  3306d
  M68KSYS(ci)                         112d   313d   872d
  MINIT(asmr)                        1331d  1335d  1337d  1342d  1346d  1348d
  MINIT(hminit)                       254d   299d   304d   345d   346d   347d   350d   351d   352d   412d   413d   414d
  MINIT(mminit)                       116d   143d   150d
  MINIT(qminit)                       885d  1098d
  MINIT(xminit)                      1365d  1366d  1384d
  MIUI                                223d   286d   395d   398d   399d   401d   402d   410d   458d
  MOREFSYS(mfs)                        19d    22d    35d    38d    57d   401d   536d   553d   560d   564d   667d
  PRINTER(prtdvr)                      77d    87d
  SRMDAM(srmdammodule)                 76d   100d   912d   961d   962d  1020d  1021d
  SRM_DRV(srm)                        215d   216d   997d  1405d
  SYSDEVS(sysdevs)                     49d    59d    60d    88d    91d   128d   129d   133d   134d   135d   320d   321d   322d   323d
                                      598d
  TAPEBKUP                           1110d  1164d  1225d
  UCSD_AM(ucsd_am)             ***     45d
  UCSD_DAM(ucsdmodule)         ***     55d
shortpatch
  LIBRARIAN                          2368d  2447c  2457c
shortpointer
  HEAPT(hpm)                           93d    96c   140c   141c   270c   311c
showprompt
  ETU                          ***    111d
shuffle
  LIFDAM(lifmodule)                   683d   907c
```

```
si
  INITLOAD(loader)              ***    893d
  LIBRARIAN                           2579c  2677c
si1
  SRMDAM(srmdammodule)                961d  1020d
  SRM_DRV(srm)                  ***    215d
si2
  SRMDAM(srmdammodule)                 92c   603c   912d   924c   934c   943c   962d  1021d  1050c  1109c
  SRM_DRV(srm)                  ***    216d
sign
  LIBRARIAN                           156d   181c   191c   193c   209c   215c
signbit
  MOREFSYS(mfs)                        33d   412c   423c
signed16
  CS80(cs80)                           71d    88d    95d    96d   101d   104d   105d   106d   107d   114d   393c   427c   439c   451c
                                      579c   580c
  CS80(cs80dsr)                       788d   789d
  CS80(cs80dvr)                 ***  1036d
  MINIT(cs80ir)                        524d   532d   535d   560d   564d   585c   586c   625d   635d   639c   818d
  MINIT(qminit)                 ***  1110d
  TAPEBKUP(cs80tbdvr)           ***   704d
  TAPEBKUP(cs80tbr)                    50c    51c    65c    66c
signed4
  CTABLE(brstuff)                      942d   962d   963d
signed8
  CS80(cs80)                           72d    99d   124d   209d   219d   223d
  CTABLE(brstuff)                      943d   957d   958d   959d   961d   971d
  MINIT(cs80ir)                 ***    784d
  TAPEBKUP(cs80tbdvr)                  706d   710d
silently
  TAPEBKUP                             963d  1239c  1248c  1269c
simon_dma
  AMIGO(amigodvr)                      515c   681c
  DISCHPIB(dischpib)                   193d   237d   240c
simon_no_dma
  AMIGO(amigodvr)               ***    604c
  CS80(cs80dvr)                 ***   1388c
  DISCHPIB(dischpib)                   192d   230d   233c
simple_init
  KERNEL(general_0)                    803d  1235c
sin_angle
  DGL_POLY(dgl_poly)                 20880d 20900c 20903c
sindx
  SRMDAM(srmdammodule)                 211d   223c   224c   225c   226c   227c   229c   230c   231c   234c   235c   243c   245c   262c
                                       652d   673c   674c   678c   680c   682c   689c   698c   709c   721c   722c   724c   729c   734c
                                       735c   738c   746c   759c   763c   773c   774c   776c   779c   780c   781c   791c   800c  1523d
                                      1535c  1580c  1585c  1586c  1599c  1601c  1608c  1610c
sinit
  SRM_DRV(srm)                  ***    900d
sint
  INITLOAD(loader)                     857d   932d  1460c  1464c
  LIBRARIAN                            176c  1549c  1599c  1643c  1644c  1646c  1822c  1828c  1925c  1929c  2393c  2486c  2674c
  MOUSE(mouse)                          37d    46d    47d    48d    68d   132d
sinx
  INIT(fs)                      ***    793d
  UCSD_DAM(ucsdmodule)                 184d   199c   204c   239c
sinx_table
  GLE_STEXT(gle_stext)          ***   4039c
  GLE_TYPES(gle_types)          ***   1168d
siny_table
  GLE_STEXT(gle_stext)          ***   4042c
  GLE_TYPES(gle_types)          ***   1169d
siz
  LIBRARIAN                            417d   439d   565d   619d   626d   651d   710d
```

```
size
  EPROMS(eproms)                        42d    66c    67c    68c    69c    70c    71c
  ETU                                  487d   506c   513c   565d   572c   578c   593c
  GLE_UTLS(gle_utls)                 21008d 21009d 21038d 21064c 21066c 21069d 21082c 21083c 21094c 21100c
  HEAPT(hpm)                           100d   104c   105c   106c   110d   113c   114c   116c   217d   231c   233c   236c   237c   244c
                                       245c   254c   255c
  INIT(fs)                             770d   786d   787d
  INITLOAD(loader)                    1084d  1089d  1111d  1113c  1114c  1118d  1120c  1253d  1265c  1267c  1282d  1291c  1292c  1567d
                                      1570c
  KEYS(keys)                            83c   112c   114c   115c   366c   409c   563c
  LIBRARIAN                            125d   127c   137d   139c   439d   478d   480c   481c   482c   483c   484c   485c   510d   514c
                                       517c   528c   530c   565d   566c   567c   568c   673c   678c   689c   692c   699c   703c   814c
                                       850c   861c   865c   869c   875c   881c   893c   899c   905c   910c   913c   918c   923c   972c
                                      1005c  1029c  1051c  1078c  1113c  1115c  1122c  2740d  2742c
  SEGMENTER(asm)                        44d    45d   292d   294c   298c
  SYSDEVS(sysdevs)                     240d   529c   533c   536c   543c   550c   560c   561c   563c   564c   575c   640c
  UCSD_DAM(ucsdmodule)                 190d   194c   197c   198c   202c   203c
size1
  GLE_UTLS(gle_utls)                 21011d 21118d 21126c 21130c 21131c
size16
  HPHIL(hphil)                  ***    115c
  MOUSE(mouse)                          88c    92c    93c   159c
  SYSDEVS(sysdevs)              ***    316d
size2
  GLE_UTLS(gle_utls)                 21012d 21119d 21126c
size_from_are_you_alive
  SRM_DRV(srm)                  ***     89d
size_from_cat
  SRM_DRV(srm)                  ***     90d
size_from_catprotect
  SRM_DRV(srm)                  ***     91d
size_from_change_vol_label
  SRM_DRV(srm)                  ***     93d
size_from_changeprotect
  SRM_DRV(srm)                  ***     92d
size_from_close
  SRM_DRV(srm)                  ***     94d
size_from_copy
  SRM_DRV(srm)                  ***     95d
size_from_create
  SRM_DRV(srm)                  ***     96d
size_from_createlink
  SRM_DRV(srm)                  ***     97d
size_from_flock
  SRM_DRV(srm)                  ***     98d
size_from_funlock
  SRM_DRV(srm)                  ***     99d
size_from_gang_cleanup
  SRM_DRV(srm)                  ***    100d
size_from_gang_error
  SRM_DRV(srm)                         101d  1439c  1442c  1449c
size_from_info
  SRM_DRV(srm)                  ***    102d
size_from_init_vol
  SRM_DRV(srm)                  ***    103d
size_from_open
  SRM_DRV(srm)                  ***    104d
size_from_position
  SRM_DRV(srm)                  ***    105d
size_from_purge
  SRM_DRV(srm)                  ***    106d
size_from_read
  SRM_DRV(srm)                         107d  1449c
```

```
size_from_set_date
  SRM_DRV(srm)                  ***    109d
size_from_set_eof
  SRM_DRV(srm)                  ***    108d
size_from_volstatus
  SRM_DRV(srm)                  ***    110d
size_from_write
  SRM_DRV(srm)                  ***    111d
size_from_xchg_data
  SRM_DRV(srm)                  ***    112d
size_from_xchg_open
  SRM_DRV(srm)                  ***    113d
size_to_are_you_alive
  SRM_DRV(srm)                          57d  1534c
size_to_cat
  SRM_DRV(srm)                          58d  1561c
size_to_catprotect
  SRM_DRV(srm)                          59d  1599c
size_to_change_vol_label
  SRM_DRV(srm)                          62d  1666c
size_to_changeprotect
  SRM_DRV(srm)                          60s  1638c
size_to_close
  SRM_DRV(srm)                          63d  1687c
size_to_copy
  SRM_DRV(srm)                          64d  1712c
size_to_create
  SRM_DRV(srm)                          65s  1750c
size_to_createlink
  SRM_DRV                       ***   1799c
  SRM_DRV(srm)                  ***     67d
size_to_flock
  SRM_DRV                       ***   1889c
  SRM_DRV(srm)                  ***     68d
size_to_funlock
  SRM_DRV                       ***   2051c
  SRM_DRV(srm)                  ***     69d
size_to_gang_cleanup
  SRM_DRV                       ***   1870c
  SRM_DRV(srm)                  ***     70d
size_to_info
  SRM_DRV                       ***   1851c
  SRM_DRV(srm)                  ***     71d
size_to_init_vol
  SRM_DRV(srm)                  ***     72d
size_to_open
  SRM_DRV                       ***   1916c
  SRM_DRV(srm)                  ***     73d
size_to_position
  SRM_DRV                       ***   1949c
  SRM_DRV(srm)                  ***     74d
size_to_purge
  SRM_DRV                       ***   1977c
  SRM_DRV(srm)                  ***     75d
size_to_read
  SRM_DRV                       ***   2005c
  SRM_DRV(srm)                  ***     76d
size_to_set_date
  SRM_DRV(srm)                  ***     78d
size_to_set_eof
  SRM_DRV                       ***   2030c
  SRM_DRV(srm)                  ***     77d
```

```
size_to_volstatus
  SRM_DRV                       ***   2069c
  SRM_DRV(srm)                  ***     79d
size_to_write
  SRM_DRV                       ***   2091c
  SRM_DRV(srm)                  ***     80d
size_to_xchg_data
  SRM_DRV(srm)                  ***     81d
size_to_xchg_open
  SRM_DRV                       ***   1832c
  SRM_DRV(srm)                  ***     82d
sizechk
  LIFDAM(lifmodule)                     99d   102c
skip
  SRMDAM(srmdammodule)                 1528d  1550c  1555c  1573c  1576c
skipfor
  IOLIB(general_2)                      821d  1070d
skipping
  M68KSYS(ci)                           982d  1061c  1074c  1076c  1081c  1086c  1145c
sl
  ETU                                   176d   179c   180c   182c
  GLE_UTLS(gle_utls)                  21010d 21106d 21113c 21114c
  LIFDAM(lifmodule)                     134d   139c   140c   144c   148c   151c   162d   167c   171c   176c   177c   180c   181c
  SEGMENTER(asm)                         54d   131c   266c
slast
  UCSD_DAM(ucsdmodule)                  185d   200c   205c   239c
slink
  CTABLE(ctr)                           448d   454d   461c   464c
  CTABLE(scanstuff)             ***    1061d
slist
  DGL_INQ(dgl_inq)                     6012d  6088d  6441c  6450c  6461c  6475c  6488c  6503c  6514c  6530c  6543c
  LIBRARIAN                            1716d  1729c  1754c  1771c  1781c  1786c  1997d  2015c  2039c  2059c  2070c  2080c  2121c
slock
  SRM_DRV                       ***    1886c
  SRM_DRV(srm)                  ***     902d
slop
  INITLOAD(loader)                     1138d  1147c  1148c  1156c  1158c
slowterm
  CRT(crt)                      ***     113d
  GCRT(crtb)                    ***      46d
  SYSDEVS(sysdevs)              ***      69d
slp
  INITLOAD(loader)              ***     908d
  LIBRARIAN                            1729c  2015c  2121c
small
  KEYS(keys)                            376d   442c   489c   491c   510c   512c
smallest
  HEAPT(hpm)                             55d    59d    60d   122d   123d   143c   144c   230c   293c   310c
smallkbd
  A804XDVR(a804xdvr)                    216c   223c
  KEYS(keys)                    ***     442c
  NONUSKBD1(non_us_kbd1)                116c   165c
  SYSDEVS(sysdevs)              ***     145d
smallsize
  HEAPT(hpm)                             70d   104c   114c   116c   122d   143c
smh
  SRM_DRV(srm)                         1477d  1482c
sn
  CTABLE(ctr)                           339d   812d   815c   816c
so
  CS80(cs80)                            524d   530c   531c   532c
soft_clip_cpx
  GLE_TYPES(gle_types)          ***    1189d
```

```
soft_clip_cpy
  GLE_TYPES(gle_types)          ***   1190d
soft_clip_savex0
  GLE_TYPES(gle_types)          ***   1183d
soft_clip_savex1
  GLE_TYPES(gle_types)          ***   1184d
soft_clip_savey0
  GLE_TYPES(gle_types)          ***   1185d
soft_clip_savey1
  GLE_TYPES(gle_types)          ***   1186d
soft_clip_switch
  GLE_TYPES(gle_types)          ***   1187d
soft_font_ptr
  GLE_HPGL(gle_hpgl_out)        ***   7561c
  GLE_RGL(gle_ras_out)          ***   8427c
  GLE_TYPES(gle_types)          ***   1174d
soft_text_temp1
  GLE_TYPES(gle_types)          ***   1176d
soft_text_temp2
  GLE_TYPES(gle_types)          ***   1177d
softfield
  CRT(crt)                            54d   274c   286c   680c
  GCRT(crtb)                    ***   101d
softkey
  GLE_HPGLI(gle_hpgl_in)        ***  18236d
softwc
  GLE_RGL(gle_ras_out)          ***   8063d
solid_linestyle
  DGL_VARS(dgl_vars)            ***   1206d
soper
  SRM_DRV                       ***   1911c
  SRM_DRV(srm)                  ***    904d
sort
  DGL_POLY(dgl_poly)                20547d 20757c 20814c 20815c
sortasc
  INITLOAD(loader)                    907d  1035d  1041d
  LIBRARIAN                          1736c  2022c
sortlen
  LIBRARIAN                          1719d  1748c  1754c  1761c  1768c  1777c  1796c  2000d  2034c  2039c  2046c  2056c  2066c  2107c
sortlist
  INITLOAD(loader)                    877d  1041d
sortlistptr
  INITLOAD(loader)                    877d   908d
  LIBRARIAN                          1716d  1997d
source
  LIBRARIAN                           546d   758c   805c   900c
source_blk_addr
  TAPEBKUP(cs80tbr)                    32d   209d   243c
source_block_address
  TAPEBKUP                           1262c  1344c  1382c
  TAPEBKUP(cs80tbcvr)                 333d   900c
source_block_size
  TAPEBKUP                           1221d  1302c  1382c
source_file_id
  SRM_DRV(srm)                        474d  1716c
source_is_a_7914
  TAPEBKUP                           1220d  1258c  1298c  1324c  1374c
source_is_a_tape
  TAPEBKUP                           1219d  1237c  1301c  1368c  1385c
source_lun
  TAPEBKUP                           1217d  1294c  1295c  1297c  1310c
source_offset
  SRM_DRV(srm)                        475d  1717c

source_size
  TAPEBKUP                           1222d  1268c  1282c  1283c  1303c  1323c  1381c
source_uep
  TAPEBKUP                           1218d  1239c  1266c  1295c  1296c  1298c  1312c  1369c  1386c
  TAPEBKUP(cs80tbdvr)                 335d   693d   724c   725c   730c   731c   735c   740c   758c   759c   813c   814c   816c   879c
                                      880c   884c   886c   889c   900c   907c
  TAPEBKUP(cs80tbr)                    31d   209d   239c   240c   249c   250c   251c
sourceaddr
  SRM_DRV(srm)                  ***   301d
sourceblock
  INITLOAD(loader)              ***   959d
  LIBRARIAN                           379c  1394c  1976c
sourcefile
  ETU                                 350d   375c   845d   910c   918c   920c   922c   933c   945c   996c  1013c  1072c  1091c
sourcesize
  INITLOAD(loader)              ***   960d
  LIBRARIAN                           378c   380c  1388c  1391c  1977c
sourcetype
  MOREFSYS(mfs)                        30d    44d
sp
  CONVERT(convert_text)                46d    64c    65c    66c    67c
  INITLOAD(loader)                   1422d  1441c  1442c  1443c
  LIBRARIAN                           554d   572c  1654d  1694c  1695c  1697c  1702c
space
  LIBRARIAN                           467d   500c   503c   698c   948c   968c  1007c  1036c  1043c  1059c
  LIFOAM(lifmodule)                   676d   741c   744c   746c   747c   781c   839c   867c   892c   893c   898c   904c   929d   931c
                                      932c   938c   939c
  SEGMENTER(asm)                      176d   186c   187c   189c
spacerec
  LIFOAM(lifmodule)                    70d   676d   681d   716d   929d
spacewait
  ETU                                 287d  1207c
spanish_eur_kbd
  A804XDVR(a804xdvr)            ***   189d
  NONJSKBD2                           199c   212c   221c
  SYSDEVS(sysdevs)              ***   148d
spanish_kbd
  A804XDVR(a804xdvr)            ***   194d
  NONJSKBD1(non_us_kbd1)              117c   192c
  NONJSKBD2                     ***   200c
  SYSDEVS(sysdevs)              ***   149d
spanish_latin_kbd
  A804XDVR(a804xdvr)            ***   192d
  NONJSKBD2                           199c   219c
  SYSDEVS(sysdevs)              ***   148d
spare
  CRT(crt)                            107d   153d
  GCRT(crtb)                    ***    86d
  SYSDEVS(sysdevs)              ***   101d
spare_block
  MINIT(cs80ir)                       569d   847d   868c
  MINIT(qminit)                 ***  1234c
spareblock
  MINIT(cs80ir)                       859d   864c
sparetrack
  MINIT(xminit)                      1439d  1448c
sparing_required
  MINIT(qminit)                      1109d  1215c  1217c  1222c  1246c  1248c  1254c
sparing_tries
  MINIT(qminit)                      1110d  1221c  1222c  1224c
sparingmode
  MINIT(cs80ir)                       569d   847d   860d   865c
```

```
spd_message
  IOLIB(hpib_3)                  ***   2102c
  KERNEL(iodeclarations)         ***    309d
spdcontrol
  MINIT(xminit)                        1384d  1626c  1630c  1705c  1723c  1742c  1758c  1799c
spdword
  MINIT(asmr)                          1337d  1348d
spe_message
  IOLIB(hpib_3)                  ***   2100c
  KERNEL(iodeclarations)         ***    308d
special_key
  KEYS(keys)                           298c   440c
  SYSDEVS(sysdevs)               ***    258d
specialcrt1
  SYSDEVS(sysdevs)               ***    113d
specialcrt2
  SYSDEVS(syscevs)               ***    113d
specialkbd1
  SYSDEVS(syscevs)               ***    145d
specialkbd2
  SYSDEVS(sysdevs)               ***    145d
specials
  NONUSKBD2                             60d   147c   148c
specific_msus
  CTABLE                               1290d  1298c  1301c  1305c  1308c
specified_system_unit
  CTABLE                               1828c  1829c
  CTABLE(options)                ***     53d
splen
  INITLOAD                             1666d  1667d  1770c
spm
  LIFDAM(lifmodule)                      52d   292c   513c   521c
spoll
  IOLIB(hpib_3)                        2033d  2094d  2104c
spooling
  DGL_C_OUT(dgl_confg_out)             11266c 11379c 11429c
  DGL_HPGL(dgl_hpgl)             ***   17034c
  GLE_GEN(gle_gen)               ***    2135c
  GLE_HPGL(gle_hpgl_out)                7175c  7236c  7297c  7564c  7670c
  GLE_TYPES(gle_types)           ***    1076d
  LIB(dgl_lib)                         21214c 21267c
spos
  SRM_DRV                        ***    1946c
  SRM_DRV(srm)                   ***     906d
spptr
  MINIT(xminit)                        1411d  1777c  1789c  1793c  1808c  1809c
spt
  LIFDAM(lifmodule)                      53d   292c   513c   521c
sptable
  MINIT(xminit)                  ***    1409d
sptr
  CRT(crt)                              505d   528c   529c   530c   531c
  GCRT(crtb)                            373d   399c   400c   401c   402c
  GLE_HPGL(gle_hpgl_out)               7141d  7148c  7151c
  GLE_HPGLI(gle_hpgl_in)              18080d 18086c 18088c
  LIBRARIAN                            1717d  1736c  1737c  2008d  2022c  2023c
spurge
  SRM_DRV                        ***    1972c
  SRM_DRV(srm)                   ***     908d
sr
  CS80(cs80)                            502d   511c   512c   513c   514c   515c   516c
srcfid
  SRM_DRV(srm)                         1036d  1703d  1716c

srcoff
  SRM_DRV(srm)                         1037d  1704d  1717c
sread
  SRM_DRV                        ***    2002c
  SRM_DRV(srm)                   ***     910d
srec
  LIFDAM(lifmodule)                     676d   716d   718c   719c   722c   723c   724c   728c   730c   731c   732c   908c
srm
  CTABLE                               1256c  1415c
  CTABLE(brstuff)                ***    1003c
  CTABLE(ctr)                           256d   498c   673c
  SRMAM(srmammodule)             ***     35d
  SRMDAM(srmdammodule)           ***     37d
  SRM_DRV(srm)                   ***     27d
srm_card
  DC_DRV(init_dc)                ***    687c
  KERNEL(iodeclarations)         ***    330d
srm_cat_pass
  SRMDAM(srmdammodule)                  499d  1706c
srm_catalog
  SRMDAM(srmdammodule)                  562d  1700c
srm_change_name
  SRMDAM(srmdammodule)                 1170d  1764c
srm_clearunit
  SRMAM(srmammodule)                    144d   207c
srm_close_file
  SRMDAM(srmdammodule)                 1329d  1762c
srm_close_fileid
  SRMDAM(srmdammodule)                  331d   352c  1112c  1154c  1284c  1285c  1352c  1366c  1381c  1389c  1427c
srm_close_pathid
  SRMDAM(srmdammodule)                  347d   821c   844c   850c   893c  1205c  1239c  1240c  1305c  1382c  1428c  1433c  1697c  1707c
                                       1716c  1728c  1744c  1754c  1774c
srm_create_dir
  SRMDAM(srmdammodule)                  986d  1753c
srm_create_file
  SRMDAM(srmdammodule)                 1013d  1459c  1741c
srm_dam_name
  CTABLE(ctr)                           354d   673c
srm_dav
  CTABLE                               1209d  1416c  1447c  1504c  1620c
srm_default_dav
  CTABLE                         ***   1447c
  CTABLE(options)                ***    119d
srm_dup_link
  SRMDAM(srmdammodule)                 1210d  1766c
srm_get_dir_info
  SRMDAM(srmdammodule)                  381d   451c  1692c
srm_get_vol_date
  SRMDAM(srmdammodule)                 1438d  1791c
srm_get_vol_name
  SRMDAM(srmdammodule)                  446d   895c  1789c
srm_init
  SRMDAM(srmdammodule)           ***   1619c
  SRM_DRV(srm)                          999d  1212d
srm_inited
  SRM_DRV(srm)                         1126d  1214c  1219c
srm_lock_file
  SRMDAM(srmdammodule)                 1472d  1779c
srm_open_dir
  SRMDAM(srmdammodule)                  645d   882c  1185c  1224c  1228c  1690c  1703c  1712c  1723c  1738c  1750c  1770c
srm_open_file
  SRMDAM(srmdammodule)                  956d  1726c
srm_purge_file
  SRMDAM(srmdammodule)                 1263d  1355c  1372c  1384c  1394c  1430c  1466c  1760c  1783c
```

```
srm_purge_name
  SRMDAM(srmdammodule)              1245d  1773c
srm_read
  SRMMAM(srmammodule)                50d   196c
srm_set_pass
  SRMDAM(srmdammodule)              456d  1715c
srm_set_unit_prefix
  SRMDAM(srmdammodule)              875d  1785c
srm_stretch
  SRMDAM(srmdammodule)             1310d  1787c
srm_strip
  SRMDAM(srmdammodule)             1519d  1796c
srm_tm_name
  CTABLE(ctr)                       361d   673c
srm_unlock_file
  SRMDAM(srmdammodule)             1496d  1781c
srm_write
  SRMMAM(srmammodule)                97d   203c
srmam
  SRMMAM(srmammodule)                39d   177d
srmammodule
  SRMMAM(srmammodule)          ***   27d
srmdam
  SRMDAM(srmdammodule)               43d  1624d
srmdaminit
  SRMDAM                       ***  1834c
  SRMDAM(srmdammodule)               41d  1617d
srmdammodule
  SRMDAM                       ***  1832d
  SRMDAM(srmdammodule)         ***   28d
srmdate
  SRMDAM(srmdammodule)              356d   362c
srmnoce
  CTABLE                       ***  1806c
  INITLOAD(bootdammodule)           546d   778d   791c   792c   800c
  SRMDAM(srmdammodule)         ***  163c
  SRM_DRV(srm)                     1403d  1469c
srmsavesc
  SRMMAM(srmammodule)               184c   215c   216c
  SRMDAM(srmdammodule)             1639c  1813c  1814c
  SRM_DRV(srm)                      997d  1361c  1362c
srmsysprefix
  CTABLE                           1189d  1806c  1807c
srmtype
  INITLOAD(bootdammodule)           796d   800c
srp
  UCSD_DAM(ucsdmodule)               44d   439c   460c   661c
srq
  GLE_HPGLI(gle_hpgl_in)       ***  18237d
srq_line
  IOLIB(hpib_3)                ***  2068c
  KERNEL(iodeclarations)       ***   390d
srt
  SRM_DRV(srm)                     1479d  1485c
ss
  AMIGO(amigodvr)                   478c   529c   670c   890c
  AMIGO(csamigo)               ***  114d
  MINIT(hminit)                ***  481c
ss9895_map
  AMIGO(csamigo)                    194d   210c
sset
  A804XDVR(a804xdvr)                329d   346c   348c
ssetdate
  SRM_DRV(srm)                 ***   912d
```

```
sseteof
  SRM_DRV                      ***  2027c
  SRM_DRV(srm)                 ***   914d
ssize
  DGL_INQ(dgl_inq)                 6009d  6085d  6439c  6458c  6473c  6485c  6500c  6511c  6527c  6540c
  LIFDAM(lifmodule)                  72d   718c   743c   746c   747c   754c   756c   773c   781c   783c   795c   871c   891c   892c
                                    898c   904c   905c   939c
  UCSD_DAM(ucsdmodule)              186d   198c   202c   203c   238c   239c
ssm
  CS80(cs80)                        540d   547c   548c   549c   550c
sstart
  LIFDAM(lifmodule)                  71d   719c   774c   796c   813c   815c   826c   839c   858c   870c   885c   896c   938c
st
  INIT(misc)                   ***  367d
  INITLOAD                         1727d  1730c  1731c
stack_ptr
  DGL_POLY(dgl_poly)              20051d 20939c 21063c 21066c 21108c 21111c 21139c 21142c 21170c 21173c
standard_tc
  DISCHPIB(dischpib)                208d   293c   316c   364c   452c   457c
standardam
  INIT(misc)                        534d   582c   720c
star
  AMIGO(csamigo)               ***  104d
start
  ETU                               487d   501c   502c   506c   517c  1160d
  GLE_UTLS(gle_utls)               21078d 21093c 21100c
  INITLOAD(loader)                  1089d  1567d  1570c
  LIBRARIAN                         749d   760c   768c   772c   784c
  TAIL                               34d    50c    55c    61c    68c    85c    86c
start_add
  MIUI                              411d   445d
start_alternate
  SRMDAM(srmdammodule)              492c   529c   588c   732c   825c   980c  1007c  1081c  1086c  1095c  1159c  1202c  1203c  1235c
                                   1236c  1257c  1296c  1367c  1392c  1404c  1420c  1424c
  SRM_DRV(srm)                 ***  234d
start_digitize
  GLE_GENI(gle_geni)           ***  3076c
  GLE_HPGLI(gle_hpgl_in)       ***  18371c
  GLE_KNOB(gle_knob_in)        ***  18222c
  GLE_TYPES(gle_types)         ***  1210d
start_name_sets
  SRM_DRV                           1795c  1796c  1913c  1974c
  SRM_DRV(srm)                      379d   402d   425d   505d   523d   540d   654d   697d  1558c  1596c  1634c  1635c  1745c  1747c
start_of_buffer
  GLE_HPGL(gle_hpgl_out)           7021d  7048c  7112c  7524c
start_protect_code_sets
  SRM_DRV(srm)                 ***  615d
start_root
  SRMDAM(srmdammodule)              737c   741c  1123c  1129c  1134c  1143c  1146c  1164c  1291c  1391c
  SRM_DRV(srm)                 ***  233d
start_tfr_in
  DC_DRV(extdc)                ***  211d
  DC_DRV(intdc)                ***  551c
start_tfr_out
  DC_DRV(extdc)                ***  212d
  DC_DRV(intdc)                ***  556c
startaddr
  INIT(ldr)                        2335c  2337c  2361c
  INITLOAD                         1719c  1720c
  INITLOAD(loader)                 1017d  1443c  1447c
  M68KSYS(ci)                       883c   885c   887c   925c
  SEGMENTER(asm)                    264c   267c
startblock
  LIBRARIAN                        2830d  2857c  2867c  2869c
```

```
startcase
  LIBRARIAN                          298d  1216c  1218c
startcmd
  HPHIL(hphil)                        39d    83c
startdefs
  INIT(ldr)                  ***     2478c
  INITLOAD(loader)                   1076d  1428c  1471c  1479c  1490c  1491c  1641c
  SEGMENTER(asm)             ***      323c
startglobal
  INIT(ldr)                  ***     2470c
  INITLOAD(loader)                   1064d  1405c  1633c
  LIBRARIAN                          1669c  1708c  1971c  2976c  3203c  3571c
  SEGMENTER(asm)             ***      205c
startgvr
  LIBRARIAN                           53d   358c   365c  1666c  1698c  1702c  1906c  1908c  1967c
startgvrmod
  LIBRARIAN                           54d  1666c  1701c  1909c
startindx
  SRMDAM(srmdammodule)               220d   223c   230c   231c
starting
  DGL_POLY(dgl_poly)               20547d 20566c 20568c
startoffset
  SRMAM(srmammodule)                  51d    63c    98d   110c
startread
  AMIGO(amigodvr)                    564c   592c   621c
  BUBBLES(bubble)            ***      70c
  CRT(crt)                           434c   446c
  CS80(cs80dvr)                     1106c  1262c  1339c  1385c
  F9885(f9885dvr)                    167c   277c   298c
  GCRT(crtb)                         304c   316c
  INITLOAD(mini)                     425c   438c   505c   526c
  INITLOAD(sysglobals)       ***      67d
  KEYS(keys)                          99c   136c
  UNITIO(uio)                ***      84c
startreloc
  INIT(ldr)                         2431c  2476c
  INITLOAD(loader)                  1063d  1405c  1516c  1534c  1535c  1563c  1599c  1639c
  LIBRARIAN                         1669c  1707c  1969c  2976c  3197c  3570c
  SEGMENTER(asm)                     238c   304c   322c   329c
startstate
  MOREFSYS(mfs)                      337d   380c
startstream
  M68KSYS(ci)                         73d    93d
starttext
  LIBRARIAN                         2169d  2543c  2626c
startwrite
  AMIGO(amigodvr)                    564c   592c
  BUBBLES(bubble)            ***      87c
  CRT(crt)                           452c   496c
  CS80(cs80dvr)                     1288c  1339c  1385c
  F9885(f9885dvr)                    172c   223c   277c   298c
  GCRT(crtb)                         323c   364c
  INITLOAD(mini)                     427c   443c   505c   526c
  INITLOAD(sysglobals)       ***      67d
  KEYS(keys)                 ***      93s
  UNITIO(uio)                ***      63c
stat
  DGL_C_OUT(dgl_confg_out)         11029d 11109c 11110c 11133c 11134c
stat5datatype
  HPHIL(hphil)                        45d   303d
```

```
statbyte
  A804XDVR(a804xdvr)                  56d    71d    74d   119c   134c   326d   340c   372c   374c
  BAT(bat)                            44d    55c    63c
  HPHIL(hphil)                        71d    96d   282d   302d   316c
  MOUSE(mouse)                        75d   110c   111c   118c   119c   129d   140c   145c   163c   170c
  SYSDEVS(sysdevs)                    37d   402d
state
  LIBRARIAN                          749d   759c   761c   765c   768c   772c   784c
  SEGMENTER(asm)                     141d   142c   144c   231d   250c   290d   311c   312c   317d   327c   328c
  SRMDAM(srmdammodule)               210d   241c   247c   254c   259c   269c   278c   280c
state_ptr
  SEGMENTER(asm)                      57d    61d    73d   290d   317d
state_rec
  SEGMENTER(asm)                      57d    60d   141d   231d   311c   312c   327c   328c
states
  LIBRARIAN                          755d   758c
statetype
  SRMDAM(srmdammodule)               202d   210d
statrec
  MINIT(asmr)                ***    1324d
  MINIT(xminit)                     1400d  1406d
statreg
  EDRIVER(edriver)                    53d    67d    68d
status
  A804XDVR(a804xdvr)                  91d    94c    97c   100c   102c   103c   105c   106c   108c
  AMIGO(amigodvr)                    476c   527c   661c   801c
  AMIGO(csamigo)                     148d   304d
  CS80(cs80)                         342d   457d   469c
  CS80(cs80dsr)              ***     650c
  EDRIVER(edriver)                    68d   157c   167c   178c
  F9885(f9885dvr)                    111d   129c   131c   134c   196c   201c   204c
  GLE_HPGLI(gle_hpgl_in)           18228d 18265c 18268c 18271c
  GLE_RGL(gle_ras_out)              8117c  8264c  8389c
  GLE_TYPES(gle_types)       ***    1273d
  HPHIL(hphil)                       303d   305c   307c   311c   314c   322c   324c   336c   343c   349c
  KEYS(keys)                         191c   419c   426c   432c   486c   554c
  MINIT(asmr)                ***    1325d
  MINIT(hminit)                      316c   470c
  MINIT(xminit)                     1400d  1429c  1593c  1611c  1613c  1618c  1621c  1631c  1632c  1712c  1726c  1727c  1735c  1740c
                                    1760c  1761c  1801c  1802c
  NONUSKBD1(non_us_kbd1)     ***      42c
  SRMDAM(srmdammodule)               341c   399c   423c
  SRM_DRV(srm)                       325d  1131d  1134c  1135c  1467c  1469c  1543c
  SYSDEVS(sysdevs)           ***     262d
  TAPEBKUP(cs80tbdvr)                530c   758c
status5hook
  A804XDVR(a804xdvr)                  37d   102c   405c
  HPHIL(hphil)                       366c   367c
status5isr
  A804XDVR(a804xdvr)         ***      74d
  HPHIL(hphil)                       302d   367c
status6hook
  A804XDVR(a804xdvr)                  38d   103c   406c
  HPHIL(hphil)                       189c   200c   286c   310c   326c   332c   344c   368c
status_2_error
  AMIGO(amigodvr)            ***     885c
  AMIGO(csamigo)             ***      78d
```

```
status_bytes
  AMIGO(amigodvr)                  444d  476c  478c  482c  484c  509d  527c  529c  532c  534c  656d  661c  662c  792d
                                   801c  803c
  AMIGO(csamigo)                   148d  304d  312c  313c  317c  321c
  CS80(cs80)                       342d  457d  466c  467c
  CS80(cs80dsr)                    640d  650c  658c  682c  755c  757c  763c
  MINIT(hminit)                    302d  316c  317c  325c  470c  478c
  TAPEBKUP(cs80tbdvr)              520d  530c  538c  562c  637c  639c  645c  647c  653c  700d  758c  766c  794c  871c
                                   874c  878c  880c  886c
status_cmd_buf
  AMIGO(csamigo)                   306d  310c
status_def
  GLE_TYPES(gle_types)            1243d 1273d
status_mask
  CS80(cs80)                       347d  538d  549c
status_mask_type
  CS80(cs80)                       201d  211d  347d  538d  544d
  CS80(cs80dsr)             ***    795d
status_type
  AMIGO(amigodvr)                  444d  509d  656d  792d
  AMIGO(csamigo)                    95d  148d  157d  304d
  CS80(cs80)                       205d  342d  362d  457d
  CS80(cs80dsr)             ***    640d
  MINIT(hminit)             ***    302d
  TAPEBKUP(cs80tbdvr)              520d  700d
statusline
  SYSDEVS(sysdevs)                 248d  582c  585c  644c
statusreg
  A804XDVR(a804xdvr)                62d   94c  119c  134c  383c  386c
statustype
  A804XDVR(a804xdvr)                51d  114d  132d
  BAT(bat)                          39d   52d   60d
stdblk
  HEAPT(hpm)                        56d   57d   59d  231c
stdlists
  HEAPT(hpm)                ***     59d
stemp
  LIFDAM(lifmodule)                163d  183c  184c  189c
stepsize
  EDRIVER(edriver)                  83d  171c  178c  182c  186c
  EPROMS(eproms)                    17d   95c   98c
stio
  F9885(f9885dvr)           ***     88c
stil
  F9885(f9885dvr)           ***     88c
stime
  SYSDEVS(sysdevs)          ***    451d
stimulus
  CTABLE(scanstuff)         ***   1088d
stop
  CRT(crt)                          99d  145d
  GCRT(crtb)                ***     78d
  SYSDEVS(sysdevs)          ***     94d
stopaction
  KEYS(keys)                        59d  516c
stopgoing
  M68KSYS(ci)                      870d  891c  902c  904c  909c
stopline
  MIUI                             224d  260c  261c  270c
  TAPEBKUP                        1111d 1147c 1148c 1157c
storage
  CTABLE(scanstuff)                1087d 1121c
store_dit
  GLE_RGL(gle_ras_out)             8208c 8215c
```

```
stp
  INITLOAD(loader)                  902d 1187c 1265c 1289c 1290c 1291c 1309c 1348c
  LIBRARIAN                 ***     481c
str9
  INITLOAD(loader)                 1585d 1587d
str_count
  IOLIB(general_4)                 1242d 1614d 1617c 1622c 1623c 1629c 1633c
strcnt
  DGL_INQ(dgl_inq)                 6166d 6456c 6457c 6458c 6460c 6462c 6483c 6484c 6485c 6487c 6489c
stream
  M68KSYS(ci)                      1031d 1071c 1135c
streamchain
  M68KSYS(ci)                        58d   96c 1035c 1071c
streamdvr
  M68KSYS(ci)                       306d  547c  548c
streamfib
  M68KSYS(ci)                        54d   89c  321c  322c  330c  331c  342c  348c  349c  357c  367c  368c  372c  472c
                                    484c  499c  510c  514c  518c  545c  546c  550c  862c  863c  945c 1054c 1055c
streaming
  ETU                               106c  126c  133c  197c  206c  228c  552c  814c 1007c 1257c
  LIBRARIAN                          90c  273c 2093c 3406c 3683c
  M68KSYS(ci)                        74d   88d   90c 1076c 1081c 1142c
streamopen
  M68KSYS(ci)                       387d 1056c 1180c 1181c
streamsyntax
  M68KSYS(ci)                       403d  435c  530c  543c
streof
  MOREFSYS(mfs)                      56d   93c   94c   95c  377c
stretch
  UCSD_DAM(ucsdmodule)              352d  642c
stretchf
  LIFDAM(lifmodule)                1014d 1132c
stretchit
  ASCII(asciimodule)        ***     81c
  INIT(misc)                        433c  498c  617c
  INITLOAD(sysglobals)      ***    136d
  LIFDAM(lifmodule)         ***   1132c
  SRMDAM(srmdammodule)             1666c 1787c
  UCSD_AM(ucsd_am)          ***     57c
  UCSD_DAM(ucsdmodule)      ***    642c
strg
  MOREFSYS(mfs)                      30d   89c  544c
strin
  KEYS(keys)                         65d   66d
string2
  INITLOAD                  ***   1758d
  INITLOAD(bootdammodule)           538d  546d  547d  778d  782d  795d
string255
  CRT(crt)                  ***    505d
  CTABLE(ctr)                       318d  382d  422d  423d  506d
  ETU                       ***    422d
  GCRT(crtb)                ***    373d
  INIT(fs)                          764d  766d  817d  820d  825d  827d  830d  832d  836d
  INITLOAD(bootdammodule)           552d  588d  648d
  INITLOAD(sysglobals)               35d   37d
  LIBRARIAN                 ***   3302d
  LIFDAM(lifmodule)                  98d  109d 1060d
  M68KSYS(ci)                       308d  391d  455c
  MINIT(midecs)                      37d   41d
  MINIT(qminit)                     881d  932d  936d 1034d 1036d
  MIUI                               23d  124d  131d  290d
  MOREFSYS(mfs)                      21d   46d  533d  554d  563d  669d
  SRMDAM(srmdammodule)               98d  110d
  UCSD_DAM(ucsdmodule)      ***     42d
```

```
string255ptr
  INITLOAD(sysglobals)                37d   105d
  SRMDAM(srmdammodule)        ***     194d
string27
  MIUI                                102d   103d
string3
  A804XDVR(a804xdvr)                  58d   232d
string80
  ETU                                 40d  1148d
  INIT(fs)                           749d   807d
  INIT(ldr)                   ***   2412d
  INITLOAD                    ***   1727d
  INITLOAD(sysglobals)        ***     34d
  KEYS(keys)                          40d    42d
  LIBRARIAN                           41d    43d    63d    72d    98d   284d   737d   740d  2919d  3232d  3246d  3301d
  M68KSYS(ci)                         69d   174d   387d   408d   593d   625d   747d  1033d
  MINIT(xminit)               ***   1395d
  SYSDEVS(sysdevs)            ***     144d
  TAPEBKUP                            972d   987d   996d  1048d  1054d  1182d
  TAPEBKUP(cs80tbdvr)                 315d   452d   455d   715d
string80ptr
  SYSDEVS(sysdevs)                    144d   160d   161d
stringnlen
  MOREFSYS(mfs)                        29d    47d   396d   517d   535d
stringptr
  M68KSYS(ci)                         391d   393d
stringsp
  INITLOAD                           1667d  1668d
stringsysp
  INITLOAD                           1735d  1736d
stripname
  INIT(misc)                  ***     436c
  INITLOAD(sysglobals)        ***     137d
  LIFDAM(lifmodule)           ***    1163c
  SRMDAM(srmdammodule)        ***    1796c
  UCSD_DAM(ucsdmodule)        ***     664c
strng
  SRMDAM(srmdammodule)                 98d   105c   106c   110d   114c   115c
strptr
  CONVERT(convert_text)                44d    46d
  INIT(ldr)                          2385d  2395c  2396c
  INITLOAD(bootdammodule)             552d   692c
  LIBRARIAN                          1725d  1781c  1782c  1784c  2006d  2072c  2073c  2075c  2076c
  M68KSYS(ci)                         309d   363c
strrec
  UCSD_DAM(ucsdmodule)                 41d    44d
strsize
  CONVERT(convert_text)                42d    43d    60c
strtoany
  LIFDAM(lifmodule)                   109d  1136c
strtolifname
  LIFDAM(lifmodule)                   160d   636c   936c  1077c  1165c
strtopac
  LIFDAM(lifmodule)                    98d   172c   178c   511c  1141c
strtopaoc16
  SRMDAM(srmdammodule)                110d   127c   128c   165c   303c   484c   489c   524c   584c   979c  1004c  1006c  1040c  1195c
                                     1201c  1233c  1234c  1256c  1299c  1358c
stsmsk
  CS80(cs80)                          544d   549c
stupid
  CRT(crt)                    ***     112d
  GCRT(crtb)                  ***      45d
  SYSDEVS(sysdevs)            ***      69d
```

```
stype
  INIT(fs)                    ***     766d
  MOREFSYS(mfs)                        44d    64c    75c    89c
sub
  LIBRARIAN                           156d   158c  1482d  1486d  1492c  1495c  1505c  1523c  1547c  1565c  1566c  1574c  1579c
sub_poly_count
  DGL_POLY(dgl_poly)                20925d 20955c 20968c
sub_poly_start
  DGL_POLY(dgl_poly)                20924d 20953c 20964c 20965c 20982c
subit
  INITLOAD(loader)            ***     865d
  LIBRARIAN                           191c   193c  1505c  1523c  1574c  1579c  2395c
subset80
  CS80(cs80)                  ***      81d
  MINIT(qminit)                       973c  1063c  1074c  1162c
success
  SRMDAM(srmdammodule)        ***    1479c
  SRM_DRV(srm)                ***     636d
suffix
  INIT(fs)                    ***     756d
  LIFDAM(lifmodule)                   142c   168c
suffixtable
  ASCII                       ***     320c
  INIT(misc)                          701c   705c   714c   722c   726c
  INITLOAD(sysglobals)        ***     263d
  LIFDAM(lifmodule)                   147c   148c   152c   176c   180c   187c   188c
  UCSD_AM(ucsd_am)            ***     270c
suffixtype
  INITLOAD(sysglobals)                 36d   169d
suftableptrtype
  INITLOAD(sysglobals)                174d   263d
suftabletype
  INITLOAD(sysglobals)                169d   174d
sunlock
  SRM_DRV                     ***    2048c
  SRM_DRV(srm)                ***     916d
sup_dev_type
  MIUI                                 15d    20d   319d
suppress_release_timeout
  CS80(cs80)                  ***     231d
surface_ert
  MINIT(cs80ir)               ***     551d
surface_mode
  AMIGO(amigodvr)             ***     737c
  AMIGO(csamigo)                      139d   183d   185c   291c   349c   371c
suv
  TAPEBKUP(cs80tbr)                    63d    65c    66c    67c
suv_cmd_pair
  MINIT(cs80ir)                       583d   585c   586c   612c   666c   692c   718c   750c   790c   835c   863c
  TAPEBKUP(cs80tbr)                    48d    50c    51c    90c   120c   148c   171c   198c   274c
sva
  CS80(cs80)                          572d   583c   584c
  TAPEBKUP(cs80tbr)                   117d   125c   126c
sva_type
  CS80(cs80)                           86d   115d   215d   217d   221d   222d   224d   225d   572d
  TAPEBKUP(cs80tbdvr)         ***     338d
  TAPEBKUP(cs80tbr)                    28d   117d   228d   231d   255d
svdate
  LIFDAM(lifmodule)                   324d   327c   339d   344c
svol
  SRM_DRV                     ***    2066c
  SRM_DRV(srm)                ***     918d
svol_bytes
  CTABLE(ctr)                         312d   595d   600c   658c   693c   694c   706c   707c   716c   729c   746c
```

```
sw
  INITLOAD(loader)                  892d  1573c
  LIBRARIAN                        2373c  2436c  2494c  2507c  2568c  2600c  2670c  2671c  2672c
  UCSD_DAM(ucsdmodule)              188d   252c   256c   257c   382d   386c   389c   390c
swap
  ETU                        ***    386d
swap8
  UCSD_DAM(ucsdmodule)               55d    97c    98c   100c   102c   142c   152c   162c   163c   215c   220c   222c   228c   229c
                                    233c   245c   246c   253c   254c   273c   295c   336c   343c   361c   362c   365c   368c   369c
                                    370c   371c   387c   392c   416c   417c   418c   482c   488c   489c   490c   495c   517c   518c
                                    519c   536c   547c   548c   549c   586c   587c   596c   598c   599c   601c   612c   616c   620c
swapem
  UCSD_DAM(ucsdmodule)               52d    93c   479c   526c   558c
swapk
  UCSD_DAM(ucsdmodule)               56d    95c   247c   291c   313c   484c   545c   546c   595c
swedish_kbd
  A804XDVR(a804xdvr)                188d   193d
  NONUSKBD1(non_us_kbd1)     ***    176c
  NONUSKBD2                          34d    36d   128c   152c
  SYSDEVS(sysdevs)           ***    149d
swiss_fr_kbd
  A804XDVR(a804xdvr)         ***    186d
  NONUSKBD2                         191c   206c
  SYSDEVS(sysdevs)           ***    147d
swiss_gr_kbd
  A804XDVR(a804xdvr)         ***    191d
  NONUSKBD2                         191c   206c
  SYSDEVS(sysdevs)           ***    147d
switch_xy
  DGL_POLY(dgl_poly)               20435d 20449c
switchvols
  SWVOL                      ***      3d
sword
  INITLOAD(loader)                  857d   932d
  LIBRARIAN                        2440c  2488c  2561c  2667c
swrite
  SRM_DRV                    ***   2088c
  SRM_DRV(srm)               ***    920d
sx
  DGL_POLY(dgl_poly)               20552d 20561c 20563c 20568c 20571c 20574c 20575c
sxmax
  GEN(dgl_gen)                      3399d  3421c  3425c
sxmin
  GEN(dgl_gen)                      3398d  3420c  3424c
sy
  DGL_POLY(dgl_poly)               20553d 20561c 20563c 20577c 20578c 20579c
symax
  GEN(dgl_gen)                      3401d  3421c  3427c
symbol
  CTABLE(ctr)                        316d   382d   397c
  INITLOAD(loader)                   854d   855d
  SEGMENTER(asm)                     109d   125c   278d   281c   284c
  SEGMENTER(segmenter)                33d    35d
symbolptr
  INITLOAD(loader)                   855d   901d  1038d
  LIBRARIAN                          195c   196c  1172c  1188c  1767c
symboltable
  INITLOAD(loader)                   851d   852d
symfid
  M68KSYS(ci)                         47d   631c   636c
symin
  GEN(dgl_gen)                      3400d  3420c  3426c
symptr1
  INITLOAD(loader)                  1249d  1269c  1270c  1272c  1273c
```

```
symptr2
  INITLOAD(loader)                  1250d  1266c  1273c  1274c
symtable
  INITLOAD(loader)           ***    1213d
symtableptr
  INITLOAD(loader)                   852d   902d  1172d  1213d  1218d
  LIBRARIAN                  ***    1181c
sync_bit_not_received_in_time
  AMIGO(amigodvr)            ***     810c
  AMIGO(csamigo)             ***      75d
syndrome
  AMIGO(amigodvr)            ***     837c
  AMIGO(csamigo)                     149d   328d
syndrome_bytes
  AMIGO(amigodvr)                    793d   835c   837c
  AMIGO(csamigo)                     149d   328d   335c
syndrome_cmd_buf
  AMIGO(csamigo)                     330d   334c
syndrome_type
  AMIGO(amigodvr)            ***     793d
  AMIGO(csamigo)                     117d   149d   328d
syp
  CTABLE(ctr)                        397c   398c
  INIT(idr)                         2396c  2496c
  INITLOAD(loader)                   901d  1270c  1442c  1455c  1524c  1525c
  LIBRARIAN                          328c   329c   354c  1278c  1330c  1331c  1357c  1577c  1578c  1697c  1702c  1781c  1782c  1787c
                                    1895c  1897c  1901c  1939c  1940c  2051c  2059c  2073c  2082c  2087c  2092c  2101c  2343c  2528c
                                    2529c  2615c  2619c  2620c  2753c  2788c  2894c  2975c  3001c  3002c  3236c  3241c  3568c  3569c
  SEGMENTER(asm)                     125c   126c
syscloc
  CLOCK(clock)               ***      76c
  SYSDEVS(sysdevs)                   383d   457d   460c
syscom
  CRT(crt)                           185c   666c   719c   720c
  GCRT(crtb)                         502c   568c
  M68KSYS(ci)                        462c   493c   505c  1089c
  MINIT(minit)               ***     158c
  SYSDEVS(sysdevs)                   116d   632c
sysdate
  LIBRARIAN                  ***    3712c
  LIFDAM(lifmodule)          ***     117c
  M68KSYS(ci)                        123c   155c   255c
  MIUI                       ***      61c
  SYSDEVS(sysdevs)                   384d   463d
  UCSD_DAM(ucsdmodule)               415c   492c
sysdefs
  CTABLE(ctr)                ***     390c
  INIT(idr)                         2358c  2432c
  INITLOAD(loader)                  1071d  1127c  1133c  1450c  1451c  1480c  1481c  1600c  1655c
  M68KSYS(ci)                ***     922c
  SEGMENTER(asm)                     118c   149c   164c   191c
sysdeftable
  INITLOAD(loader)                  1101d  1655c
sysdev_init
  SYSDEVS                    ***     663c
  SYSDEVS(sysdevs)                   364d   597d
sysdevices
  SYSDEVS                    ***      28d
```

```
sysdevs
  A804XDVR(a804xdvr)              ***      31d
  BAT                             ***      99d
  BAT(bat)                        ***      33d
  CLOCK(clock)                    ***      33d
  CRT(crt)                        ***      33d
  C_HOOK                          ***      35d
  DGL_C_OUT(dgl_confg_out)        ***   11004d
  DGL_RAS(dgl_raster)             ***   17012d
  ETU                             ***      26d
  GCRT(crtb)                      ***      33d
  GLE_RGL(gle_ras_out)            ***    8004d
  HPHIL(hphil)                    ***      30d
  KEYS(keys)                      ***      33d
  LIB(dgl_lib)                    ***   20097d
  LIBRARIAN                       ***      27d
  LIFDAM(lifmodule)               ***      26d
  M68KSYS(ci)                     ***      33d
  MINIT(mminit)                   ***     111d
  MIUI                            ***      11d
  MOREFSYS(mfs)                   ***      10d
  MOUSE(mouse)                    ***      29d
  NONUSKBD1(non_us_kbd1)          ***      31d
  NONUSKBD2                       ***      30d
  PRINTER(prtdvr)                 ***      37d
  SYSDEVS                         ***     661d
  SYSDEVS(sysdevs)                ***      29d
  TAIL                            ***      32d
  TAPEBKUP                        ***     953d
  UCSD_AM(ucsd_am)                ***      29d
  UCSD_DAM(ucsdmodule)            ***      29d
sysescapecode
  INIT(misc)                             234c   241c
  INITLOAD(sysglobals)            ***    239d
  M68KSYS(ci)                            859c   866c
  TAPEBKUP                        ***   1170c
sysfile
  INIT(misc)                             726c   727c
  INITLOAD(bootdammodule)         ***    671c
  INITLOAD(sysglobals)            ***     52d
  LIBRARIAN                       ***   3140c
sysfilenames
  M68KSYS(ci)                             41d    55d   581d
sysfiles
  M68KSYS(ci)                             39d    40d    41d   586d   587d   591d   657d   661d
sysfilevols
  M68KSYS(ci)                             40d   583d
sysflag
  CRT(crt)                               238c   253c   687c   720c
  DGL_C_IN(dgl_confg_in)          ***  12050c
  DGL_TOOLS(dgl_tools)                 20028d 20032c 20034c 20061c
  INITLOAD(sysglobals)            ***    283d
  KERNEL(general_0)               ***   1015c
sysflag2
  INITLOAD(sysglobals)            ***    289d
sysflag_def
  C_HOOK                                  38d    52d
  DGL_TOOLS(dgl_tools)                 20016d 20028d
```

```
sysglobals
  A804XDVR(a804xdvr)              ***      31d
  AMIGO(amigodvr)                 ***     425d
  AMIGO(csamigo)                  ***      37d
  ASCII                           ***     317d
  ASCII(asciimodule)              ***       4d
  BAT(bat)                        ***      33d
  BUBBLES(bubble)                 ***      26d
  CLOCK(clock)                    ***      33d
  CONVERT(convert_text)           ***      27d
  CRT(crt)                        ***      33d
  CS80(cs80)                      ***      66d
  CS80(cs80dsr)                   ***     598d
  CS80(cs80dvr)                   ***     972d
  CS80(tapebuf)                   ***      35d
  CTABLE                          ***    1186d
  CTABLE(brstuff)                 ***     928d
  CTABLE(ctr)                     ***     239d
  CTABLE(options)                 ***      47d
  CTABLE(scanstuff)               ***    1035d
  C_HOOK                          ***      35d
  DC_DRV(extdc)                   ***     194d
  DC_DRV(init_dc)                 ***     594d
  DC_DRV(intdc)                   ***     232d
  DGL_C_IN(dgl_confg_in)          ***   12017d
  DGL_C_OUT(dgl_confg_out)        ***   11017d
  DGL_RAS(dgl_raster)             ***   17014d
  DISCHPIB(bkgnd)                 ***      31d
  DISCHPIB(dischpib)              ***     189d
  DI_DRV(extdi)                   ***     150d
  DI_DRV(init_discint)            ***     200d
  DMA_DRV(init_dma)               ***     164d
  EDRIVER(edriver)                ***      24d
  EPROMS(eproms)                  ***       6d
  ETU                             ***      26d
  F9885(f9885dvr)                 ***      37d
  GCRT(crtb)                      ***      33d
  GLE_KNOB(gle_knob_in)           ***   18028d
  GLE_RGL(gle_ras_out)            ***    8004d
  G_DRV(extg)                     ***     167d
  G_DRV(init_gpio)                ***     218d
  HEAPT(sysglobals)               ***      26d
  HPHIL(hphil)                    ***      30d
  H_DRV(exth)                     ***     166d
  H_DRV(init_hpib)                ***     216d
  INIT                            ***    2514d
  INIT(fs)                        ***     739d
  INIT(initunits)                 ***    2135d
  INIT(isr)                       ***      34d
  INIT(ldr)                       ***    2270d
  INIT(misc)                      ***     153d
  INITLOAD                        ***    1664d
  INITLOAD(bootdammodule)         ***     535d
  INITLOAD(loader)                ***     809d
  INITLOAD(mini)                  ***     304d
  INITLOAD(sysglobals)            ***       6d
  IOLIB(general_2)                ***     826d
  KERNEL(general_0)               ***     771d
  KERNEL(lodeclarations)          ***     236d
  KEYS(keys)                      ***      33d
  LIB(dgl_lib)                    ***   20096d
  LIBRARIAN                       ***      27d
  LIFDAM(lifmodule)               ***      26d
  LOCKMOD(lockmodule)             ***      27d
  M68KSYS(ci)                     ***      33d
```

```
  MINIT(asmr)                  ***   1319d
  MINIT(bminit)                ***     57d
  MINIT(cs80ir)                ***    516d
  MINIT(hminit)                ***    248d
  MINIT(iramigo)               ***    187d
  MINIT(midecs)                ***     13d
  MINIT(mminit)                ***    111d
  MINIT(qminit)                ***    878d
  MINIT(xminit)                ***   1358d
  MIUI                         ***     11d
  MOREFSYS(mfs)                ***     10d
  MOUSE(mouse)                 ***     29d
  PRINTER(prtdvr)              ***     35d
  RS_DRV(init_rs)              ***    212d
  RS_DRV(rs)                   ***    167d
  SEGMENTER(segmenter)         ***     18d
  SRMDAM(srmdammodule)         ***     32d
  SRMDRM(srmdammodule)         ***     33d
  SRM_DRV(srm)                 ***     30d
  SWVOL                        ***      4d
  SYSDEVS(sysdevs)             ***     33d
  TAIL                         ***     32d
  TAPEBKUP                     ***    953d
  TAPEBKUP(cs80tbdvr)          ***    305d
  TAPEBKUP(cs80tbr)            ***     14d
  UCSD_AM(ucsd_am)             ***     29d
  UCSD_DAM(ucsdmodule)         ***     29d
  UNITIO(uio)                  ***     25d
sysioresult
  INITLOAD(sysglobals)         ***    248d
syslibrary
  INIT                         ***   2577c
  INIT(ldr)                    ***   2444c
  INITLOAD(sysglobals)         ***    267d
  M68KSYS(ci)                         621c   693c   728c
sysmenu
  KEYS(keys)                          470c   584c
  SYSDEVS(sysdevs)             ***    160d
sysmenushift
  KEYS(keys)                          471c   585c
  SYSDEVS(sysdevs)             ***    161d
sysname
  INITLOAD                           1742c  1750c
  INITLOAD(sysglobals)         ***    281d
  TAIL                         ***     51c
sysnorm
  KEYS(keys)                           39d   584c
sysp
  INITLOAD                           1736d  1742c
sysprefix
  INITLOAD                           1668d  1764c
sysshft
  KEYS(keys)                           41d   585c
sysstart
  INITLOAD                           1669d  1740c  1745c  1751c  1753c  1765c  1771c  1790c
system_card
  KERNEL(general_0)            ***    989c
  KERNEL(iodeclarations)       ***    325d
system_cmap
  GLE_RGL(gle_ras_out)               8064d  8330c  8350c
system_cmap_def
  GLE_RGL(gle_ras_out)               8021d  8064d
```

```
system_controller
  IOLIB(hpib_1)                       400d   481d   487c   490c
  IOLIB(hpib_2)                      1816c  1871c  1957c  1964c
system_init
  DGL_VARS(dgl_vars            ***   1240d
  GEN(dgl_gen)                 ***   3078c
  LIB(dgl_lib)                      20582c 20612c 20641c 20672c 20731c 20946c 21108c 21378c 21381c
system_unit
  CTABLE(ctr)                         305d   859d   861c   863c
systemdate
  SRMDAM(srmdammodule)                357d   364c
systemid
  INITLOAD(loader)                    944d  1580c
  LIBRARIAN                           350c   351c  1963c
systemname
  TAIL                                 34d    51c    52c    53c    56c    58c    59c    61c    62c
systemstartup
  M68KSYS                      ***   1215c
  M68KSYS(ci)                          77d  1164d
systemtime
  SRMDAM(srmdammodule)                358d   370c
systime
  LIBRARIAN                    ***    234c
  LIFDAM(lifmodule)            ***    117c
  M68KSYS(ci)                         122c   158c   256c
  MIUI                         ***     62c
  SYSDEVS(sysdevs)                    385d   466d
systmtype
  SRMDAM(srmdammodule)                 69d  1090c  1387c
sysunit
  CTABLE                       ***   1846c
  CTABLE(ctr)                  ***    861c
  INIT(initunits)              ***   2260c
  INIT(ldr)                    ***   2316c
  INITLOAD                     ***   1673c
  INITLOAD(loader)                   1605c  1609c  1610c
  INITLOAD(sysglobals)         ***    265d
  M68KSYS(ci)                         160c   287c   678c   849c
  SWVOL                        ***     13c
sysunit_list
  CTABLE                       ***   1837c
  CTABLE(options)              ***    198d
sysunit_list_length
  CTABLE                       ***   1838c
  CTABLE(options)                     192d   196d
sysunit_list_type
  CTABLE(options)                     195d   199d
sysunit_ok
  CTABLE                             1829c  1831c  1837c  1840c
  CTABLE(ctr)                         305d   859d   864c   898c   909c
sysvol
  M68KSYS(ci)                         670d   674c   678c   685c   687c   688c
sysvolname
  M68KSYS(ci)                         583d   620c
syvid
  INIT(ldr)                          2316c  2317c  2318c  2445c
  INITLOAD(sysglobals)         ***    266d
  M68KSYS(ci)                         194c   611c   616c   678c   688c   706c   843c
  SWVOL                        ***     13c
```

```
t
  CLOCK(clock)                          74d    76c    79c    80c    81c   137d   142c
  CONVERT(convert_text)                 35d    36d    40d    65c    72c    76c    79c    87d   100c   104c   117c   119c
  CS80(cs80)                           345d   500d   513c
  CTABLE(ctr)                          345d   639c   645c   651c   658c   664c   673c   682c   694c   707c   716c   733c   752c   768c
                                       787c   809c   816c
  DGL_POLY(dgl_poly)                 20411d 20795c 20797c 20798c 20834c 20836c 21032d 21040c 21078d 21086c 21123d 21130c 21154d 21161c
  GLE_RGL(gle_ras_out)                8160d  8163d  8164d  8165d  8166d  8167d  8168d  8169d  8170d  8171d  8172d  8173d  8174d  8175d
                                      8176d  8177d  8178d
  GLE_UTLS(gle_utls)                 21044d 21055c 21061c 21066c
  HEAPT(hpm)                            87d    89c    90c    94d    96c
  INIT(fs)                             795d   796d   800d   801d   802d   803d   804d   805d   806d   807d   808d   809d   810d   811d
                                       812d   813d   814d   825d   827d
  M68KSYS(ci)                          398d   540c
  MOREFSYS(mfs)                         18d    19d   515d   520c   523c   526c   667d   675c
  SYSDEVS(sysdevs)                     474d   475c
t1
  CLOCK(clock)                          90d    94c    96c
  NONUSKBD1(non_us_kbd1)                76d    77d    82d    85d    88d    91d    94d
  NONUSKBD2                             33d    34d    61d    63d    66d    70d    74d    78d    82d    87d    93d    99d   103d   108d
                                       110d   114d   117d   119d
t2
  CLOCK(clock)                          90d    95c    96c
  NONUSKBD1(non_us_kbd1)                77d    80d
  NONUSKBD2                             34d    60d
t3
  NONUSKBD2                             35d    36d    40d    41d    42d    43d    44d    45d    46d    47d    48d    49d    50d    54d
                                        55d    56d    57d    58d
t4
  NONUSKBD2                             36d    39d
t_cnt
  IOLIB(general_4)                    1203d  1355d  1371c  1396c  1410c  1426c
t_count
  IOLIB(general_4)                    1229d  1258d  1267c  1271c  1442d  1444c  1445c  1466d  1468c  1469c  1491d  1493c  1494c  1534d
                                      1538c  1541c  1543c
t_dir
  DISCHPIB(dischpib)                   443d   453c   458c   474c
  IOLIB(general_4)                    1201d  1207d  1213d  1220d  1225d  1353d  1389c  1424c  1438d  1445c  1462d  1469c  1488d  1494c
                                      1496c  1531d  1536c  1543c
t_tfr
  IOLIB(general_4)                    1200d  1206d  1212d  1219d  1224d  1352d  1421c  1437d  1445c  1461d  1469c  1487d  1494c  1508c
                                      1509c  1510c  1511c  1530d  1543c
t_type
  CS80(cs80)                           229d   345d   500d   506d
ta
  CS80(cs80)                    ***    221d
tab
  INIT(misc)                    ***    164d
  KEYS(keys)                    ***    481c
table
  TAIL                                  34d    50c    62c    75c
tablenumber
  TAPEBKUP(cs80tbr)                    269d   277c
tablepc
  LIBRARIAN                            309d  1154c  1220c  1233c
tableptr
  INITLOAD(loader)                    1218d  1236c  1238c  1243c
tag_base
  DISCHPIB(dischpib)                   212d   278c   338c   402c
tagbase
  PRINTER(prtdvr)                       58d   220c
tail
  TAIL                          ***     30d
```

```
talk
  IOLIB(hpib_2)                       1797d  1981d
talk_constant
  GLE_HPIB(gle_hpib_io)              10213c 10301c
  IOLIB(hpib_1)                        535c   623c
  IOLIB(hpib_2)                       1827c  1905c  1984c
  KERNEL(iodeclarations)        ***    316d
talker
  IOLIB(hpib_3)                       2041d  2145d  2148c
tape_dt
  CTABLE                              1377d  1381c
tape_uep
  TAPEBKUP                            1218d  1386c  1387c  1393c  1397c  1405c
tapebkup
  TAPEBKUP                      ***      8d
tapebuf
  CS80                          ***   1411d
  CS80(cs80dsr)                 ***    598d
  CS80(cs80dvr)                 ***    972d
  CS80(tapebuf)                 ***     32d
  TAPEBKUP                      ***    953d
  TAPEBKUP(cs80tbdvr)           ***    305d
tapebuf_block
  CS80(cs80dvr)                       1157c  1172c  1177c  1185c  1193c
  CS80(tapebuf)                 ***     47d
tapebuf_maxsize
  CS80(cs80dvr)                 ***   1241c
  CS80(tapebuf)                         39d    41d
tapebuf_ptr
  CS80(cs80dvr)                       1157c  1193c  1268c  1283c  1295c  1318c
  CS80(tapebuf)                         44d    55c    56c
tapebuf_size
  CS80(cs80dvr)                       1157c  1192c  1193c
  CS80(tapebuf)                 ***     48d
tapebuf_state
  CS80(cs80dsr)                 ***    629c
  CS80(cs80dvr)                       1149c  1156c  1158c  1172c  1178c  1186c  1194c  1294c  1306c  1317c  1369c
  CS80(tapebuf)                         45d    57c
  TAPEBKUP                            1245c  1482c
  TAPEBKUP(cs80tbdvr)                  583c   815c
tapebuf_state_type
  CS80(tapebuf)                         42d    45d
tapebuf_type
  CS80(tapebuf)                         41d    44d
tapebuf_uep
  CS80(cs80dsr)                 ***    628c
  CS80(cs80dvr)                       1150c  1157c  1172c  1176c  1185c  1193c  1369c  1377c
  CS80(tapebuf)                 ***     46d
  TAPEBKUP(cs80tbdvr)                  582c   814c
tapesetup
  TAPEBKUP(cs80tbdvr)                  366d   383c
target_h
  DGL_RAS(dgl_raster)                17071d 17105c 17116c 17117c
target_l
  DGL_RAS(dgl_raster)                17073d 17105c 17121c 17125c
target_s
  DGL_RAS(dgl_raster)                17072d 17105c 17120c 17137c
tb
  GEN(dgl_gen)                        3193d  3229c  3233c  3240c
tbit
  CS80(cs80)                           506d   513c
tc
  DISCHPIB(dischpib)                   270d   277c   329d   336c
```

```
tcnt
  GLE_HPGL(gle_hpgl_out)          7166d  7181c  7183c  7184c  7185c  7186c  7187c  7188c  7222d  7241c  7243c  7244c  7245c  7246c
                                  7247c  7248c
  GLE_HPGLI(gle_hpgl_in)         18097d 18107c 18109c 18110c 18111c 18112c 18113c 18114c 18134d 18147c 18149c 18150c 18151c 18152c
                                 18153c 18154c 18291d 18300c 18302c 18303c 18304c 18305c
tct_message
  IOLIB(hpib_2)           ***    1910c
  KERNEL(iodeclarations)  ***     304d
td
  A804XDVR(a804xdvr)              256d   267c   269c   274c   276c   279c   280c   296c   298c   302c   307c   308c   309c   316c
                                  317c   319c   320c
  LIBRARIAN               ***    2196d
  SYSDEVS(sysdevs)                221d   477d   478c   480c
tdata
  A804XDVR(a804xdvr)              257d   269c   276c   279c   282c   283c   289c   293c   296c   300c   302c   305c   307c   308c
                                  309c
tdate
  INITLOAD(bootdammodule)         556d   563d
  LIFDAM(lifmodule)                41d    54d    56d    64d   113d   548d
tdp
  INITLOAD(loader)                900d  1528c
  LIBRARIAN                      1282c  1870c  1873c  1877c  1883c  1913c  1921c  1943c  1947c  2171c  2199c  2624c  2628c  2765c
tdx
  DGL_POLY(dgl_poly)            20446d 20452c 20457c 20467c 20468c
tdy
  DGL_POLY(dgl_poly)            20446d 20452c 20457c 20467c 20468c
te
  INITLOAD(mini)                  332d   338d   342d   343d   344d   345d   346d   347d   348d   349d   350d   351d   352d   353d
                                  354d   355d   356d   357d   358d   359d   360d   361d   362d   363d   364d   365d   366d   367d
                                  368d   369d   370d
tea
  CTABLE(ctr)                     420d   633c   639c   645c   651c   658c   664c   673c   682c   694c   707c   716c   733c   752c
                                  768c   787c   809c   816c
tea_amigo_mv
  CTABLE                         1554c  1693s  1700s  1711s  1716s
  CTABLE(ctr)                     334d   737d
tea_amigo_sv
  CTABLE                         1540c  1661s  1671s  1681s  1727s  1738s
  CTABLE(ctr)                     335d   720d
tea_boot
  CTABLE(ctr)                     324d   662d
tea_bubble
  CTABLE                  ***    1781s
  CTABLE(ctr)                     338d   805d
tea_crt
  CTABLE                  ***    1499c
  CTABLE(ctr)                     326d   643d
tea_cs80_mv
  CTABLE                         1574c  1754s
  CTABLE(ctr)                     336d   756d
tea_cs80_sv
  CTABLE                         1596c  1600c  1764s  1769s  1774s
  CTABLE(ctr)                     337d   772d   796c
tea_eprom
  CTABLE                  ***    1787s
  CTABLE(ctr)                     339d   812d
tea_flpy
  CTABLE                         1274c  1278c
  CTABLE(ctr)                     333d   791d
tea_hp9290x
  CTABLE                         1643s  1644s
  CTABLE(ctr)                     332d   711d   795c
tea_hp9885
  CTABLE(ctr)                     330d   686d   797c
```

```
tea_hp9895
  CTABLE                         1531c  1648s  1649s  1654s
  CTABLE(ctr)                     331d   698d   798c
tea_kbd
  CTABLE                  ***    1500c
  CTABLE(ctr)                     327d   649d
tea_local_printer
  CTABLE                  ***    1508c
  CTABLE(ctr)                     328d   677d
tea_memory_volume_dam
  CTABLE                  ***    1497c
  CTABLE(ctr)                     323d   637d
tea_mini
  CTABLE                         1638s  1639s
  CTABLE(ctr)                     329d   655d   794c
tea_nounit
  CTABLE(ctr)                     631d   826c   870c
tea_srm
  CTABLE                         1505c  1623c
  CTABLE(ctr)                     325d   668d
temp
  A804XDVR(a804xdvr)              379d   383c   396c   397c
  CRT(crt)                        554d   642c   645c
  DC_DRV(extdc)                   198d   199d   200d   202d   204d   205d   207d   209d   211d   212d
  DC_DRV(intdc)                   244d   245d   246d   247d   248d   249d   250d   252d   254d   263d   267c   269c   273c   305d
                                  307c   312d   317c   322d   325c   330d   335c   340d   343c   348d   351c   356d   374c   382c
                                  389c   390c   398c   408d   417c   445c   450c   464c   494c   551c   556c
  DGL_POLY(dgl_poly)            20256d 20259c 20260c 20261c 20262c 20263c
  DI_DRV(extdi)                   155d   156d   157d   158d   159d   160d   161d   163d   165d   166d   167d   168d   169d   170d
                                  172d
  ETU                              72d   730c   732c   733c   948c   952c   960c   961c   966c  1231c
  GEN(dgl_gen)                   3390d  3493c  3494c  3495c  3517c  3518c  3519c  3552c  3554c  3557c  3567c  3569c  3573c
  GLE_HPGL(gle_hpgl_out)         7167d  7192c  7194c  7198c  7200c
  GLE_HPGLI(gle_hpgl_in)        18098d 18118c 18120c 18124c 18126c
  GLE_RGL(gle_ras_out)           8321d  8359c  8361c  8362c  8364c
  G_DRV(extg)                     172d   173d   174d   175d   176d   177d   178d   180d   182d   183d   184d   185d
  HEAPT(hpm)                      216d   223c   224c   226c   233c   234c   236c   237c   248c   257c   259c
  HPHIL(hphil)                    227d   252c   253c   256c   257c   259c   260c   262c   264c   265c
  H_DRV(exth)                     170d   171d   172d   173d   174d   175d   176d   178d   180d   181d   182d   183d   184d   185d
                                  187d
  KERNEL(general_0)               803d   809d   811c   814d   816c   819d   821c   824d   826c   829d   831c   834d   836c   839d
                                  841c   844d   846c
  KERNEL(iocomasm)                701d   702d
  KERNEL(iodeclarations)          418d   419d   421d   423d   425d   427d   430d   433d   435d   438d   440d
  LIBRARIAN                      1161d  1233c  1234c
  LIFDAM(lifmodule)               738d   765c   767c   768c   769c   775c   787c   789c   790c   791c   797c   992d   997c   999c
                                 1006c  1007c
  RS_DRV(rs)                      172d   173d   174d   175d   176d   177d   178d   180d   182d
  UCSD_DAM(ucsdmodule)            187d   243c   251c   256c   257c   381d   389c   390c
temp1
  SYSDEVS(sysdevs)                514d   549c   552c   553c
temp2
  LIFDAM(lifmodule)               163d   186c   187c   188c   189c
temp_color_map
  GLE_RGL(gle_ras_out)           8319d  8336c  8346c
temp_color_model
  DGL_RAS(dgl_raster)           17582d 17619c 17645c
temp_control
  DGL_HPGL(dgl_hpgl)            17257d 17289c 17290c
  DGL_RAS(dgl_raster)           17580d 17608c 17609c
temp_dav
  CTABLE                         1341d  1344c  1345c  1347c  1348c  1349c
temp_file_pass
  SRMDAM(srmdammodule)             65d   160c  1067c  1094c  1142c  1158c  1163c  1301c  1397c  1400c
```

```
temp_msus
  CTABLE                          1356d  1358c  1359c  1365c
temp_point
  DGL_POLY(dgl_poly)             20556d 20571c 20572c 20575c 20577c 20579c 20584c
temp_unitable
  CTABLE(ctr)                      371d   458c   469c   825c   828c   836c   840c   852c   871c
temp_value
  GLE_UTLS(gle_utls)             21046d 21050c 21054c 21056c 21057c
tempaddr
  GCRT(crtb)                       421d   481c   483c
tempcapbits
  SRMDAM(srmdammodule)             470d   485c   486c   512d   542c   543c
tempcounter
  SRMDAM(srmdammodule)              76d  1043c  1044c
tempd
  LIFDAM(lifmodule)                362d   367c   368c   370c   372c   374c   375c
tempdirptr
  LIBRARIAN                       1859d  1894c  1958c  1965c  1990c
tempec
  INITLOAD(bootdammodule)          618d   622c   623c   624c   630d
tempentry
  LIFDAM(lifmodule)                685d   695c   696c   698c   708c   709c   712c   713c
tempf
  LIBRARIAN                       3305d  3318c  3321c  3384c  3385c  3388c  3390c  3398c  3447c  3460c  3485c  3494c
tempfibspace
  SRMDAM(srmdammodule)            1443d  1445c  1459c  1466c
tempindex
  LIFDAM(lifmodule)               1017d  1020c  1028c  1029c  1032c  1062d  1070c  1076c  1078c
tempioresult
  SRMDAM(srmdammodule)             137d   162c   164c   391d   416c   418c   425c  1335d  1403c  1406c
tempname
  ETU                              223d   234c   235c   247c   248c   255c   274c   276c
  LIFDAM(lifmodule)                633d   636c   648c
temporary
  LIFDAM(lifmodule)                630d   651c
tempptr
  MINIT(xminit)                   1411d  1702c  1703c  1706c  1745c  1754c  1756c  1767c  1776c  1785c  1787c  1792c  1806c  1817c
temprec
  SRMDAM(srmdammodule)             915d   942c   943c   944c   967d  1027d  1049c  1050c
temps
  BUBBLES(bubble)                  126d   129c   130c   131c
tempstr
  SRMDAM(srmdammodule)             138d   161c   163c   165c
tempstring
  LIBRARIAN                  ***    40d
tempstuff
  HPHIL(hphil)                     133d   140c   160d   167c
term
  IOLIB(general_2)                 805d   819d   965d   983c  1050d  1062c
  IOLIB(general_4)                1217d  1485d  1505c
term_char
  DC_DRV(intdc)                    460c   498c
  DISCHPIB(dischpib)         ***   475c
  IOLIB(general_4)                1425c  1505c
  KERNEL(iodeclarations)     ***   578d
  SRM_DRV(srm)               ***  1347c
term_count
  DC_DRV(intdc)                    445c   447c   448c   458c   465c   475c   477c   496c   497c   511c
  DISCHPIB(dischpib)               514c   515c   516c   530c
  IOLIB(general_4)           ***  1426c
  KERNEL(iodeclarations)     ***   579d
  SRM_DRV(srm)               ***  1348c
```

```
termhpgl
  DGL_C_IN(dgl_confg_in)         12067d 12106c
  DGL_C_OUT(dgl_confg_out)       11367d 11434c 11475c
termknob
  DGL_C_IN(dgl_confg_in)         12026d 12052c
termraster
  DGL_C_OUT(dgl_confg_out)       11047d 11278c
test_and_toggle
  AMIGO(amigodvr)                  703c   706c
  DISCHPIB(dischpib)               498c   501c
test_pass
  MINIT(hminit)                    346d   357c   359c
test_pattern
  MINIT(hminit)                    339d   359c
test_pattern_type
  MINIT(hminit)                    337d   339d
test_point
  DGL_POLY(dgl_poly)             20555d 20568c 20572c 20584c
testarea
  MINIT(cs80ir)                    786d   798c
testbyte
  MINIT(xminit)                   1462s  1470c  1473c  1476c  1481c
testioresult
  M68KSYS(ci)                      420d   473c   520c   529c
testpasses
  MINIT(hminit)                    334d   337d   357c
text_cos_dir
  GLE_SMARK(gle_smark)       ***  5127c
  GLE_STEXT(gle_stext)            4038c  4043c  4047c  4052c  4086c  4089c
  GLE_TYPES(gle_types)       ***  1126d
text_dir
  GLE_GEN(gle_gen)                2090c  2102c
  GLE_HPGL(gle_hpgl_out)     ***  7535c
  GLE_RGL(gle_ras_out)       ***  8403c
  GLE_SMARK(gle_smark)            5134c  5150c
  GLE_TYPES(gle_types)       ***  1058d
text_just
  GLE_HPGL(gle_hpgl_out)     ***  7536c
  GLE_RGL(gle_ras_out)       ***  8404c
  GLE_TYPES(gle_types)       ***  1059d
text_line_x
  GLE_STEXT(gle_stext)       ***  4051c
  GLE_TYPES(gle_types)       ***  1164d
text_line_y
  GLE_STEXT(gle_stext)       ***  4052c
  GLE_TYPES(gle_types)       ***  1165d
text_sin_dir
  GLE_SMARK(gle_smark)       ***  5128c
  GLE_STEXT(gle_stext)            4039c  4042c  4048c  4051c  4086c  4090c
  GLE_TYPES(gle_types)       ***  1125d
text_space_x
  GLE_STEXT(gle_stext)       ***  4047c
  GLE_TYPES(gle_types)       ***  1161d
text_space_y
  GLE_STEXT(gle_stext)       ***  4048c
  GLE_TYPES(gle_types)       ***  1162d
text_spacing
  GLE_GEN(gle_gen)           ***  2084c
  GLE_HPGL(gle_hpgl_out)     ***  7533c
  GLE_RGL(gle_ras_out)       ***  8402c
  GLE_TYPES(gle_types)       ***  1060d
textam
  UCSD_AM(ucsd_am)                 40d   180c   271c
```

```
textbuf
  LIBRARIAN                    1368d  1382c  1393c  1395c  1399c  1403c
textbufblocks
  LIBRARIAN                    2156d  2513c  2516c  2519c  2520c  2522c
textbuffer
  INITLOAD(loader)             1500d  1515c  1516c  1563c
  LIBRARIAN                    2134d  2177c  2179c  2204c  2206c  2207c  2232c  2237c  2238c  2250c  2251c  2455c  2522c  2548c
                               2599c
textbuftop
  LIBRARIAN                    2135d  2246c  2523c  2547c  2598c
textdescptr
  INITLOAD(loader)              873d   900d
textdescriptor
  INITLOAD(loader)              873d   988d  1530c
  LIBRARIAN                    1291c  1875c  1876c  1920c  1945c  1948c  2181c  2216c  2714c  2801c
textfield
  CRT(crt)                       54d   273c   285c   679c
  GCRT(crtb)              ***   101d
textfile
  ETU                     ***   930c
  INITLOAD(sysglobals)    ***    49d
  LIBRARIAN               ***  2930c
  M68KSYS(ci)             ***  1046c
  UCSD_AM(ucsd_am)              270c   271c   272c
  UCSD_DAM(ucsdmodule)    ***   294c
textinblock
  LIBRARIAN                    2157d  2258c  2259c  2633c
textindex
  LIBRARIAN                    2136d  2177c  2204c  2206c  2234c  2238c  2244c  2246c  2250c  2251c  2258c  2261c  2547c  2548c
                               2592c  2595c  2596c  2598c  2599c  2600c  2601c  2707c
textinsize
  LIBRARIAN                    2158d  2247c  2260c  2631c  2707c
textoutblock
  LIBRARIAN                    2159d  2173c  2207c  2232c  2233c
textoutsize
  LIBRARIAN                    2160d  2177c  2201c  2604c  2634c
textptr
  INITLOAD(sysglobals)           33d   257d
textrecctr
  LIBRARIAN                    1273d  1281c  1282c  1284c  1286c
textrecords
  INITLOAD(loader)              962d  1527c
  LIBRARIAN                     383c  1281c  1286c  1942c  1978c  2622c  2765c  2822c
textrecs
  LIBRARIAN                    1856d  1942c  1943c  1952c
textsize
  INITLOAD(loader)              990d  1532c  1539c
  LIBRARIAN                    1288c  1295c  1296c  1297c  1304c  1870c  1877c  1883c  1921c  1946c  1947c  2174c  2201c  2221c
                               2624c  2631c  2634c  2814c  2816c  2817c  2818c  2820c
textstart
  INITLOAD(loader)              989d  1539c
  LIBRARIAN                    1287c  1296c  2173c  2633c  2817c
textstep
  LIBRARIAN                     319d   326c   328c   329c   354c   355c   358c   360c  1272d  1277c  1278c  1279c  1280c  1282c
                               1291c  1293c  1294c
texttype
  SRMDAM(srmdammodule)    ***    72d
tfname
  INITLOAD(bootdammodule)       651d   662c   665c
tfr_data_sec
  AMIGO(amigodvr)               764d   768c
tfr_end
  SRM_DRV(srm)                 1300d  1306c  1349c
```

```
tg
  GEN(dgl_gen)                 3193d  3228c  3234c  3244c
thedate
  CLOCK(clock)                   49d    65c   100d   104c
  SYSDEVS(sysdevs)              384d   386d   463d   464c   469d   471c
thedatetime
  M68KSYS(ci)                   150d   153c   160c
thetime
  A804XDVR(a804xdvr)            239d   243c   245c   250c   251c
  CLOCK(clock)                   73d    77c    85d    93c    94c    95c   114d   118c   127d   139c   140c   150c   156c   160c
  SYSDEVS(sysdevs)              385d   387d   466d   467c   473d   475c
thirdparm
  LIBRARIAN                    3121d  3123c  3142c  3149c
this
  HEAPT(hpm)                    135d   149c   150c   155c   156c   159c
this_device
  AMIGO(csamigo)                205d   207c   208c   215c
this_entry_ptr
  MINIT(qminit)                1108d  1203c  1204c  1217c  1248c
thislink
  HEAPT(hpm)                    135d   150c   151c   152c   155c   156c   158c
ti
  ETU                           150d   161c   164c   165c
tid
  INIT(ldr)               ***  2352d
  INIT(misc)              ***   180d
  INITLOAD(loader)        ***  1018d
  INITLOAD(sysglobals)           42d   102d
  M68KSYS(ci)             ***   915d
  SRMDAM(srmdammodule)    ***  1631d
tidlenq
  INIT(ldr)               ***  2355c
  INIT(misc)              ***   437c
  INITLOAD(bootdammodule) ***   697c
  INITLOAD(sysglobals)           19d    42d
  LIFDAM(lifmodule)             222c  1071c
  M68KSYS(ci)             ***   920c
  SRMDAM(srmdammodule)    ***  1669c
time
  GLE_HPIB(gle_hpib_io)        10082d 10084c 10089c 10098c
  INITLOAD(sysglobals)    ***   231d
  IOLIB(general_1)              247d   331d   333c   338c   347c
  LIBRARIAN                     222d   234c   235c
  LIFDAM(lifmodule)             548d   555c   556c   557c
  M68KSYS(ci)                   109d   122c   124c   159c   166c   286c
  MIUI                           53d    62c    78c    80c    81c    83c    84c
  SRMDAM(srmdammodule)          360d   372c   373c   374c   375c  1465c
  SRM_DRV(srm)                 1114d  1115d  1116d  1235d  1237c  1241d  1243c  1247d  1249c  1254d  1258c  1260c
timed_cut
  GLE_HPIB(gle_hpib_io)        10061d 10251c 10339c
  IOLIB(hpib_1)                 430d   573c   661c
timeout
  DISCHPIB(dischpib)            277c   336c
  GLE_HPIB(gle_hpib_io)        10100c 10206c 10232c 10294c 10320c
  IOLIB(general_1)        ***   343c
  IOLIB(hpib_1)                 528c   554c   616c   642c   711c   761c
  KERNEL(general_0)            1209c  1267c
  KERNEL(iodeclarations)        501d   515d   529d
  PRINTER(prtdvr)               114c   219c   233c
  SRM_DRV(srm)            ***  1260c
timeout20
  HPHIL(hphil)                   78d    86c
timeout_blanked
  PRINTER(prtdvr)                74d   171c   186c   189c
```

```
timeout_counter
  PRINTER(prtdvr)                    77d   172c   189c   191c   192c   193c   194c
timeoutrec
  GLE_HPIB(gle_hpib_io)           10018d 10061d 10192d 10280d
  IOLIB(hpib_1)                     422d   430d   514d   602d
timeouts_per_beep
  PRINTER(prtdvr)                    54d   193c
timer
  A804XDVR(a804xdvr)                255d   262c   285c   290c   294c   317c   319c
  GLE_HPIB(gle_hpib_io)           10191d 10255c 10257c 10258c 10279d 10343c 10345c 10346c
  IOLIB(hpib_1)                     513d   577c   579c   580c   601d   665c   667c   668c   747d
  SYSDEVS(sysdevs)                  221d   477d
timerdata
  A804XDVR(a804xdvr)          ***   256d
  SYSDEVS(sysdevs)                  215d   221d   477d
timerec
  CLOCK(clock)                       73d   114d
  INIT(misc)                  ***   191d
  INITLOAD(sysglobals)              223d   231d
  LIBRARIAN                   ***   222d
  LIFDAM(lifmodule)                 115d   548d   585d
  M68KSYS(ci)                       109d   151d   177d
  MIUI                        ***    53d
  SRMDAM(srmdammodule)              358d   388d
  SYSDEVS(sysdevs)                  202d   218d   385d   387d   466d   473d   474d
timerexists
  GLE_HPIB(gle_hpib_io)           10059d 10247c 10335c
  IOLIB(hpib_1)                     428d   569c   657c
timeriohook
  A804XDVR(a804xdvr)          ***   394c
  SYSDEVS(sysdevs)                  223d   611c
timeriotype
  SYSDEVS(sysdevs)                  221d   223d
timerisrhook
  A804XDVR(a804xdvr)          ***   100c
  SYSDEVS(sysdevs)                  224d   612c
timermask
  A804XDVR(a804xdvr)          ***   385c
  SYSDEVS(sysdevs)            ***   166d
timerops
  A804XDVR(a804xdvr)                255d   394c
timeroptype
  A804XDVR(a804xdvr)          ***   255d
  SYSDEVS(sysdevs)                  214d   221d   477d
timertypes
  A804XDVR(a804xdvr)          ***   255d
  SYSDEVS(sysdevs)                  213d   221d   477d
times
  MINIT(xminit)                    1411d  1567c  1569c  1640c
timestring
  MIUI                               57d    76c    79c    80c    82c    83c    86c    87c    90c
timetype
  CLOCK(clock)                      167c   169c
  SYSDEVS(sysdevs)            ***   202d
tioresult
  M68KSYS(ci)                        56d  1191c  1198c
title
  INIT(fs)                          754d   764d   768d   769d
  M68KSYS(ci)                       988d   995c   996c   997c   999c  1000c  1001c  1017d  1025c  1026c
```

```
tm
  ASCII(asciimodule)                 37c    50c   137c
  CTABLE(ctr)                       453d   472c   842c
  ETU                         ***  1052c
  GLE_KNOB(gle_knob_in)           18054c 18067c
  INIT(initunits)             ***  2205c
  INIT(misc)                        269c   458c   469c   503c   505c   521c   548c   564c   578c
  INITLOAD                    ***  1678c
  INITLOAD(loader)            ***  1611c
  INITLOAD(sysglobals)        ***   145d
  LIFDAM(lifmodule)                 251c   319c   352c   444c   447c   469c   528c   530c  1144c
  M68KSYS(ci)                       351c   352c   383c   812c   963c
  MINIT(bminit)                      79c    86c
  MIUI                              493c   495c   497c
  SWVOL                       ***    24c
  UCSD_AM(ucsd_am)                   91c   165c   176c
  UCSD_DAM(ucsdmodule)              604c   606c
  UNITIO(uio)                        65c    67c    86c    97c   109c
tm_fib
  MIUI                              459d   485c   493c   495c   497c
tm_name
  CTABLE(ctr)                       423d   463c
tm_proc
  CTABLE(ctr)                       451d   463c   464c   466c   472c
tmp
  DGL_RAS(dgl_raster)             17384d 17389c 17390c
tmp2
  DGL_RAS(dgl_raster)             17385d 17390c 17391c
tmpcnt
  DC_DRV(intdc)                     412d   493c   494c   511c   512c
tmpptr
  DC_DRV(intdc)                     411d   417c   418c   427c   428c   531c   532c
to_memory
  DC_DRV(intdc)               ***   548c
  DISCHPIB(dischpib)                453c   517c   528c   538c
  IOLIB(general_4)                 1389c  1597c
  KERNEL(iodeclarations)      ***   564d
  SRM_DRV(srm)                     1325c  1435c  1440c  1445c  1450c  1460c
toc
  MIUI                              413d   447d
tod
  LIFDAM(lifmodule)                 115d   117c   118c
todaysdate
  LIBRARIAN                          38d   233c  1960c  2897c  3160c  3712c
  MIUI                               56d    67c    68c    69c    70c    71c    74c    90c
todindex
  LIFDAM(lifmodule)                 386d   415c   423c   424c   431c   432c   457c   463c   466c
toggle_c
  C_HOOK                            117d   136c
toggle_graphics
  DGL_C_OUT(dgl_confg_out)        11212d 11235c
togglea
  CRT(crt)                          279d   699c
togglealphahook
  CRT(crt)                    ***   699c
  DGL_RAS(dgl_raster)         ***  17494c
  GCRT(crtb)                  ***   518c
  KEYS(keys)                        383c   391c
  SYSDEVS(sysdevs)                  120d   622c
toggleg
  CRT(crt)                          204d   700c
```

```
togglegraphicshook
  CRT(crt)                          ***     700c
  C_HOOK                            ***     136c
  DGL_RAS(dgl_raster)               ***   17302c
  GCRT(crtb)                        ***     519c
  KEYS(keys)                              384c   390c
  SYSDEVS(sysdevs)                        121d   623c
toggleprinter
  LIBRARIAN                              2918d  3646c
token
  SRMDAM(srmdammodule)                    214d   231c   286c   287c   289c   291c   303c   308c   311c   318c
tokenlen
  SRMDAM(srmdammodule)                    205d   230c
tokentype
  SRMDAM(srmdammodule)                    201d   209d
took_type_ahead
  DGL_C_OUT(dgl_confg_out)               11037d 11066c 11188c 11274c
top1
  C_HOOK                                   44d   144c
  DGL_TOOLS(dgl_tools)                   20022d 20047c 20050c
top2
  C_HOOK                                   44d   144c
  DGL_TOOLS(dgl_tools)                   20022d 20047c 20050c
top_down_sort
  DGL_POLY(dgl_poly)                     20423d 20699c 20813c 20814c
topbyte
  CRT(crt)                                 52d   275c   287c   681c
  GCRT(crtb)                        ***     99d
topos
  LIFDAM(lifmodule)                       386d   415c   425c   435c   438c   447c   448c   453c   474c
total
  LIBRARIAN                         ***  3178d
total_words
  F9885(f9885dvr)                         105d   143c   158c   159c   212c
totaldefs
  INIT(ldr)                         ***  2478c
  INITLOAD(loader)                        1077d  1471c  1490c  1641c
  SEGMENTER(asm)                           241c   244c   306c   323c
totalglobal
  INIT(ldr)                               2468c  2472c
  INITLOAD(loader)                        1062d  1373c  1379c  1631c  1635c
  LIBRARIAN                               1708c  1970c  3003c
  SEGMENTER(asm)                           206c   208c
totalpatchspace
  LIBRARIAN                                 78d  1667c  1676c  2182c  2514c
totalreloc
  INIT(ldr)                         ***  2476c
  INITLOAD(loader)                        1061d  1373c  1378c  1515c  1535c  1563c  1639c
  LIBRARIAN                               1707c  1968c  3003c
  SEGMENTER(asm)                           235c   238c   304c   322c
tp
  DGL_POLY(dgl_poly)                     20442d 20472c 20473c 20474c
  LIBRARIAN                               1487d  1517c  1518c  1519c  1528c  1531c  1532c
tp1x
  GLE_HPGL(gle_hpgl_out)                  7223d  7307c  7314c
  GLE_HPGLI(gle_hpgl_in)                 18135d 18202c 18208c
tp1y
  GLE_HPGL(gle_hpgl_out)                  7223d  7308c  7316c
  GLE_HPGLI(gle_hpgl_in)                 18135d 18203c 18210c
tp2x
  GLE_HPGL(gle_hpgl_out)                  7223d  7307c  7318c
  GLE_HPGLI(gle_hpgl_in)                 18135d 18202c 18212c

tp2y
  GLE_HPGL(gle_hpgl_out)                  7223d  7308c  7320c
  GLE_HPGLI(gle_hpgl_in)                 18135d 18203c 18214c
tpm
  CS80(cs80dvr)                            978d  1073c  1074c
  CTABLE                                  1383c  1559c  1569c  1697s  1746s
  CTABLE(ctr)                              287d   527d   528d   529d   530d   531d   532d   533d   534d   535d   536d   537d   538d   540d
                                           586c   587c   600c   608c   610c   617c   623c
tposit
  ASCII(asciimodule)                       101d   112c   119c   120c
tps
  LIFDAM(lifmodule)                         51d   292c   513c   521c
tptr
  F9885(f9885dvr)                           81d   154c   184c   264c
tr
  GEN(dgl_gen)                            3193d  3227c  3239c  3245c
  LIFDAM(lifmodule)                        585d   588c
track
  AMIGO(csamigo)                           342d   348c   351c   352c   356c   357c   366d   372c   373c   374c
  F9885(f9885dvr)                           60d   116d   149c
track1
  MINIT(asmr)                             1338d  1349d
track2
  MINIT(asmr)                             1336d  1347d
track_ert
  MINIT(cs80ir)                     ***    551d
  MINIT(qminit)                     ***   1247c
track_ert_passes
  MINIT(qminit)                           1114d  1247c
tracka
  MINIT(xminit)                           1404d  1627c  1630c  1706c  1707c  1710c  1724c  1734c  1759c  1792c  1795c  1800c
trackb
  MINIT(xminit)                           1404d  1707c  1723c  1758c  1793c  1796c  1799c
tracknum
  MINIT(asmr)                             1333d  1344d
tracknumber
  MINIT(xminit)                           1405d  1571c  1581c  1583c  1587c  1617c  1624c  1627c  1629c  1643c  1651c  1652c  1656c  1662c
                                          1669c  1673c  1675c  1687c
tracks
  CTABLE(ctr)                              606d   608c   610c   611c
  INIT(misc)                               262d   290c   291c
tracks_per_medium
  MINIT(hminit)                            304d   366c   384c   403c   439c   450c   487c
transblocks
  LIBRARIAN                               2832d  2865c  2866c  2867c  2868c  2869c  2870c  2871c
transfer
  CS80(cs80dvr)                           1214d  1395c
  F9885(f9885dvr)                          105d   286c
  INITLOAD(mini)                           406d   514c
  IOLIB(general_4)                        1205d  1436d
transfer_cmd_buf
  AMIGO(amigodvr)                          715d   753c
transfer_command
  AMIGO(amigodvr)                          714d   725c   726c   732c   733c   753c
transfer_end
  IOLIB(general_4)                        1222d  1529d
transfer_setup
  IOLIB(general_4)                        1198d  1350d  1430c  1445c  1469c  1494c  1543c
transfer_successful
  AMIGO(amigodvr)                          786d   833c   854c   861c   864c   907c   919c   944c   948c   956c
  INITLOAD(mini)                           409d   450c   469c   475c
transfer_until
  IOLIB(general_4)                        1216d  1484d
```

```
transfer_word
  IOLIB(general_4)                1210d  1459d
transfercomplete
  F9885(f9885dvr)                   61d   208c
transition
  LIBRARIAN                        754d   790c   793c   799c   802c
translatedate
  SRMDAM(srmdammodule)             356d   435c   436c   608c   609c  1465c
transmode
  A804XDVR(a804xdvr)               330c   335c   343c   368c   400c
  KEYS(keys)              ***      264c
  NONUSKBD1(non_us_kbd1)  ***      127c
  NONUSKBD2               ***      136c
  SYSDEVS(sysdevs)                 288d   651c
transparent_sec
  CS80(cs80)                       275d   395c
trick
  M68KSYS(ci)                      786d   808c   813c
trick_proc
  SEGMENTER(asm)                    52d   114d   258d
trickint
  CLOCK(clock)                      38d    90d   137d
trickrec
  SRMDAM(srmdammodule)             909d   915d   959d   967d  1018d  1027d
tries
  F9885(f9885dvr)                  112d   142c   210c   232c   233c
trigger
  HPHIL(hphil)                      41d    88c
  IOLIB(hpib_2)                   1799d  1989d
trim
  LIBRARIAN                       2902d  3134c  3235c  3251c  3274c  3369c
trk_per_cyl
  AMIGO(csamigo)                   130d   193d   194d   196d   197d   198d   199d   200d   201d   202d   203d   225c   356c   357c
                                   373c
  MINIT(hminit)                    291c   487c
try_tracks
  INIT(misc)                       265d   273c   276c   289c   291c   292c
tryit
  HEAPT(hpm)                       213d   259c   264c   266c
tryrec
  HEAPT(sysglobals)       ***       31d
trysrmboot
  INITLOAD                        1757d  1786c
tshift
  KEYS(keys)                       189d   216c   217c
tsize
  UCSD_DAM(ucsdmodule)             570d   603c   604c   606c   608c   609c   610c
ttime
  CLOCK(clock)                      91d    93c
tttt
  AMIGO(amigodvr)                  482c   484c   532c   534c   670c   898c
  AMIGO(csamigo)                   106d   317c
ttttttt
  MINIT(iramigo)                   227d   235c
tva
  AMIGO(csamigo)                   142d   363d   369c   384d   387c   397d   400c   408d   414c   415c
  MINIT(cs80ir)                    832d   838c   839c   840c
tva_type
  AMIGO(csamigo)                    88d   122d   142d   339d   363d   384d   397d   408d
  MINIT(cs80ir)                    520d   832d
twobytes
  EDRIVER(edriver)                  85d    90d   102d
  ETU                             1157d  1169d
```

```
twosets
  MOUSE(mouse)            ***       42d
  SYSDEVS(sysdevs)        ***      314d
tx
  GEN(dgl_gen)                    3280d  3292c  3293c
  LIB(dgl_lib)                   20267d 20277c 20281c
txmax
  LIB(dgl_lib)                   20495d 20512c 20519c 20522c 20888d 20908c 20915c 20920c
txmin
  LIB(dgl_lib)                   20494d 20511c 20519c 20522c 20887d 20907c 20914c 20920c
ty
  GEN(dgl_gen)                    3280d  3294c  3295c
  LIB(dgl_lib)                   20267d 20278c 20282c
tymax
  LIB(dgl_lib)                   20497d 20515c 20520c 20522c 20890d 20910c 20917c 20920c
tymin
  LIB(dgl_lib)                   20496d 20514c 20520c 20522c 20889d 20909c 20916c 20920c
typ
  INIT(fs)                          764d   765d
type_card_id
  KERNEL(iodeclarations)            382d   543d
type_device
  GLE_HPIB(gle_hpib_io)          10188d 10276d
  IOLIB(general_2)                  797d   799d   802d   806d   808d   811d   814d   817d   820d   822d   832d   913d   936d   966d
                                    992d  1012d  1025d  1038d  1051d  1071d
  IOLIB(general_4)                 1199d  1205d  1211d  1218d  1223d  1351d  1436d  1460d  1486d  1529d
  IOLIB(hpib_1)                     403d   406d   408d   411d   510d   598d   696d   744d
  IOLIB(hpib_2)                    1780d  1783d  1787d  1789d  1792d  1793d  1799d  1838d  1861d  1897d  1916d  1935d  1951d  1989d
  IOLIB(hpib_3)                    2033d  2094d
  KERNEL(iodeclarations)  ***       369d
type_hpib_addr
  GLE_HPIB(gle_hpib_io)   ***    10120d
  IOLIB(hpib_1)                     396d   444d
  IOLIB(hpib_2)                    1782d  1796d  1798d  1854d  1974d  1982d
  KERNEL(iodeclarations)  ***       387d
type_hpib_line
  IOLIB(hpib_0)                    1710d  1712d  1714d  1723d  1734d  1745d
  KERNEL(iodeclarations)  ***       388d
type_isc
  DC_DRV(init_dc)         ***       602d
  DC_DRV(intdc)           ***       410d
  DI_DRV(init_discint)    ***       208d
  GLE_HPIB(gle_hpib_io)          10064d 10072d 10081d 10108d 10119d 10143d 10189d 10190d 10277d 10278d 10511d 10532d
  G_DRV(init_gpio)        ***       226d
  H_DRV(init_hpib)        ***       224d
  IOLIB(general_1)                  237d   238d   240d   242d   244d   246d   274d   282d   293d   304d   317d   330d
  IOLIB(general_2)                  837d   917d   939d   969d   995d  1014d  1027d  1040d  1053d  1073d
  IOLIB(general_4)                 1204d  1249d  1356d  1357d  1441d  1465d  1490d  1533d  1667d
  IOLIB(hpib_0)                    1709d  1711d  1713d  1722d  1733d  1744d
  IOLIB(hpib_1)                     393d   395d   398d   401d   404d   407d   409d   412d   413d   432d   443d   467d   482d   496d
                                    511d   512d   599d   600d   697d   698d   745d   746d
  IOLIB(hpib_2)                    1779d  1781d  1785d  1795d  1797d  1800d  1801d  1811d  1839d  1853d  1862d  1889d  1898d  1918d
                                   1936d  1952d  1973d  1981d  1995d  2001d
  IOLIB(hpib_3)                    2029d  2031d  2037d  2039d  2041d  2043d  2046d  2063d  2080d  2096d  2117d  2135d  2145d  2154d
                                   2164d
  IOLIB(serial_0)                  2215d  2217d  2219d  2231d  2294d  2357d
  IOLIB(serial_3)                  2503d  2506d  2509d  2512d  2515d  2518d  2529d  2598d  2682d  2736d  2801d  2836d
  KERNEL(general_0)                 747d   750d   753d   756d   759d   762d   852d   871d   890d   909d   929d   944d   965d  1253d
  KERNEL(iodeclarations)            368d   616d
  PRINTER(prtdvr)         ***        66d
  RS_DRV(init_rs)         ***       219d
  SRM_DRV(srm)                    1253d  1297d
type_of_card
  KERNEL(iodeclarations)            381d   541d
```

```
type_of_position
  SRM_DRV                          ***   1956c
  SRM_DRV(srm)                     ***    680d
type_parity
  IOLIB(serial_3)                        2513d  2737d
  KERNEL(iodeclarations)           ***    401d
type_serial_line
  IOLIB(serial_0)                        2216d  2218d  2220d  2232d  2295d  2358d
  KERNEL(iodeclarations)           ***    407d
typeoftoken
  SRMDAM(srmdammodule)                    209d   253c   258c   268c   276c   284c
typepos
  SRM_DRV                                1943d  1956c
  SRM_DRV(srm)                     ***   1087d
typestring
  INIT(fs)                                768d   769d
  LIBRARIAN                              3120d  3142c  3149c
u
  GLE_KNOB(gle_knob_in)                 18045d 18048c 18050c 18059d 18061c 18064c
  MIUI                                     15d   173c   332c   349c   474c
  SRMMM(srmmmodule)                       146d   166c   167c   168c   169c
  UNITIO(uio)                              28d    30d    32d    33d    34d    53d    57c    59c    74d    78c    80c    92d    94c    96c
                                          101d   103c   106c   117d   121c
ub
  INITLOAD(loader)                        843d   895d
  LIBRARIAN                              2683c  2684c  2685c
ubyte
  INITLOAD(loader)                        857d   933d
  LIBRARIAN                        ***   2680c
ubyterec
  INITLOAD(loader)                        843d   895d
  LIBRARIAN                        ***   2682c
uc
  CS80(cs80)                              421d   427c   428c   429c
  M68KSYS(ci)                             407d   433c   450c   452c   455c   456c   497c   498c   499c
ucase
  INIT(ldr)                        ***   2362c
  INITLOAD(loader)                       1019d  1438c
  M68KSYS(ci)                      ***    926c
uclr_timeout_const
  PRINTER(prtdvr)                          51d   114c
ucsd
  CTABLE(options)                  ***     60d
ucsd_am
  UCSD_AM                          ***    27?d
  UCSD_AM(ucsd_am)                 ***     27d
ucsd_am_init
  UCSD_AM                          ***     25d
ucsd_dam
  CTABLE(ctr)                             295d   519c
  UCSD_DAM(ucsdmodule)                     32d    66d
ucsd_dam_name
  CTABLE(ctr)                             352d   512c   515c   520c
ucsd_to_any
  CONVERT(convert_text)                    36d    87d
ucsdmodule
  UCSD_DAM                         ***    684d
  UCSD_DAM(ucsdmodule)             ***     27d
ue
  CTABLE(scanstuff)                      1091d  1105c  1114c  1115c  1116c  1117c  1118c  1119c  1120c  1127c  1129c
uee
  CS80(cs80)                       ***    219d
  TAPEBKUP(cs80tbdvr)                     871c   874c
```

```
uee_byte
  TAPEBKUP(cs80tbdvr)                     706d   717c
uee_pending
  TAPEBKUP(cs80tbdvr)                     703d   871c   872c   873c   875c
ueovbytes
  INIT(misc)                              212d   257d   281c   282c   295c   296c   462c
  LIFDAM(lifmodule)                       207c   507c
  MIUI                             ***    521c
  UCSD_DAM(ucsdmodule)             ***    490c
```

```
uep
  AMIGO(amigocvr)          428d  439d  470c  472c  473c  474c  475c  476c  477c  478c  483c  490d  511c  512c
                           513c  515c  516c  519c  521c  524c  525c  527c  528c  529c  530c  533c  535c  543d
                           544d  545d  546d  547d  548d  557d  563c  566c  577c  578c  580c  581c  582c  583c
                           587c  594c  596c  598c  604c  605c  606c  607c  609c  611c  613c  614c  615c  618c
                           624c  627c  633c  634c  645c  646c  648c  654d  659c  661c  666c  667c  668c  671c
                           675c  678c  680c  681c  682c  685c  686c  689c  690c  691c  693c  700d  702c  705c
                           710d  720c  734c  737c  752c  753c  755c  756c  758c  762d  766c  768c  770c  774d
                           777c  778c  780c  784d  790d  799c  801c  802c  819c  822c  824c  825c  828c  837c
                           839c  873c  874c  880c  882c  888c  889c  891c  892c  893c  897c  898c  899c  902c
                           903c  906c  910c  914c  917c  921c  922c  923c  926c  931c  938c  941c  943c  946c
                           956c  958c  959c  960c  961c  964c  966c  969c
  AMIGO(csamigo)           137d  138d  139d  140d  141d  142d  144d  145d  146d  147d  148d  149d  150d  151d
                           152d  160d  162c  172c  177d  179c  183d  185c  189d  207c  209c  212c  219d  223c
                           229d  235c  240d  268c  269c  274d  291c  292c  296d  300c  304d  310c  311c  312c
                           314c  315c  328d  334c  335c  339d  344c  349c  351c  363d  368c  371c  372c  379d
                           387c  388c  392d  400c  401c  405d  412c  413c  414c  415c
  CS80(cs80)               339d  340d  341d  342d  343d  344d  345d  346d  347d  355d  356d  365d  376c  381d
                           393c  395c  396c  397c  401d  412c  416d  429c  433d  439c  440c  441c  445d  451c
                           452c  453c  457d  464c  465c  466c  468c  469c  473d  479c  480c  481c  485d  492c
                           493c  494c  495c  496c  500d  516c  517c  518c  522d  532c  533c  534c  538d  550c
                           551c  552c  556d  579c  580c  589c
  CS80(cs80dsr)            602d  604d  606d  611d  620c  621c  622c  635d  650c  651c  701c  702c  756c  757c
                           761c  763c  766c  771d  870c  882c  883c  888c  889c  897c  898c  903c  904c  909c
                           910c  923c  944c  945c  950c  951c
  CS80(cs80dvr)            982d  999d 1002c 1003c 1004c 1005c 1006c 1007c 1009c 1027d 1043c 1047c 1048c 1053c
                          1054c 1088d 1090c 1094d 1100c 1101c 1103c 1104c 1110c 1114c 1116c 1124c 1126c 1127c
                          1129c 1131c 1133c 1134c 1168d 1172c 1176c 1192c 1214d 1228c 1230c 1231c 1236c 1250c
                          1267c 1278c 1282c 1293c 1307c 1316c 1342c 1353d 1357c 1359c 1366c 1367c 1368c 1369c
                          1373c 1377c 1382c 1387c 1388c 1390c 1395c
  CTABLE(ctr)              877d  879c  881c  882c  883c
  CTABLE(scanstuff)       1051d 1052d 1053d
  DISCHPIB(bkgnd)           62d   63d   64d   65d   66d   67d  105d  115c  125d  129c  130c  133c  138c  146d
                           149c  150c  155d  157c  158c  163d  165c  170c  175d  177c
  DISCHPIB(dischpib)       192d  193d  194d  195d  196d  197d  198d  199d  200d  201d  202d  219d  221c  222c
                           223c  225c  230d  232c  237d  239c  244d  246c  249c  256d  261c  264c  266c  287d
                           292c  293c  307c  310d  315c  316c  325c  349d  351c  353c  357d  363c  364c  371c
                           379c  383d  394c  395c  400c  405c  410d  412c  416d  426c  429c  431c  432c  436d
                           437d  438d  440d  446c  447c  452c  457c  482c  487c  491c  495d  497c  500c  504d
                           508c  522d  526c  535c  542c  545c
  F9885(f9885dvr)           79d  126c  131c  136c  137c  148c  164c  201c  254c  255c  260c  279c  286c
  INITLOAD(mini)           393d  466c  467c  481c  482c  489c  492c  507c  514c
  MINIT(cs80ir)            559d  560d  561d  562d  563d  564d  565d  567d  569d  583d  585c  586c  590d  612c
                           616c  617c  619c  620c  621c  625d  643c  644c  645c  649d  666c  670c  671c  672c
                           676d  692c  695c  696c  697c  701d  718c  722c  723c  724c  728d  750c  754c  755c
                           757c  759c  765c  766c  770d  790c  800c  803c  804c  806c  813c  814c  818d  835c
                           841c  842c  843c  847d  863c  866c  867c  868c
  MINIT(hminit)            252d  253d  254d  270d  278c  283d  288c  299d  308c  316c  325c  327c  359c  360c
                           361c  364c  365c  366c  367c  368c  373c  379c  391c  392c  393c  394c  395c  398c
                           399c  400c  401c  402c  403c  404c  405c  424c  425c  426c  437c  438c  439c  440c
                           441c  446c  447c  448c  449c  460c  467c  470c  474c  475c  480c  481c  482c  485c
                           489c  492c  495c  499c  500c  501c
  MINIT(iramigo)           190d  191d  192d  196d  205c  208d  215c  216c  218c  221d  239c
  MINIT(qminit)            881d  882d  883d  885d  906d  919d  932d  955c  957c  958c  962c  963c  968c  969c
                           978c  979c  984c  991c  994c  996c 1017c 1098d 1120c 1124c 1129c 1130c 1133c 1137c
                          1150c 1151c 1154c 1158c 1159c 1167c 1168c 1173c 1174c 1179c 1180c 1187c 1188c 1197c
                          1198c 1209c 1210c 1216c 1218c 1228c 1229c 1234c 1235c 1240c 1241c 1247c 1249c 1254c
                          1267c 1268c 1271c 1275c 1276c 1277c
  MIUI                      19d   35c   36c   37c  167c  168c  192c  331c  337c  346c  349c  352c  474c  476c
                           492c  493c  495c  497c  499c  509c  514c
  TAPEBKUP                1021d 1023c 1031d 1033c 1061d 1085c 1086c 1091c 1182d 1192c 1427d 1434c 1435c 1449c
                          1452c 1461d 1468c 1469c 1483c 1486c
  TAPEBKUP(cs80tbdvr)      309d  313d  315d  320d  322d  323d  324d  325d  329d  339d  345d  349c  350c  351c
                           352c  353c  356c  357c  358c  364d  366d  372c  373c  377c  378c  383c  387d  389c
                           393d  395d  401c  402c  407c  411d  413d  419c  420c  425c  429d  431d  437c  438c
-------------------------------------------------------------------------------------------------------------
                           442c  443c  448c  452d  472c  493c  509d  530c  531c  581c  582c  584c  646c  647c
                           651c  653c  656c  661d  670c  671c  679c  684c  689c  894d  911d  913d  919c  920c
                           925c  929d  931d  937c  938c  943c
  TAPEBKUP(cs80tbr)         24d   25d   26d   27d   28d   29d   30d   48d   50c   51c   55d   67c   68c   69c
                            73d   90c   94c   95c   96c  100d  120c  127c  128c  129c  133d  148c  150c  151c
                           152c  156d  171c  173c  174c  175c  179d  198c  203c  204c  205c  255d  274c  278c
                           279c  281c  284c  291c  292c  294c
uep1
  TAPEBKUP                1005d 1007c 1009c 1010c 1011c 1015d 1017c
uep2
  TAPEBKUP                1005d 1008c 1009c 1010c 1011c 1015d 1017c
uep_proc
  TAPEBKUP                1181d 1192c
  TAPEBKUP(cs80tbdvr)      345d  352c
uep_proc_type
  CTABLE(scanstuff)       1051d 1062d
  TAPEBKUP             ***1181d
  TAPEBKUP(cs80tbdvr)      309d  345d
uep_type
  AMIGO(amigodvr)          428d  439d  490d  543d  545d  557d  654d  702c  710d  766c  777c  790d  938c
  AMIGO(csamigo)           137d  138d  139d  140d  141d  142d  144d  145d  146d  147d  148d  149d  150d  151d
                           152d  160d  177d  183d  189d  219d  229d  240d  274d  296d  304d  328d  339d  363d
                           379d  392d  405d
  CS80(cs80)               339d  340d  341d  342d  343d  344d  345d  346d  347d  355d  356d  365d  381d  401d
                           416d  433d  445d  457d  473d  485d  500d  522d  538d  556d
  CS80(cs80dsr)            602d  604d  606d  611d  614d  635d  771d
  CS80(cs80dvr)            982d  999d 1027d 1090c 1094d 1168d 1214d 1353d
  CS80(tapebuf)         ***  46d
  CTABLE(scanstuff)       1049d 1051d 1052d 1053d
  DISCHPIB(bkgnd)           35d   62d   63d   64d   65d   66d   67d  105d  125d  146d  155d  163d  175d
  DISCHPIB(dischpib)       192d  193d  194d  195d  196d  197d  198d  199d  200d  201d  202d  219d  230d  237d
                           244d  256d  287d  310d  349d  357d  383d  410d  416d  436d  437d  440d  495d  504d
                           526c
  MINIT(cs80ir)            559d  560d  561d  562d  563d  564d  565d  567d  569d  583d  590d  625d  649d  676d
                           701d  728d  770d  818d  847d
  MINIT(hminit)            252d  253d  254d  270d  283d  299d
  MINIT(iramigo)           190d  191d  192d  196d  208d  221d
  MINIT(qminit)            881d  882d  883d  885d  906d  919d  932d 1098d
  MIUI                  ***  19d
  TAPEBKUP                1005d 1015d 1021d 1031d 1061d 1182d 1218d 1427d 1461d
  TAPEBKUP(cs80tbdvr)      309d  313d  315d  320d  322d  323d  324d  325d  329d  332d  335d  339d  345d  364d
                           366d  387d  393d  395d  411d  413d  429d  431d  452d  509d  661d  693d  894d  911d
                           913d  929d  931d
  TAPEBKUP(cs80tbr)         24d   25d   26d   27d   28d   29d   30d   31d   33d   48d   55d   73d  100d  133d
                           156d  179d  209d  210d  255d
uinfo
  EDRIVER(edriver)         122d  124c
uio
  UNITIO(uio)           ***  23d
uisblkd
  CTABLE(ctr)              490c  491c  843c  864c
  INIT(initunits)         2222c 2223c
  INIT(misc)               281c  455c  463c
  INITLOAD              ***1696c
  INITLOAD(sysglobals)  ***  163d
  LIFDAM(lifmodule)     ***  249c
  M68KSYS(ci)              616c  964c
uisfixed
  CS80(cs80dsr)         ***  936c
  CTABLE(ctr)           ***  487c
  INIT(initunits)       ***2219c
  INITLOAD              ***1693c
  INITLOAD(sysglobals)  ***  160d
```

```
uisinteractive
  CTABLE(ctr)                  ***    484c
  INIT(initunits)              ***   2216c
  INITLOAD                     ***   1690c
  INITLOAD(sysglobals)         ***    157d
  M68KSYS(ci)                  ***    350c
uk_kbd
  A804XDVR(a804xdvr)           ***    190d
  NONUSKBD2                    ***    198c
  SYSDEVS(sysdevs)             ***    148d
ul
  ETU                                 535d   545c   549c
umaxbytes
  BUBBLES(bubble)                     111c   112c
  CS80(cs80dsr)                ***    939c
  CTABLE                       ***   1830c
  CTABLE(ctr)                         492c   900c   902c
  EPROMS(eproms)               ***    107c
  INIT(initunits)              ***   2224c
  INIT(misc)                          282c   295c   296c   454c
  INITLOAD                     ***   1697c
  INITLOAD(sysglobals)         ***    164d
  M68KSYS(ci)                         816c   824c
  MINIT(bminit)                        76c    84c   100c
umedia_valid
  AMIGO(amigodvr)                     577c   606c   666c   888c
  CS80(cs80dsr)                ***    626c
  CS80(cs80dvr)                ***   1390c
  CTABLE(ctr)                  ***    485c
  ETU                          ***    392c
  F9885(f9885dvr)                     136c   279c
  INIT(initunits)              ***   2217c
  INIT(misc)                   ***    465c
  INITLOAD                     ***   1691c
  INITLOAD(mini)                      466c   507c
  INITLOAD(sysglobals)         ***    158d
  LIFDAM(lifmodule)                   215c   259c  1118c  1120c  1160c
  M68KSYS(ci)                         865c   961c
  MIUI                         ***    266c
  TAPEBKUP                           1025c  1035c  1153c
  TAPEBKUP(cs80tbdvr)                 581c   813c
  UCSD_DAM(ucsdmodule)                 88c   633c   651c   677c
un
  CTABLE(brstuff)                     957d  1016c
  CTABLE(ctr)                         324d   325d   326d   327d   328d   329d   330d   331d   332d   333d   334d   335d   336d   337d
                                      338d   339d   421d   469c   500c   631d   633c   643d   645c   649d   651c   655d   658c   662d
                                      664c   668d   673c   677d   682c   686d   694c   698d   707c   711d   716c   720d   733c   737d
                                      752c   756d   768c   772d   787c   791d   794c   795c   796c   797c   798c   805d   809c   812d
                                      816c
  INIT(initunits)                    2181d  2203c
  MINIT(bminit)                        59d    60d    62d    72c    75c    95d   100c
un4
  CTABLE(brstuff)                     963d  1011c
unaddressing_char
  DISCHPIB(dischpib)                  524d   539c   540c   541c
unassemble
  LIBRARIAN                           294d  3651c
unassigned6
  CTABLE(scanstuff)            ***   1088d
unassigned7
  CTABLE(scanstuff)            ***   1088d
unblocked_dam_name
  CTABLE(ctr)                         353d   645c   651c   682c

unblockeddam
  INIT(initunits)                    2164d  2244c  2246c  2250c
  INIT(misc)                          213d   426d
  LIFDAM(lifmodule)            ***   1159c
  UCSD_DAM(ucsdmodule)         ***    650c
unbuf_read_cmd
  AMIGO(amigodvr)              ***    732c
  AMIGO(csamigo)               ***     53d
unbuf_write_cmd
  AMIGO(amigodvr)              ***    733c
  AMIGO(csamigo)               ***     55d
unbuffedam
  INIT(misc)                          483d   706c
unclipped_draw
  GLE_HPGL(gle_hpgl_out)       ***   7526c
  GLE_TYPES(gle_types)         ***   1181d
unclipped_move
  GLE_HPGL(gle_hpgl_out)       ***   7525c
  GLE_TYPES(gle_types)         ***   1180d
uncorrectable_data_error
  AMIGO(amigodvr)                     807c   872c
  AMIGO(csamigo)               ***     73d
  MINIT(hminit)                ***    319c
undefined
  CS80(cs80dsr)                ***    629c
  CS80(cs80dvr)                       1156c  1172c  1178c  1186c  1306c  1369c
  CS80(tapebuf)                         42d    57c
  TAPEBKUP                            1245c  1482c
  TAPEBKUP(cs80tbdvr)                  583c   815c
under_lock_do
  TAPEBKUP                            1181d  1239c  1247c  1248c  1266c  1269c  1286c  1369c  1372c  1417c  1449c  1452c  1483c  1486c
uninitialized_media
  CS80(cs80)                   ***    166d
  CS80(cs80dsr)                ***    681c
  TAPEBKUP(cs80tbdvr)                  561c   793c
unit
  AMIGO(csamigo)                        64d   102d   268c
  CS80(cs80)                           340d   343d   416d   427c   433d   439c   473d   479c
  ETU                                  222d   235c   355d
  INIT(misc)                           212d   257d   267c   268c   269c   280c
  MINIT(cs80ir)                        560d   625d   639c
  TAPEBKUP(cs80tbdvr)                  713d   719c   721c   724c   730c   740c   742c   745c
  TAPEBKUP(cs80tbr)                     24d    55d    65c
unit_available
  AMIGO(csamigo)               ***     70d
unit_busy
  AMIGO(amigodvr)              ***    587c
  CS80(cs80dvr)                ***   1373c
  DISCHPIB(bkgnd)                       66d   163d   166c   169c   177c
unit_fault
  CS80(cs80)                   ***    154d
  CS80(cs80dsr)                ***    669c
  TAPEBKUP(cs80tbdvr)                  549c   781c
unit_missing
  CTABLE                              1342d  1347c  1349c  1350c
unit_num
  SRM_DRV(srm)                         262d  1504c
unit_number
  SWVOL                                  8d    13c    14c    18c    19c
unit_prefix_successful
  CTABLE                              1229d  1235c  1801c  1806c  1807c
unit_unavailable
  AMIGO(amigodvr)              ***    879c
  AMIGO(csamigo)               ***     80c
```

```
unit_wait
  AMIGO(amigodvr)                    578c   594c   633c
  DISCHPIB(bkgnd)                     67d   175d
unitable
  AMIGO(amigodvr)            ***     563c
  ASCII(asciimodule)                  35c    46c    81c   137c
  BUBBLES(bubble)            ***      61c
  CS80(cs80dsr)              ***     618c
  CS80(cs80dvr)              ***    1357c
  CTABLE                            1806c  1820c  1830c  1847c
  CTABLE(ctr)                        837c   840c   863c   871c   879c   880c   897c   900c   902c
  EPROMS(eproms)                      46c    59c    81c    82c   104c
  ETU                                341c   366c   392c   405c   434c   904c   926c  1052c
  F9885(f9885dvr)            ***     254c
  GLE_KNOB(gle_knob_in)             18041c 18048c 18061c
  INIT(initunits)                   2203c  2238c
  INIT(misc)                         267c   269c   280c   431c   487c   519c   548c   564c   578c   617c
  INITLOAD                          1675c  1676c
  INITLOAD(loader)                  1390c  1438c  1605c
  INITLOAD(mini)             ***     481c
  INITLOAD(sysglobals)       ***     251d
  LIFDAM(lifmodule)                  215c   216c   246c   319c   350c   440c   469c   526c  1118c  1120c  1121c  1136c  1142c  1160c
  LOCKMOD(lockmodule)                 49c    72c    90c
  M68KSYS(ci)                        160c   287c   344c   613c   734c   809c   810c   865c   958c
  MINIT(bminit)                       72c   100c
  MIUI                                34c   167c   235c   513c
  PRINTER(prtdvr)            ***     271c
  SRMAM(srmammodule)                 149c   167c   168c   169c   188c
  SRMDAM(srmdammodule)               163c   337c   393c   451c   744c   867c   880c  1183c  1222c  1643c
  SRM_DRV(srm)                      1230c  1375c  1379c  1411c  1502c
  SWVOL                      ***      22c
  TAPEBKUP                          1007c  1008c  1085c  1122c  1295c  1308c  1434c  1468c
  UCSD_AM(ucsd_am)                    57c    91c   165c   176c
  UCSD_DAM(ucsdmodule)                74c    89c   132c   139c   160c   329c   463c   576c   631c
  UNITIO(uio)                         48c    57c    78c    94c   103c
unitableptr
  CTABLE(ctr)                ***     371d
  INITLOAD(sysglobals)                172d   251d
unitabletype
  INITLOAD(sysglobals)                167d   172d
unitbusy
  UNITIO(uio)                          34d   101d   105c   110c   121c
unitclear
  UNITIO(uio)                          32d    92d
unitentry
  CTABLE(ctr)                ***     877d
  CTABLE(scanstuff)                  1049d  1091d
  DISCHPIB(bkgnd)            ***      36d
  F9885(f9885cvr)            ***      79d
  INITLOAD                   ***    1675c
  INITLOAD(miri)             ***     393d
  INITLOAD(sysglobals)                142d   167d
unitid
  MINIT(xminit)                      1401d  1512c  1546c  1561c  1562c  1569c  1581c  1617c  1618c  1624c  1643c  1661c  1680c  1681c
                                     1698c  1699c  1710c  1720c  1744c  1751c  1756c  1781c  1782c  1795c  1796c  1826c  1827c
unitioinit
  INIT                       ***    2560c
  INIT(initunits)                    2143d  2254d
  INITLOAD                           1671d  1780c
```

```
unitnum
  CS80(cs80dsr)              ***     613d
  CTABLE                            1213d  1231d  1234c  1268d
  CTABLE(ctr)                        299d   303d   305d   306d   307d   308d   324d   325d   326d   327d   328d   329d   330d   331d
                                     332d   333d   334d   335d   336d   337d   338d   339d   421d   631d   643d   649d   655d   662d
                                     668d   677d   686d   698d   711d   720d   737d   756d   772d   791d   805d   812d   823d   832d
                                     850d   859d   868d   875d   887d   890d
  CTABLE(options)            ***     196d
  GLE_KNOB(gle_knob_in)      ***   18033d
  INIT(fs)                   ***     759d
  INIT(initunits)                   2141d  2160d  2165d  2181d  2232d  2236d  2255d
  INIT(misc)                         212d   213d   257d   426d
  INITLOAD(bootcammodule)            543d   607d
  INITLOAD(sysglobals)                40d   117d   140d   167d
  LIFDAM(lifmodule)                   28d   195d
  M68KSYS(ci)                        588d   953d
  MIUI                                32d   222d   374d
  SRMAM(srmammodule)         ***     146d
  SRMDAM(srmdammodule)                44d   331d   347d   383d   447d   457d   500d   563d   646d   876d   900d   957d   987d  1014d
                                     1171d  1211d  1246d  1264d  1311d  1330d  1439d  1473d  1497d  1625d
  SRM_DRV                            1780d  1825d  1845d  1864d  1881d  1902d  1941d  1965d  1995d  2022d  2045d  2064d  2081d
  SRM_DRV(srm)                       1000d  1001d  1002d  1004d  1005d  1013d  1021d  1029d  1033d  1035d  1041d  1055d  1067d  1070d
                                     1072d  1074d  1077d  1085d  1089d  1095d  1101d  1105d  1107d  1108d  1228d  1356d  1371d  1399d
                                     1492d  1529d  1547d  1585d  1623d  1658d  1681d  1702d  1728d
  TAPEBKUP                          1003d  1048d  1106d  1109d  1217d  1426d  1460d
  UCSD_DAM(ucsdmodule)                32d    66d
  UNITIO(uio)                ***      40d
unitnumber
  ETU                                173d   178c   183c   362c   364c   896c   898c
unitread
  UNITIO(uio)                         30d    74d
unitstatus
  AMIGO(amigodvr)            ***     586c
  BUBBLES(bubble)            ***     114c
  CRT(crt)                   ***     415c
  CS80(cs80dvr)              ***    1372c
  F9885(f9885dvr)            ***     271c
  GCRT(crtb)                 ***     285c
  GLE_KNOB(gle_knob_in)      ***   18067c
  INITLOAD(mini)             ***     499c
  INITLOAD(sysglobals)       ***      67d
  KEYS(keys)                 ***      83c
  M68KSYS(ci)                         119c   382c
  UNITIO(uio)                ***     109c
unitwait
  UNITIO(uio)                          33d   117d
unitwrite
  UNITIO(uio)                          28d    53d
unknown
  GLE_HPGL(gle_hpgl_out)             7021d  7103c  7112c  7125c  7177c  7408c  7437c  7475c  7494c
unl
  TAPEBKUP(cs80tbr)                   165d   171c   172c   173c
unl_message
  IOLIB(hpib_2)                      1903c  1997c
  KERNEL(iodeclarations)     ***     310d
unlisten
  IOLIB(hpib_2)                      1800d  1995d
unload
  TAPEBKUP(cs80tbdvr)        ***     919c
  TAPEBKUP(cs80tbr)                   29d   156d   175c
unload_all
  SEGMENTER(asm)             ***     170d
  SEGMENTER(segmenter)       ***      30d
```

```
unload_segment
  SEGMENTER(asm)                          155d   172c   216c   252c
  SEGMENTER(segmenter)          ***        29d
unloadcmd
  TAPEBKUP(cs80tbr                        168d   172c
unlock
  LOCKMOD(lockmodule)                      31d    62d
unlockfile
  INITLOAD(sysglobals)          ***       138d
  LOCKMOD(lockmodule)           ***        72c
  SRMRAM(srmdammodule)                   1663c  1781c
unlockpack
  SRMRAM(srmdammodule)          ***      1510c
  SRM_DRV                       ***      2045d
  SRM_DRV(srm)                  ***      1105d
unmasked
  CS80(cs80dsr)                           793d   798d   799d   800d   801d   802d   803d   804d   805d   806d   807d   808d   809d   810d
                                          811d   812d   813d   815d   816d   817d   818d   819d   820d   821d   822d   823d   824d   825d
                                          826d   827d   828d   829d   830d   832d   833d   834d   835d   836d   837d   838d   839d   840d
                                          841d   842d   843d   844d   845d   846d   847d   849d   850d   851d
unmodified
  CS80(cs80dvr)                          1158c  1194c
  CS80(tapebuf)                 ***        42d
unop
  LIBRARIAN                               736d   813c
unoptype
  LIBRARIAN                               732d   736d
unrecognized
  KEYS(keys)                              402d   444c   476c   491c   510c   512c   530c   545c
unrecoverable_data
  CS80(cs80)                              174d   305d
  CS80(cs80dsr)                 ***       695c
  TAPEBKUP(cs80tbdvr)                     575c   635c   807c
unrecoverable_data_overflow
  CS80(cs80)                    ***       173d
  CS80(cs80dsr)                 ***       694c
  TAPEBKUP(cs80tbdvr)                     574c   806c
unrestits
  INITLOAD(loader)              ***      1023d
  LIBRARIAN                               194c   398c   399c   400c  1573c  1688c  1689c  1690c  1790c
unsgn24
  CS80(cs80)                               74d   100d   112d
  CS80(cs80dsr)                 ***       780d
  CS80(cs80dvr)                 ***      1033d
  MINIT(cs80ir)                           522d   564d   818d
  MINIT(qminit)                 ***       940d
  TAPEBKUP(cs80tbdvr)           ***       459d
unsgn4
  CS80(cs80)                               76d   122d   123d   340d   343d   416d   433d   473d
  CS80(cs80dsr)                 ***       781d
  CS80(cs80dvr)                 ***      1034d
  MINIT(cs80ir)                           560d   625d
  MINIT(qminit)                           941d  1102d
  TAPEBKUP(cs80tbdvr)                     460d   713d
  TAPEBKUP(cs80tbr                         24d    55d
```

```
unsgn8
  CS80(cs80)                               75d   102d   103d   108d   109d   110d   113d   116d   220d   339d   340d   341d   342d   343d
                                          344d   345d   346d   347d   356d   365d   373d   381d   390d   433d   445d   457d   473d   485d
                                          500d   522d   527d   538d
  CS80(cs80dsr)                 ***       785d
  CS80(cs80dvr)                 ***      1039d
  MINIT(cs80ir)                           523d   533d   534d   536d   537d   538d   539d   544d   545d   559d   560d   561d   562d   563d
                                          564d   565d   566d   567d   568d   569d   590d   604d   605d   609d   625d   636d   649d   662d
                                          663d   676d   689d   701d   714d   715d   728d   729d   737d   746d   747d   770d   771d   782d
                                          783d   785d   786d   787d   818d   847d   860d
  MINIT(qminit)                           892d   895d   896d   897d   946d  1034d  1105d  1107d
  TAPEBKUP(cs80tbdvr)                     328d   465d
  TAPEBKUP(cs80tbr                         24d    25d    26d    27d    28d    29d    30d    34d    55d    73d    86d    87d   100d   114d
                                          133d   156d   179d   210d   255d   268d   269d   272d
unsigned16
  AMIGO(csamigo)                ***       134d
  MINIT(iramigo)                           190d   196d   201d
unt_message
  IOLIB(hpib_2)                 ***      2003c
  IOLIB(hpib_3)                 ***      2103c
  KERNEL(iodeclarations)        ***       311d
untalk
  IOLIB(hpib_2)                           1801d  2001d
untilmodname
  LIBRARIAN                               3299d  3325c  3340c  3489c  3494c
untilmodnum
  LIBRARIAN                               3300d  3314c  3326c  3424c  3426c  3433c  3437c  3451c  3454c  3459c  3468c  3490c  3494c
untnumb
  M68KSYS(ci)                              783d   797c   809c   810c   820c   825c   833c
untypedfile
  ETU                           ***       903c
  INIT(misc)                               471c   703c   710c
  INITLOAD                      ***      1707c
  INITLOAD(bootdammodule)                  669c   708c   713c
  INITLOAD(loader)              ***       831d
  INITLOAD(sysglobals)          ***        46d
  LIBRARIAN                     ***      3156c
  LIFDRM(lifmodule)                        145c   542c   920c
  M68KSYS(ci)                   ***       775d
  SRMDRM(srmdammodule)                      91c   948c
  UCSD_DAM(ucsdmodule)                      95c   164c   484c
```

```
unum
  GLE_KNOB(gle_knob_in)          18033d 18035c 18039c 18041c
  INIT(initunits)                 2141d  2160d  2165d  2232d
  INIT(misc)                       213d   426d   431c   462c
  INITLOAD(bootdammodule)          543d   607d
  INITLOAD(sysglobals)      ***    140d
  LIFDAM(lifmodule)                 28d   195d   207c   215c   216c   246c   319c   350c   507c   526c  1118c  1120c  1121c  1122c
                                  1136c  1142c  1159c  1160c
  SRMDAM(srmdammodule)              44d   331d   337c   340c   347d   352c   383d   393c   396c   421c   447d   451c   457d   492c
                                   500d   529c   563d   588c   646d   744c   807c   821c   839c   844c   850c   867c   876d   880c
                                   882c   893c   895c   900d   922c   927c   957d   980c   987d  1007c  1014d  1081c  1086c  1095c
                                  1102c  1112c  1123c  1129c  1134c  1143c  1146c  1154c  1159c  1164c  1171d  1183c  1185c  1202c
                                  1205c  1211d  1222c  1224c  1228c  1235c  1239c  1240c  1246d  1257c  1264d  1284c  1285c  1302c
                                  1305c  1311d  1330d  1352c  1355c  1366c  1367c  1372c  1380c  1381c  1382c  1384c  1389c  1391c
                                  1394c  1404c  1420c  1424c  1426c  1427c  1428c  1430c  1433c  1439d  1447c  1459c  1462c  1466c
                                  1473d  1477c  1483c  1497d  1508c  1510c  1625d  1643c  1690c  1692c  1697c  1700c  1703c  1706c
                                  1707c  1712c  1715c  1716c  1723c  1726c  1728c  1738c  1741c  1744c  1750c  1753c  1754c  1760c
                                  1762c  1764c  1766c  1770c  1773c  1774c  1779c  1781c  1783c  1785c  1787c  1789c  1791c
  SRM_DRV                         1775c  1776c  1780d  1803c  1820c  1821c  1825d  1840c  1841c  1845d  1859c  1860c  1864d  1877c
                                  1881d  1897c  1898c  1902d  1920c  1936c  1937c  1941d  1960c  1961c  1965d  1981c  1990c  1991c
                                  1995d  2018c  2022d  2040c  2041c  2045d  2059c  2060c  2064d  2073c  2076c  2077c  2081d  2107c
  SRM_DRV(srm)                    1000d  1001d  1002d  1004d  1005d  1013d  1021d  1029d  1033d  1035d  1041d  1055d  1067d  1070d
                                  1072d  1074d  1077d  1085d  1089d  1095d  1101d  1105d  1107d  1108d  1228d  1230c  1356d  1367c
                                  1371d  1375c  1379c  1395c  1399d  1411c  1473c  1492d  1502c  1529d  1539c  1541c  1547d  1569c
                                  1580c  1581c  1585d  1607c  1618c  1619c  1623d  1642c  1653c  1654c  1658d  1670c  1676c  1677c
                                  1681d  1697c  1698c  1702d  1723c  1724c  1728d  1754c
  UCSD_DAM(ucsdmodule)              32d   86d    74c    80c    89c   132c   139c   463c   576c   585c   631c   650c
  UNITIO(uio)                       40d   42c    46c    48c
up
  A804XDVR(a804xdvr)                49d   343c   369c
  CRT(crt)                          97d   143d
  CTABLE(scanstuff)               1062d  1118c  1119c  1127c  1129c
  GCRT(crtb)                ***     76d
  MOUSE(mouse)                      55d   115c   116c
  SYSDEVS(sysdevs)          ***     93d
upc
  INIT(ldr)                 ***   2357c
  INIT(misc)                       211d   246d
  LIBRARIAN                       2726c  3039c  3067c  3239c  3254c  3258c  3278c  3282c  3369c
  M68KSYS(ci)               ***    921c
  SRMDAM(srmdammodule)              286c   308c
  UCSD_DAM(ucsdmodule)              131c   177c   443c   452c   466c   487c   514c
upcchar
  ETU                               187d   202c   299c
  TAPEBKUP                          966d   980c  1505c
upchar
  CRT(crt)                  ***     465c
  GCRT(crtb)                ***     336c
  INIT(misc)                ***     172d
upcname
  INIT(ldr)                       2352d  2357c  2362c
  M68KSYS(ci)                      915d   920c   921c   926c
update
  M68KSYS(ci)                      658d   685c   693c   695c   708c
updatecursorhook
  CRT(crt)                  ***     701c
  GCRT(crtb)                ***     513c
  SYSDEVS(sysdevs)                 124d   626c
updatesysunit
  M68KSYS(ci)                      647d   685c   851c
upon_data_transfer_comp
  AMIGO(amigodvr)                  547d   768c   774d
upon_transfer_complete
  DISCHPIB(dischpib)               438d   481c   522d
```

```
upper_bound
  CTABLE(ctr)                      374d   376c
upper_byte
  CTABLE(scanstuff)               1095d  1122c
upper_char
  DISCHPIB(dischpib)               389d   398c
uppercase
  M68KSYS(ci)                       71d   247c   299d   302c   303c   450c   497c   576c
ureportchange
  AMIGO(amigodvr)                  606c   667c   902c
  CS80(cs80dsr)                    702c   874c   876c   958c
  CS80(cs80dvr)                   1043c  1153c  1155c  1161c  1390c
  CTABLE(ctr)               ***    488c
  F9885(f9885dvr)                  137c   279c
  INIT(initunits)           ***   2220c
  INIT(misc)                       285c   294c   453c   465c
  INITLOAD                  ***   1694c
  INITLOAD(mini)                   467c   507c
  INITLOAD(sysglobals)      ***    161d
  LIFDAM(lifmodule)                216c   258c  1121c
  MIUI                             492c   499c   520c   522c
  TAPEBKUP                        1026c  1036c
  TAPEBKUP(cs80tbdvr)              584c   816c
  UCSD_DAM(ucsdmodule)              86c    88c
  UNITIO(uio)                       64c    68c    85c    87c   112c
urr
  CS80(cs80)                ***    223d
  CS80(cs80dsr)                    755c   757c
  TAPEBKUP(cs80tbdvr)              645c   647c   878c   880c
us
  INIT(misc)                ***    172d
  KEYS(keys)                       241d   289c   360c   483c
us_kbd
  A804XDVR(a804xdvr)               192d   193d
  KEYS(keys)                ***    259c
  NONUSKBD1(non_us_kbd1)    ***    120c
  NONUSKBD2                 ***    131c
  SYSDEVS(sysdevs)          ***    150d
use_current_ptr
  SRM_DRV                   ***   2036c
  SRM_DRV(srm)              ***    754d
use_simulation
  DGL_POLY(dgl_poly)              21036d 21045c 21054c 21059c 21060c 21082d 21090c 21099c 21104c 21105c
usecurptr
  SRM_DRV                         2024d  2036c
  SRM_DRV(srm)              ***   1103d
usefeft
  SRMDAM(srmdammodule)            1031d  1045c  1087c  1124c  1135c  1147c  1339d  1418c  1421c
user_confirmation
  MIUI                             275d   548c
  TAPEBKUP                        1041d  1337c  1438c  1474c
user_isr
  KERNEL(general_0)               1213c  1214c  1268c  1269c
  KERNEL(iodeclarations)           494d   508d   522d
user_parm
  KERNEL(general_0)               1215c  1270c
  KERNEL(iodeclarations)           495d   509d   523d
user_sequencing_field
  SRM_DRV(srm)                     324d   332d  1456c  1458c  1459c  1486c
user_spec_timeout
  PRINTER(prtdvr)                   75d   241c   277c   282c
user_tfr_type
  IOLIB(general_4)                1200d  1206d  1212d  1219d  1224d  1362d  1437d  1461d  1487d  1530d
  KERNEL(iodeclarations)           549d   574d
```

```
user_time
  GLE_HPIB(gle_hpib_io)          10098c 10100c 10206c 10232c 10239c 10249c 10255c 10294c 10320c 10327c 10337c 10343c 10472c
  IOLIB(general_1)                 347c   349c
  IOLIB(hpib_1)                    528c   554c   561c   571c   577c   616c   642c   649c   659c   665c   711c   761c
  KERNEL(general_0)                985c  1261c
  KERNEL(iodeclarations)     ***   542d
  SRM_DRV(srm)               ***  1258c
userheap
  INIT(ldr)                       2323d  2331c  2341c
  M68KSYS(ci)                      873d   886c   889c
userinfo
  M68KSYS(ci)                       59d   628c   944c  1007c  1018c  1116c  1117c
userprogram
  INIT(ldr)                  ***  2337c
  INITLOAD                   ***  1720c
  M68KSYS(ci)                ***   887c
userstack
  INIT(ldr)                       2337c  2430c  2440c  2460c  2468c  2470c  2471c
  INITLOAD                   ***  1720c
  INITLOAD(loader)                1068d  1127c  1133c  1598c  1625c  1631c  1633c  1634c  1657c
  M68KSYS(ci)                      807c   887c   947c
useunit
  ETU                               60d   242c   243c   247c   362c   363c   896c   897c
usf
  SRM_DRV(srm)                    1480d  1486c
usi
  MIUI                             288d   330c   331c   333c   334c   337c
usr_tfr
  DC_DRV(intdc)                    421c   422c   437c
  DISCHPIB(dischpib)               470c   471c   510c
  IOLIB(general_4)                1421c  1508c  1509c  1510c
  KERNEL(iodeclarations)     ***   574d
  SRM_DRV(srm)               ***  1344c
utb
  CS80(cs80)                        88d   583c
  TAPEBKUP(cs80tbr)                125c   242c   247c
uuppercase
  CTABLE(ctr)                ***   486c
  INIT(initunits)            ***  2218c
  INITLOAD                   ***  1692c
  INITLOAD(loader)           ***  1438c
  INITLOAD(sysglobals)       ***   159d
  UCSD_RAM(ucsdmodule)       ***   110c
uuuu
  CS80(cs80)                 ***   123d
  CS80(cs80dsr)              ***   763c
  TAPEBKUP(cs80tbdvr)              653c   719c   886c
  TAPEBKUP(cs80tbr)                240c   245c
uvalc
  CS80(cs80)                       566d   579c   580c   581c   582c   583c   584c   585c   586c   587c   588c   589c
uvid
  CTABLE                     ***  1847c
  CTABLE(ctr)                       328d   479c   677d   682c   864c
  INIT(initunits)            ***  2212c
  INIT(misc)                        446c   447c
  INITLOAD                   ***  1685c
  INITLOAD(sysglobals)       ***   152d
  LIFDA1(lifmodule)                 269c   270c   272c  1136c
  M68KSYS(ci)                       615c   616c   814c
  SRMAM(srmammodule)                163c   169c
  SRMDA1(srmdammodule)       ***   895c
  UCSD_RAM(ucsdmodule)              107c   110c   115c   132c   465c   466c   467c   661c

uw
  INITLOAD(loader)                  896d  1235c
  LIBRARIAN                        2492c  2566c  2577c  2690c  2691c  2692c
uword
  INITLOAD(loader)                  857d   933d
  LIBRARIAN                  ***  2687c
uwordext
  LIBRARIAN                         447d   946c
v
  IOLIB(general_4)           ***  1260d
  KERNEL(iocomasm)           ***   703d
  KERNEL(iodeclarations)            420d   422d   424d   426d   429d   432d   437d   439d   441d
  M68KSYS(ci)                       397d   540c
  MIUI                               15d   174c   332c   349c   474c
v1
  ASCII(asciimodule)                141d   143c   145c
v2
  ASCII(asciimodule)                141d   143c   149c
v3
  ASCII(asciimodule)                141d   145c   149c
val
  DC_DRV(extdc)                     208d   210d
  DC_DRV(intdc)                     247d   249d   253d   322d   325c   340d   343c   357d   374c   382c
  DI_DRV(extdi)                     158d   160d   164d   166d
  G_DRV(extg)                       175d   177d   181d
  H_DRV(exth)                       173d   175d   179d   181d
  LIBRARIAN                         143d   175c   213c   215c   216c   458d   461c   614d   616c  1317d  1333c  1334c
  RS_DRV(rs)                        175d   177d   181d
valptr
  LIBRARIAN                        2145d  2181c  2182c  2184c  2186c  2187c  2310d  2318c  2319c  2320c  2372c  2393c  2404c  2412c
                                   2453c  2556c  2557c  2558c  2585c  2586c  2647c  2652c  2657c  2736d  2759c  2760c  2806c  2807c
value
  CTABLE(ctr)                       316d   382d   388c   402c   448d   454d   460c   463c   466c
  CTABLE(scanstuff)                1061d  1103c  1137c  1155c  1156c  1157c  1158c  1159c  1161c  1162c  1163c  1165c  1166c  1167c
                                   1168c  1169c
  GLE_FILE(gle_file_io)            9019d  9020d  9037d  9043d
  GLE_HPGL(gle_hpgl_out)           7081d  7088c
  GLE_HPGLI(gle_hpgl_in)          18054d 18061c
  GLE_HPIB(gle_hpib_io)           10024d 10025d 10073d 10078c 10467d 10472c 10476d 10480c
  GLE_TYPES(gle_types)             1064d  1066d  1214d  1216d
  GLE_UTILS(gle_utils)            21009d 21037d 21050c 21058c 21074d 21092c 21097c 21102c 21103c
  INITLOAD(loader)                  933d  1465c
  IOLIB(general_1)                  239d   241d   245d   283d   288c   294d   299c   318d   321c
  IOLIB(general_2)                  800d   815d   914d   925c  1026d  1030c
  IOLIB(general_4)                 1236d  1238d  1558d  1576c  1585d  1603c
  KERNEL(general_0)                 752d   758d   764d   781d   785c   786c   789c   796d   798c   873d   884c   911d   923c   932d
                                    938c   939c   946d   949c
  LIBRARIAN                         175c   283d   288c  1179c  1492c  1493c  1495c  1496c  1547c  1548c  1600c  1641c  1821c  1827c
                                   1930c  2187c  2373c  2453c  2558c  2586c  2663c  2664c  2665c  2670c  2671c  2672c  2677c  2678c
                                   2683c  2684c  2685c  2690c  2691c  2692c  2698c  2760c  2807c
  M68KSYS(ci)                       745d   752c   757c   762c
  SEGMENTER(asm)             ***   132c
  SRMDAM(srmdammodule)              400c  1479c
  SRM_DRV(srm)               ***   209d
  SYSDEVS(sysdevs)                  40d   371d   405d   428d   430c
value1
  SYSDEVS(sysdevs)                  39d   400d
value2
  SYSDEVS(sysdevs)                  39d   400d
valueextend
  INITLOAD(loader)                  924d  1461c
  LIBRARIAN                         173c  1545c  1601c  1636c  1646c  1819c  1825c  1927c  1931c  2184c  2554c  2583c  2757c  2783c
                                   2804c
```

```
vname
  ETU                              307d   318c
  LIFDAM(lifmodule)                 35d    44d
vnh
  SRM_DRV(srm)                    1491d  1494c
vnlength
  INITLOAD(loader)                 814d   819d
  LIBRARIAN                  ***  3158c
  UCSD_DAM(ucsdmodule)              96c   461c   485c
vol
  LIFDAM(lifmodule)                197d   235c   237c   251c   252c   263c   268c   283c   284c   291c   292c   294c   327c   344c
                                   352c   489c   528c   567c   571c   573c   574c   575c  1141c  1144c
  M68KSYS(ci)                      914d   918c
  TAPEBKUP(cs80tbdvr)              713d   718c   730c   740c   742c   745c
  TAPEBKUP(cs80tbr)                 24d    55d    66c
vol_bytes
  CTABLE                          1543c  1557c  1577c  1664s  1674s  1684s  1696s  1703s  1714s  1719s  1730s  1741s  1757s
  CTABLE(ctr)                      313d   604d   611c
vol_label
  MIUI                             396d   431d
vol_name
  SWV01                              7d    11c    12c    13c    18c    20c
vol_num
  MIUI                             415d   449d
vol_offset
  CTABLE                          1541c  1555c  1575c  1662s  1672s  1682s  1694s  1701s  1712s  1717s  1728s  1739s  1755s
  CTABLE(ctr)                      314d   615d   617c
vol_stamp
  MIUI                             404d   439d
volid
  LIFDAM(lifmodule)                198d   265c   268c   269c   270c   309c   566c  1108c
volname
  INITLOAD(loader)                 819d   832d
  LIBRARIAN                  ***  3157c
  LIFDAM(lifmodule)                 44d   235c   237c   263c   268c   511c  1141c
  M68KSYS(ci)                      670d   719c   723c
  UCSD_DAM(ucsdmodule)             125d   130c   131c   132c
volnumber
  INITLOAD(bootdammodule)    ***   565d
  LIFDAM(lifmodule)                 66d   618c   943c
volpack
  SRMAM(srmammodule          ***   157c
  SRMDAM(srmdammodule)       ***   396c
  SRM_DRV                    ***  2064d
  SRM_DRV(srm)               ***  1107d
volpass
  SRMDAM(srmdammodule)              462d   489c   493c   505d   524c   530c   571d   584c   589c   650d   726c   727c   728c   808c
                                    840c   905d   922c   965d   979c   980c   993d  1006c  1007c  1024d  1040c  1081c  1086c  1096c
                                   1123c  1129c  1134c  1143c  1146c  1159c  1164c  1175d  1201c  1202c  1203c  1213d  1233c  1235c
                                   1248d  1256c  1257c  1266d  1299c  1302c  1313d  1332d  1358c  1367c  1391c  1392c  1405c  1421c
                                   1424c
volpass2
  SRMDAM(srmdammodule)             1214d  1234c  1236c
volsize
  LIFDAM(lifmodule)                 205d   208c   409c   569c   809c   817c   887c  1045c
  M68KSYS(ci)                       782d   800c   806c   816c
volume
  M68KSYS(ci)                       590d   598c
volume_copy_data
  TAPEBKUP(cs80tbdvr)        ***   900c
  TAPEBKUP(cs80tbr)                  31d   209d   251c
volume_ert
  MINIT(cs80ir)              ***   551d
  MINIT(qminit)              ***  1187c

volume_ert_passes
  MINIT(qminit)                     896d  1065c  1076c  1183c  1187c
volume_header_type
  SRM_DRV(srm)                      335d   376d   399d   422d   439d   493d   519d   537d   604d   646d   695d   784d  1491d
volume_info_type
  SRM_DRV(srm)               ***    344d
volume_label
  SRM_DRV(srm)               ***    349d
volume_name
  SRMDAM(srmdammodule)       ***    404c
  SRM_DRV(srm)                      341d   791d  1507c
volume_name_header
  SRM_DRV                           1803c  1920c  1981c  2073c
  SRM_DRV(srm)                       376d   399d   422d   439d   493d   519d   537d   604d   646d   695d   784d  1569c  1607c  1642c
                                    1670c  1754c
volume_num
  SRM_DRV(srm)                      263d  1505c
volumes_byte
  CS80(cs80dvr)                    1037d  1066c  1069c
vpass
  SRM_DRV(srm)                     1030d  1659d  1671c
vptr
  INIT(fs)                          746d   811d   812d   833d   835d
  LIBRARIAN                        1479d  1492c  1493c  1495c  1496c  1547c  1548c  1599c  1600c  1634d  1638c  1639c  1640c  1644c
                                   1645c
vsize
  LIFDAM(lifmodule)                 200d   207c   208c   292c   293c
vt
  INIT(misc)                 ***    166d
vu
  TAPEBKUP(cs80tbdvr)               711d   718c   719c
vvname
  LIFDAM(lifmodule)                 227d   232c   237c   255c
vvvv
  CS80(cs80)                 ***    122d
  TAPEBKUP(cs80tbdvr)        ***    718c
  TAPEBKUP(cs80tbr)                 239c   244c
vxmax
  LIB(dgl_lib)                    20019d 20286d 20295c 20301c 20308c
vxmin
  LIB(dgl_lib)                    20019d 20286d 20295c 20299c 20307c
vymax
  LIB(dgl_lib)                    20019d 20286d 20295c 20302c 20310c
vymin
  LIB(dgl_lib)                    20019d 20286d 20295c 20300c 20309c
w
  AMIGO(amigodvr)            ***    896c
  AMIGO(csamigo)             ***    109d
  F9885(f9885dvr)                    55d   127c   129c   151c   178c   196c
  INITLOAD(loader)                  846d   868d  1235c
  LIBRARIAN                         441d   491c   499c  1196c  1197c  1204c  1205c  1214c  1215c  1240c  1241c  1691c  1692c  1789c
                                   1834c  1835c  1841c  2492c  2566c  2577c  2690c  2691c  2692c
  MINIT(hminit)              ***    482c
  MIUI                               15d   175c   332c   349c   474c
  MOREFSYS(mfs)                      19d    22d   560d   581c   589c   590c   660c   661c   667d   673c
w3pad
  F9885(f9885dvr)                    75d    77d
w_array
  GLE_TYPES(gle_types)              1249d  1272d
w_loc_echo_x
  DGL_INQ(dgl_inq)           ***   6260c
  DGL_VARS(dgl_vars)         ***   1268d
  GEN(dgl_gen)                      3259c  3261c  3263c  3371c  3377c
  LIB(dgl_lib)                    20870c 20874c
```

```
w_loc_echo_y
  DGL_INQ(dgl_inq)              ***   6261c
  DGL_VARS(dgl_vars)            ***   1269d
  GEN(dgl_gen)                        3258c  3261c  3262c  3372c  3377c
  LIB(dgl_lib)                       20871c 20874c
wait
  A804XDVR(a804xdvr)                   113d   120c
  SRM_DRV                             1883d  1885c  1894c
  SRM_DRV(srm)                  ***   1076d
wait4batreadready
  BAT(bat)                             59d    87c
wait4batready
  BAT(bat)                             51d    70c    75c
wait4kbdready
  A804XDVR(a804xdvr)                   131d   139c   143c
wait_for_lock
  SRM_DRV                       ***   1894c
  SRM_DRV(srm)                  ***    631d
wait_for_ppol
  AMIGO(amigodvr)                      718d   752c   754c
wait_ready
  GLE_RGL(gle_ras_out)                8113d  8124c  8140c  8235c
waitforlock
  LOCKMOD(lockmodule)                  32d    78d
waitforlocktimeout
  SRM_DRV(srm)                         995d  1223c  1243c  1417c
waitingforlock
  SRM_DRV                       ***   1885c
  SRM_DRV(srm)                        1127d  1416c
warning_issued
  MIUI                                 221d   231c   239c   242c   269c
  TAPEBKUP                            1108d  1118c  1126c  1129c  1156c
wbytes
  EDRIVER(edriver)                     102d   242c   243c   244c   246c   249c
wd
  SRM_DRV                             1906d  1924c  1969d  1985c
  SRM_DRV(srm)                        1009d  1017d  1025d  1045d  1081d  1093d  1514d  1521c  1551d  1575c  1589d  1613c  1627d  1646c
                                      1732d  1758c
wd_ptr
  GLE_RGL(gle_ras_out)          ***   8097d
wdata
  F9885(f9885dvr)               ***    194c
wdmac0
  F9885(f9885dvr)                      75d    77d
wdmac1
  F9885(f9885dvr)                      75d    77d
wenab
  F9885(f9885dvr)                      75d    77d
wendbuffer
  ASCII(asciimodule)                   64d    93c   247c
wendfile
  ASCII(asciimodule)                  125d   273c   299c
wfm
  TAPEBKUP(cs80tbr)                   142d   148c   149c   150c
whatfiles
  M68KSYS(ci)                         654d  1139c
whichitflang
  A804XDVR(a804xdvr)                  186d   224c
whichnonitflang
  A804XDVR(a804xdvr)                  193d   223c
whole
  DGL_HPGL(dgl_hpgl)                17247d 17289c
  DGL_RAS(dgl_raster)               17551d 17608c
  GLE_HPGLI(gle_hpgl_in)            18230d 18265c

whole_sector_bytes
  INITLOAD(mini)                       411d   431c   432c   433c
whole_sectors
  INITLOAD(mini)                       410d   421c   422c   426c   428c   430c   431c
wholeword
  CRT(crt)                             321c   335c   344c   485c   509c   514c   520c   524c   530c   532c   537c   588c   608c   615c
                                       619c   624c   628c   675c
  SYSDEVS(sysdevs)              ***     49d
width
  CRT(crt)                              87d   133d   187c   190c   192c   196c   669c   671c   673c
  GCRT(crtb)                            66d   126d   128d   129d   130d   131d   503c
  GLE_RGL(gle_ras_out)                8136d  8139c  8141c
  LIB(dgl_lib)                       20071d 20816d 20830c 20834c 20838c
  M68KSYS(ci)                          462c   493c   505c  1089c
  MINIT(mminit)                 ***    159c
  SYSDEVS(sysdevs)              ***     88d
width_reg
  GLE_RGL(gle_ras_out)                8141c  8388c
  GLE_TYPES(gle_types)          ***   1269d
window
  A804XDVR(a804xdvr)                   126d   151d
  AMIGO(amigodvr)               ***    431d
  ASCII(asciimodule)                     7d    14d
  BUBBLES(bubble)                       30d    54d
  CRT(crt)                      ***    402d
  CS80(cs80dvr)                 ***    988d
  EDRIVER(edriver)                       8d    76d   308d
  EPROMS(eproms)                         9d    13d
  ETU                           ***    564d
  F9885(f9885dvr)               ***     40d
  GCRT(crtb)                    ***    272d
  INIT(fs)                              763d   783d   784d   786d   787d   789d   809d   810d   829d   831d
  INIT(initunits)                     2139d  2148d  2152d  2156d  2228d
  INIT(misc)                           484d   516d   535d
  INITLOAD(bootdammodule)              545d   592d   748d
  INITLOAD(mini)                       309d   384d
  INITLOAD(sysglobals)                  60d    61d    69d
  KEYS(keys)                    ***     72d
  LIBRARIAN                            131d   137d  2740d
  M68KSYS(ci)                          306d   731d
  PRINTER(prtdvr)                       40d    47d
  SRMAM(srmamodule)                     41d   179d
  SYSDEVS(sysdevs)                     373d   377d   408d   433d   485d
  UCSD_AM(ucsd_am)              ***     41d
window_delta_x
  GEN(dgl_gen)                        3407d  3432c  3436c  3443c
window_delta_y
  GEN(dgl_gen)                        3408d  3433c  3437c  3447c
window_lim
  DGL_INQ(dgl_inq)              ***   6286c
  DGL_VARS(dgl_vars)            ***   1063d
  GEN(dgl_gen)                        3369c  3420c  3421c  3430c  3555c  3558c  3570c  3574c
  LIB(dgl_lib)                       20333c 20863c 21149c 21151c 21412c
windowp
  A804XDVR(a804xdvr)            ***     67d
  EDRIVER(edriver)              ***     92d
  EPROMS(eproms)                ***     35d
  ETU                           ***    864d
  GCRT(crtb)                    ***    126d
  INIT(fs)                              763d   776d
  INITLOAD(sysglobals)                   61d    73d
  LIFDAM(lifmodule)             ***    388d
  SYSDEVS(sysdevs)              ***     56d
  UCSD_DAM(ucsdmodule)          ***    571d
```

```
windowregcopy
  GLE_RGL(gle_ras_cut)          ***   8139c
  SYSDEVS(sysdevs)              ***    134d
winptr
  EDRIVER(edriver)                     92d   285c
with_bkgnd_do
  TAPEBKUP(cs80tbdvr)                 345d   383c   389c   407c   425c   448c   689c   907c   925c   943c
wlp
  INITLOAD(loader)                    903d  1229c
wnextbyte
  ASCII(asciimodule)                   89d   107c   108c   131c   134c   241c   242c   306c
word
  CTABLE(scanstuff)                  1094d  1105c  1120c
  DISCHPIB(dischpib)                  388d   406c
  INITLOAD(loader)                    845d   846d   848d   868d
  LIFDAM(lifmodule)                    38d    43d
word15
  INITLOAD(bootdammodule)             554d   565d
  LIFDAM(lifmodule)                    40d    66d
word_array
  GLE_RGL(gle_ras_out)               8094d  8097d
wordcount
  HEAP1(hpm)                          136d   161c   180c   189c
wordext
  LIBRARIAN                           448d   491c   530c   781c  1222c  1233c
wordlen
  HEAP1(hpm)                          193d   198c   199c   200c   202c   203c   204c   210d   220c   258c   269c
wordlist
  INITLOAD(loader)                    848d   849d
wordlistptr
  INITLOAD(loader)                    849d   903d  1021d  1173d  1214d  1221d  1252d
wordrec
  INITLOAD(loader)                    846d   847d  1235c
  LIBRARIAN                     ***  2689c
wordrecptr
  INITLOAD(loader)                    847d   896d
  LIBRARIAN                     ***  1789c
words
  F9885(f9885dvr)                     113d   159c   160c   161c   178c   212c   213c
wordsdisposed
  HEAP1(hpm)                           80d   124d   189c   200c   258c   294c
wordsiz
  LIBRARIAN                           417d   698c   717c   852c   862c   866c   871c   877c   935c   940c   955c   968c  1087c  1088c
                                     1098c
wordsize
  HEAP1(hpm)                          286d   288c   290c   293c   294c   295c   298c   305d   309c   310c   313c
work
  DGL_POLY(dgl_poly)                20317d 20594c 20694c 20844d 20902c
work_mult
  DGL_POLY(dgl_poly)                20914d 20985c
work_ptr
  DGL_POLY(dgl_poly)                20915d 20985c 21033d 21041c 21052c 21079d 21087c 21097c 21124d 21131c 21133c 21155d 21162c 21164c
work_ptr_def
  DGL_POLY(dgl_poly)                20046d 20051d 20915d 21033d 21079d 21124d 21155d
workfid
  M68KSYS(ci)                          47d   631c
workfile
  M68KSYS(ci)                         626d   634c   637c   640c   643c
working_directory
  SRM_DRV(srm)                        268d  1521c
```

```
working_iorval
  CS80(cs80dsr)                       639d   653c   671c   675c   677c   680c   684c   686c   688c   690c   693c   696c   703c   744c
                                      752c
  TAPEBKUP(cs80tbdvr)                 519d   533c   551c   555c   557c   560c   564c   566c   568c   570c   573c   576c   585c   626c
                                      633c   699d   761c   783c   787c   789c   792c   796c   798c   800c   802c   805c   808c   817c
                                      857c   864c
working_verify_length
  TAPEBKUP(cs80tbdvr)                 512d   639c   640c   674c   679c
workstring
  DGL_INQ(dgl_inq)                   6165d  6455c  6456c  6461c  6482c  6483c  6488c
world_int_cpx
  DGL_INQ(dgl_inq)              ***   6275c
  DGL_VARS(dgl_vars)            ***   1253d
  LIB(dgl_lib)                       20628c 20659c 21019c 21429c
world_int_cpy
  DGL_INQ(dgl_inq)              ***   6276c
  DGL_VARS(dgl_vars)            ***   1254d
  LIB(dgl_lib)                       20629c 20660c 21019c 21430c
world_real_cpx
  DGL_INQ(dgl_inq)              ***   6280c
  DGL_VARS(dgl_vars)            ***   1258d
  LIB(dgl_lib)                       20566c 20596c 21018c
world_real_cpy
  DGL_INQ(dgl_inq)              ***   6281c
  DGL_VARS(dgl_vars)            ***   1259d
  LIB(dgl_lib)                       20567c 20597c 21018c
wptr
  EDRIVER(edriver)                     90d   244c   246c   249c   271c
  EPROMS(eproms)                ***    35d
wrapup
  LIBRARIAN                          3699d  3742c  3746c
write_file_mark
  TAPEBKUP(cs80tbdvr)           ***   401c
  TAPEBKUP(cs80tbr)                    27d   133d   152c
write_program_header
  MIUI                                 41d   545c
write_protect
  CS80(cs80)                    ***   169d
  CS80(cs80dsr)                 ***   689c
  TAPEBKUP(cs80tbdvr)                  569c   801c
write_to_status
  GLE_RGL(gle_ras_out)          ***  8264c
  GLE_TYPES(gle_types)          ***  1277d
writeappend
  INIT(fs)                      ***   745d
writeblock
  F9885(f9885dvr)                      51d    57d   174c
writeblocks
  LIBRARIAN                          2228d  2230c  2231c  2232c  2233c  2267d  2269c  2270c  2271c  2272c
writebuffer
  IOLIB(general_4)                   1237d  1584d  1652c
writebuffer_string
  IOLIB(general_4)                   1243d  1639d
```

```
writebytes
  AMIGO(amigodvr)              ***    592c
  ASCII(asciimodule)                   37c   290c
  BUBBLES(bubble)                      88c   100c
  CRT(crt)                            428c   443c   453c
  CS80(cs80dvr)                      1157c  1182c  1288c  1293c  1307c  1316c  1385c
  EPROMS(eproms)               ***     76c
  F9885(f9885dvr)                     172c   223c   277c
  GCRT(crtb)                          298c   313c   324c
  INIT(misc)                          489c   548c   582c   608c
  INITLOAD(mini)                      427c   443c   505c
  INITLOAD(sysglobals)         ***     66d
  KEYS(keys)                   ***     94s
  LIBRARIAN                           133c  2227d  2230c  2231c  2234c  2235c  2236c  2237c  2242d  2250c  2251c  2252c  2254c  2267d
                                     2270c  2273c  2274c  2742c
  LIFDAM(lifmodule)                   352c   447c   528c   530c  1144c
  M68KSYS(ci)                         352c   737c
  MINIT(bminit)                ***     86c
  MIUI                         ***    493c
  PRINTER(prtdvr)              ***    304c
  SRMAM(srmammodule)           ***    199c
  UCSD_AM(ucsd_am)                     91c   180c   182c
  UCSD_DAM(ucsdmodule)                503c   504c   606c
  UNITIO(uio)                  ***     62c
writechar
  GLE_HPIB(gle_hpib_io)              10072d 10521c 10522c
  IOLIB(general_1)                     240d   293d
  IOLIB(general_2)                    1031c  1032c  1044c  1045c
writechars
  ASCII(asciimodule)                   232d   292c   295c
writectrl
  HPHIL(hphil)                          38d   145c   172c
writedate
  LIBRARIAN                            106d   231c   233c   331c   349c  2729c
writedirectory
  LIBRARIAN                           2878d  2995c
writeendline
  ASCII(asciimodule)                    99d   129c   302c
writeeol
  ASCII(asciimodule)           ***    300c
  CRT(crt)                            442c   448c
  GCRT(crtb)                          312c   318c
  INIT(misc)                   ***    580c
  INITLOAD(sysglobals)         ***     66d
  KEYS(keys)                   ***     92s
  M68KSYS(ci)                  ***    351c
  PRINTER(prtdvr)              ***    299c
  UCSD_AM(ucsd_am)             ***    178c
writefilemark
  TAPEBKUP(cs80tbdvr)                 395d   407c
  TAPEBKUP(cs80tbr)                   145d   149c
writeln_proc
  TAPEBKUP                           1175d  1204c
writenumber
  IOLIB(general_2)                    799d   913d  1030c
writenumberln
  IOLIB(general_2)                    813d  1024d
writeonly
  INIT(fs)                     ***    745d
writeprotected
  F9885(f9885dvr)                      62d   223c
writeregcopy
  SYSDEVS(sysdevs)             ***    135d

writesector
  LIBRARIAN                          2740d  2818c
writestring
  IOLIB(general_2)                    808d   992d  1043c
writestringln
  IOLIB(general_2)                    816d  1037d
writeword
  IOLIB(general_1)                    244d   317d
writing_previous_char
  PRINTER(prtdvr)                      72d   238c   252c   254c   256c   284c
wrongbyte
  INIT(ldr)                    ***   2494c
  INITLOAD(loader)                   1055d  1549c
wrongrec
  INIT(ldr)                    ***   2495c
  INITLOAD(loader)                   1075d  1550c
wrtchar
  PRINTER(prtdvr)                     145d   255c   301c   302c   307c
ww_reg
  GLE_TYPES(gle_types)         ***   1274d
wword
  F9885(f9885dvr)                      75d    77d
wx
  GEN(dgl_gen)                        3036d  3329d  3335c
  LIB(dgl_lib)                       20074d 20091d 20092d 20246d 20254c 20263d 20272c 20846d 20866c 20870c
wxmax
  LIB(dgl_lib)                       20020d 20321d 20330c 20336c
wxmin
  LIB(dgl_lib)                       20020d 20321d 20330c 20335c
wy
  GEN(dgl_gen)                        3036d  3329d  3336c
  LIB(dgl_lib)                       20074d 20091d 20092d 20246d 20255c 20263d 20273c 20846d 20867c 20871c
wymax
  LIB(dgl_lib)                       20020d 20321d 20330c 20338c
wymin
  LIB(dgl_lib)                       20020d 20321d 20330c 20337c
x
  CRT(crt)                             293d   295c   298d   300c   301c   302c   308d   310c   316d   318c   321c   323c   324c
  DC_DRV(extdc)                        200d   202d   206d
  DC_DRV(intdc)                        246d   248d   251d   312d   317c   330d   335c   349d   351c
  DGL_VARS(dgl_vars)                  1142d  1144d
  DI_DRV(extdi)                        157d   159d   162d   167d   171d   172d
  ETU                                 1166d  1179c  1208c
  GCRT(crtb)                           117d   121d   122d   132d   202d   204c   207d   209c   210c   211c   217d   219c   225d   228c
                                       230c   231c   234c   236c
  GEN(dgl_gen)                        3026d  3029d  3192d  3220c  3223c  3227c  3228c  3229c  3233c  3239c  3244c  3274d  3286c  3292c
                                      3302d  3308c
  G_DRV(extg)                          174d   176d   179d   186d
  H_DRV(exth)                          172d   174d   177d   182d   186d   187d
  INIT(fs)                             797d   798d
  KERNEL(iocomasm)                     705d   707d   709d   711d
  LIB(dgl_lib)                       20018d 20043d 20044d 20123d 20128c 20134d 20139c 20162d 20165c 20166c 20168c 20169c 20219d 20226c
                                     20234c 20237c 20348d 20356c 20362c 20557d 20566c 20570c 20587d 20596c 20600c
  M68KSYS(ci)                          111d   128c   137c   566d   567c
  MINIT(mminit)                        148d   157c
  MIUI                                  15d   181c   351c   474c
  MOREFSYS(mfs)                         18d    19d    21d    22d    40d   397d   503c   510c   515d   528c   533d   548c   556d   560d
                                       569c   572c   582c   585c   667d   673c
  RS_DRV(rs)                           174d   176d   179d
x1
  LIB(dgl_lib)                       20850d 20853c 20854c
x2
  LIB(dgl_lib)                       20850d 20853c 20854c
```

```
x_adj
  GLE_KNOB(gle_knob_in)            18111d 18133c 18137c 18138c 18174c
x_count
  IOLIB(general_4)                  1209d  1215d  1440d  1444c  1464d  1468c
x_display_delta
  DGL_VARS(dgl_vars)          ***   1294d
  GEN(dgl_gen)                      3292c  3442c  3445c  3453c
x_display_offset
  DGL_VARS(dgl_vars)          ***   1296d
  GEN(dgl_gen)                      3293c  3444c  3452c
x_major
  DGL_POLY(dgl_poly)               20424d 20560c 20609c 20622c 20647c 20649c 20704c
x_pos
  GLE_SMARK(gle_smark)              5024d  5033d  5034d  5035d  5036d  5037d  5038d  5039d  5040d  5041d  5042d  5043d  5044d  5045d
                                    5046d  5047d  5048d  5049d  5050d  5051d  5052d  5053d  5054d  5055d  5056d  5057d  5058d  5059d
                                    5060d  5061d  5062d  5063d  5064d  5065d  5066d  5067d  5068d  5069d  5070d  5071d  5072d  5073d
                                    5074d  5075d  5076d  5077d  5078d  5079d  5156c
x_window_delta
  DGL_VARS(dgl_vars)          ***   1295d
  GEN(dgl_gen)                      3293c  3443c  3445c  3453c
xaddr
  INITLOAD(bootdammodule)            588d   648d   655c   671c   672c
  SRM_DRV(srm)                      1054d  1741d  1770c
xdata
  MOUSE(mouse)                        46d   161c   171c
xdtow_offset
  DGL_VARS(dgl_vars)          ***   1285d
  GEN(dgl_gen)                      3335c  3562c
xdtow_scale
  DGL_INQ(dgl_inq)            ***   6191c
  DGL_VARS(dgl_vars)          ***   1279d
  GEN(dgl_gen)                      3335c  3560c
xfer
  LIBRARIAN                         3054d  3480c  3650c
xfr
  CS80(cs80dvr)                     1094d  1157c  1185c  1193c  1250c  1278c  1307c
xfr_chain_semaphore
  AMIGO(amigodvr)                    622c   703c   706c
  CS80(cs80dvr)                     1090c  1115c  1117c
  DISCHPIB(bkgnd)             ***     46d
xfr_required
  CS80(cs80dvr)                     1170d  1172c  1173c  1181c  1187c  1190c
xlate_errors
  BUBBLES(bubble)                     37d    83c   102c   109c   116c
xlim
  DGL_INQ(dgl_inq)            ***   6267c
  DGL_VARS(dgl_vars)                1018d  1184d
  GEN(dgl_gen)                ***   3552c
  LIB(dgl_lib)                     20301c 20365c 20370c 20374c
xltod_scale
  DGL_VARS(dgl_vars)          ***   1288d
  GEN(dgl_gen)                      3321c  3360c
  LIB(dgl_lib)                ***   20277c
xmax
  CRT(crt)                           561c   573c   587c   593c   597c   623c   627c   640c
  DGL_INQ(dgl_inq)                  6216c  6249c  6289c  6298c
  DGL_KNOB(dgl_knot)          ***   18095c
  DGL_VARS(dgl_vars)                1012d  1173d  1178d  1188d  1193d
  GCRT(crtb)                         427c   431c   433c   436c   439c   444c   464c   466c   480c
  GEN(dgl_gen)                      3353c  3360c  3361c  3371c  3421c  3432c  3519c  3527c  3542c  3552c  3554c  3555c  3587c  3594c
                                    3616c
  LIB(dgl_lib)                     20186c 20308c 20336c 20519c 20866c 20915c 21041c 21044c 21084c 21124c 21131c 21144c 21149c 21333c
                                   21343c 21361c 21363c 21420c 21422c
  SYSDEVS(sysdevs)            ***     59d
```

```
xmin
  CRT(crt)                           561c   573c   582c   587c   593c   594c   597c   623c   634c   640c   642c   644c
  DGL_INQ(dgl_inq)                  6216c  6249c  6288c  6297c
  DGL_KNOB(dgl_knot)          ***   18094c
  DGL_POLY(dgl_poly)                20416d 20697c 20705c 20746c
  DGL_VARS(dgl_vars)                1011d  1172d  1177d  1187d  1192d
  GCRT(crtb)                         126d   128d   129d   130d   131d   427c   431c   433c   436c   444c   447c   464c   472c   480c
  GEN(dgl_gen)                      3321c  3322c  3352c  3360c  3361c  3371c  3420c  3432c  3445c  3518c  3528c  3541c  3552c  3554c
                                    3555c  3557c  3558c  3587c  3594c  3615c
  LIB(dgl_lib)                     20185c 20258c 20277c 20281c 20307c 20335c 20511c 20512c 20519c 20866c 20907c 20908c 20914c 21041c
                                   21044c 21084c 21123c 21130c 21144c 21149c 21332c 21342c 21361c 21363c 21420c 21422c
  SYSDEVS(sysdevs)            ***     59d
xminit
  MINIT(xminit)               ***   1355d
  MIUI                        ***     12d
xminitialize
  MINIT(xminit)                     1363d  1377d
  MIUI                        ***     35ic
xpos
  CRT(crt)                           272c   295c   300c   301c   302c   318c   461c   462c   463c   464c   466c   469c   471c   472c
                                     483c   486c   491c   565c   566c   568c   648c   649c   651c
  GCRT(crtb)                         199c   204c   209c   210c   211c   228c   332c   333c   334c   335c   337c   340c   342c   343c
                                     352c   354c   359c
  SYSDEVS(sysdevs)            ***   128d
xtemp
  CRT(crt)                           550d   565c   568c   648c   651c
xvalid
  MOREFSYS(mfs)                      402d   408c   429c   436c   451c   458c   467c   492c   495c   497c
xvec
  DGL_POLY(dgl_poly)               20022d 20025d 20027d 20030d 20910d 20976c 20978c 21026d 21040c 21061c 21072c 21086c 21106c 21116d
                                   21130c 21137c 21148d 21161c 21168c
  LIB(dgl_lib)                     20046d 20048d 20677d 20687c 20689c 20693d 20703c 20705c
xwtod_offset
  DGL_VARS(dgl_vars)          ***   1282d
  GEN(dgl_gen)                      3308c  3557c  3562c  3563c
  LIB(dgl_lib)                     20254c 20272c 20570c 20600c
xwtod_scale
  DGL_VARS(dgl_vars)          ***   1276d
  GEN(dgl_gen)                      3308c  3554c  3558c  3560c
  LIB(dgl_lib)                     20254c 20272c 20570c 20600c 20834c 20838c
xx
  AMIGO(csamigo)              ***   105d
xxxxx
  MOREFSYS(mfs)                      536d   544c
y
  CRT(crt)                           293d   295c   298d   303c   304c   305c   308d   310c   316d   318c   321c   324c
  DGL_RAS(dgl_raster)              17317d 17335c 17337c
  DGL_VARS(dgl_vars)                1142d  1144d
  ETU                               1166d  1179c  1208c
  GCRT(crtb)                         117d   121d   122d   132d   202d   204c   207d   212c   213c   214c   217c   219c   225d   228c
                                     234c   236c
  GEN(dgl_gen)                      3026d  3029d  3274d  3287c  3294c  3302d  3309c
  INIT(ifs)                          797d   798d
  KERNEL(iocomasm)                   706d   708d   710d
  LIB(dgl_lib)                     20018d 20043d 20044d 20123d 20129c 20134d 20140c 20162d 20165c 20166c 20168c 20169c 20219d 20227c
                                   20234c 20238c 20348d 20356c 20362c 20557d 20567c 20571c 20587d 20597c 20601c
  M68KSYS(ci)                        111d   128c   137c
  MINIT(mminit)                      148d   157c   159c
y_adj
  GLE_KNOB(gle_knob_in)            18112d 18134c 18139c 18140c 18175c
y_display_delta
  DGL_VARS(dgl_vars)          ***   1297d
  GEN(dgl_gen)                      3294c  3446c  3449c  3455c
```

```
y_display_offset
  DGL_VARS(dgl_vars)          ***   1299d
  GEN(dgl_gen)                      3295c  3448c  3454c
y_pos
  GLE_SMARK(gle_smark)              5025d  5033d  5034d  5035d  5036d  5037d  5038d  5039d  5040d  5041d  5042d  5043d  5044d  5045d
                                    5046d  5047d  5048d  5049d  5050d  5051d  5052d  5053d  5054d  5055d  5056d  5057d  5058d  5059d
                                    5060d  5061d  5062d  5063d  5064d  5065d  5066d  5067d  5068d  5069d  5070d  5071d  5072d  5073d
                                    5074d  5075d  5076d  5077d  5078d  5079d  5157c
y_window_delta
  DGL_VARS(dgl_vars)          ***   1298d
  GEN(dgl_gen)                      3295c  3447c  3449c  3455c
ydata
  MOUSE(mouse)                       47d   164c   168c
ydtow_offset
  DGL_VARS(dgl_vars)          ***   1286d
  GEN(dgl_gen)                      3336c  3578c
ydtow_scale
  DGL_INQ(dgl_inq)            ***   6192c
  DGL_VARS(dgl_vars)          ***   1280d
  GEN(dgl_gen)                      3336c  3576c
year
  CLOCK(clock)                       67c   106c
  INITLOAD(sysglobals)        ***    218d
  LIBRARIAN                          114c   119c  3711c
  LIFDAM(lifmodule)                  120c   331c   346c   550c   554c   587c
  M68KSYS(ci)                        135c   156c   266c   778d
  MIUI                                44d    65c    70c    71c    90c
  SRMDAM(srmdammodule)               368c   387d
  SRM_DRV(srm)                ***    254d
  UCSD_DAM(ucsdmodule)               146c   522c   540c   551c   592c
yencode
  CRT(crt)                           375d   386c
  GCRT(crtb)                         258d   260c
  NONUSKBD1(non_us_kbd1)      ***    104d
yenromlocation
  CRT(crt)                            42d   367c   386c
  GCRT(crtb)                         257d   260c
yes
  GLE_RGL(gle_ras_out)        ***   8341c
  GLE_TYPES(gle_types)        ***   1264d
  MINIT(midecs)                       37d    41d    50c
  MINIT(qminit)                      1072c  1084c
  MIUI                                163c   278c
  TAPEBKUP                            972d   983c  1043c  1340c  1357c  1359c  1443c  1476c  1480c
ylim
  DGL_INQ(dgl_inq)            ***   6268c
  DGL_VARS(dgl_vars)                 1019d  1185d
  GEN(dgl_gen)                ***   3567c
  LIB(dgl_lib)                      20302c 20366c 20371c 20374c
yltod_scale
  DGL_VARS(dgl_vars)          ***   1289d
  GEN(dgl_gen)                       3323c  3363c
  LIB(dgl_lib)                ***   20278c
ymax
  CRT(crt)                            561c   579c   583c   602c   622c   641c
  DGL_INQ(dgl_inq)                   6217c  6250c  6291c  6300c
  DGL_KNOB(dgl_knob)          ***   18097c
  DGL_VARS(dgl_vars)                 1014d  1175d  1180d  1190d  1195d
  GCRT(crtb)                          126d   128d   129d   130d   131d   427c   431c   433c   436c   438c   444c   446c   463c   480c
  GEN(dgl_gen)                       3355c  3363c  3364c  3372c  3421c  3433c  3495c  3503c  3544c  3567c  3569c  3570c  3587c  3594c
                                     3618c
  LIB(dgl_lib)                      20188c 20310c 20338c 20520c 20867c 20917c 21041c 21044c 21084c 21126c 21133c 21144c 21151c 21335c
                                    21345c 21361c 21363c 21420c 21422c
  SYSDEVS(sysdevs)            ***     59d
```

```
ymin
  CRT(crt)                            561c   575c   583c   594c   602c   622c   634c   641c   643c   645c
  DGL_INQ(dgl_inq)                   6217c  6250c  6290c  6299c
  DGL_KNOB(dgl_knob)          ***   18096c
  DGL_POLY(dgl_poly)                20416d 20698c 20707c 20746c
  DGL_VARS(dgl_vars)                 1013d  1174d  1179d  1189d  1194d
  GCRT(crtb)                          126d   128d   129d   130d   131d   427c   431c   433c   436c   438c   444c   446c   463c   472c
                                      480c
  GEN(dgl_gen)                       3323c  3324c  3354c  3363c  3364c  3372c  3420c  3433c  3449c  3494c  3504c  3543c  3567c  3569c
                                     3570c  3573c  3574c  3587c  3594c  3617c
  LIB(dgl_lib)                      20187c 20259c 20278c 20282c 20309c 20337c 20514c 20515c 20520c 20867c 20909c 20910c 20916c 21041c
                                    21044c 21084c 21125c 21132c 21144c 21151c 21334c 21344c 21361c 21363c 21420c 21422c
  SYSDEVS(sysdevs)            ***     59d
ypos
  CRT(crt)                            272c   295c   303c   304c   305c   318c   449c   450c   461c   462c   463c   464c   465c   466c
                                      468c   469c   471c   474c   483c   488c   489c   491c   565c   566c   568c   648c   649c   651c
  GCRT(crtb)                          199c   204c   212c   213c   214c   228c   319c   320c   332c   333c   334c   335c   336c   337c
                                      339c   340c   342c   345c   352c   356c   357c   359c
  SYSDEVS(sysdevs)            ***    129d
yr
  CLOCK(clock)                        50d    56c    57c    64c    67c   101d   106c   108c   109c
ytemp
  CRT(crt)                           550d   565c   568c   648c   651c
yvec
  DGL_POLY(dgl_poly)                20022d 20025d 20027d 20030d 20910d 20976c 20978c 21026d 21040c 21061c 21072d 21086c 21106c 21116d
                                    21130c 21137c 21148d 21161c 21168c
  LIB(dgl_lib)                      20046d 20048d 20677d 20687c 20689c 20693d 20703c 20705c
ywtod_offset
  DGL_VARS(dgl_vars)          ***   1283d
  GEN(dgl_gen)                       3309c  3573c  3578c  3579c
  LIB(dgl_lib)                      20255c 20273c 20571c 20601c
ywtod_scale
  DGL_VARS(dgl_vars)          ***   1277d
  GEN(dgl_gen)                       3309c  3569c  3574c  3576c
  LIB(dgl_lib)                      20255c 20273c 20571c 20601c 20835c 20839c
z
  CS80(cs80)                          345d   500d   514c
z_type
  CS80(cs80)                          234d   345d   500d   507d
zap_assigned_unit
  CTABLE                      ***   1808c
  CTABLE(ctr)                         306d   868d   903c
zapspaces
  ETU                                 158c   881c
  INIT(fs)                    ***     755d
  M68KSYS(ci)                 ***     675c
zaptypeahead
  M68KSYS(ci)                         565d   574c   861c
zbadblock
  AMIGO(amigodvr)             ***     873c
  BUBBLES(bubble)             ***      47c
  CS80(cs80dsr)               ***     696c
  F9885(f9885dvr)             ***     236c
  INIT(misc)                  ***     305c
  INITLOAD(bootdammodule)     ***     600c
  INITLOAD(mina)              ***     364d
  INITLOAD(sysglobals)        ***     184d
  MIUI                        ***     120d
  TAPEBKUP                    ***    1051d
  TAPEBKUP(cs80tbdvr)                 576c   655c   684c   808c
```

```
zbadama
  AMIGO(amigodvr)                 ***    605c
  CS80(cs80dvr)                   ***   1389c
  F9885(f9885dvr)                 ***    119c
  INIT(misc)                      ***    326c
  INITLOAD(sysglobals)            ***    188d
zbadhardware
  AMIGO(amigodvr)                        889c   914c
  BUBBLES(bubble)                         41c    42c
  CS80(cs80dsr)                          651c   671c
  DISCHPIB(dischpib)              ***    353c
  F9885(f9885dvr)                 ***    242c
  INIT(misc)                      ***    324c
  INITLOAD(bootdammodule)         ***    601c
  INITLOAD(mini)                         344d   352d   353d   354d   355d   356d   369d
  INITLOAD(sysglobals)            ***    188d
  MINIT(cs80ir)                          761c   808c
  MINIT(hminit)                          309c   475c
  MIUI                            ***    211c
  TAPEBKUP(cs80tbdvr)                    531c   551c   759c   783c
  TAPEBKUP(cs80tbr)               ***    286c
zbadmode
  AMIGO(amigodvr)                 ***    609c
  BUBBLES(bubble)                 ***     45c
  CS80(cs80dsr)                   ***    675c
  CS80(cs80dvr)                   ***   1249c
  EPROMS(eproms)                  ***    111c
  F9885(f9885dvr)                 ***    283c
  INIT(misc)                      ***    307c
  INITLOAD(bootdammodule)                760c   768c
  INITLOAD(mini)                  ***    511c
  INITLOAD(sysglobals)            ***    184d
  MINIT(qminit)                          986c  1134c
  PRINTER(prtdvr)                 ***    312c
  TAPEBKUP(cs80tbdvr)                    555c   787c
zbit
  CS80(cs80)                             507d   514c
zcatchall
  AMIGO(amigodvr)                        477c   528c   802c   906c   910c   926c   931c   943c
  AMIGO(csamigo)                  ***    212c
  BUBBLES(bubble)                 ***     49c
  CS80(cs80dsr)                          744c   756c
  CS80(cs80dvr)                   ***   1129c
  F9885(f9885dvr)                        132c   156c   186c   202c   209c   225c   237c   245c
  INIT(misc)                      ***    325c
  INITLOAD(bootdammodule)         ***    602c
  INITLOAD(mini)                  ***    368d
  INITLOAD(sysglobals)            ***    188d
  MINIT(hminit)                   ***    492c
  MINIT(qminit)                   ***   1137c
  TAPEBKUP(cs80tbdvr)                    626c   646c   857c   879c
  TAPEBKUP(cs80tbr)               ***    294c
zchr
  KEYS(keys)                              76d    87c
zdata
  MOUSE(mouse)                    ***     48d
zero
  M68KSYS(ci)                     ***    773d
  MIUI                                   402d   437d
zero_fields
  CTABLE                                1243d  1253c  1255c  1257c  1259c  1261c
zero_out_na_fields
  CTABLE                                1239d  1392c  1485c
```

```
zero_parity
  IOLIB(serial_3)                       2753c  2775c
  KERNEL(iodeclarations)          ***    404d
zerodate
  SRMDAM(srmdammodule)                   387d   411c   413c
zerodatetime
  LIFDAM(lifmodule)                      585d   616c
zeromem
  INIT(ldr)                       ***   2472c
  INITLOAD(loader)                      1089d  1567d  1635c
  SEGMENTER(asm)                  ***    208c
zerotime
  SRMDAM(srmdammodule)                   388d   412c   414c
zinittail
  CS80(cs80dsr)                   ***    686c
  INIT(misc)                      ***    321c
  INITLOAD(mini)                         342d   343d
  INITLOAD(sysglobals)            ***    187d
  MINIT(hminit)                          325c   327c   361c   379c   395c   400c   405c   426c   449c   460c
  MINIT(qminit)                   ***   1254c
  TAPEBKUP(cs80tbdvr)                    566c   798c
zit1
  SRM_DRV(srm)                    ***    297d
zit2
  SRM_DRV(srm)                    ***    298d
zmediumchanged
  AMIGO(amigodvr)                        607c   668c   903c
  CS80(cs80dsr)                   ***    703c
  CS80(cs80dvr)                   ***   1391c
  F9885(f9885dvr)                        138c   280c
  INIT(misc)                      ***    355c
  INITLOAD(mini)                         349d   464c   508c
  INITLOAD(sysglobals)            ***    195d
  LIFDAM(lifmodule)               ***    214c
  MINIT(hminit)                          474c   480c
  TAPEBKUP(cs80tbdvr)                    585c   817c
znoblock
  AMIGO(amigodvr)                        828c   874c   923c
  CS80(cs80dsr)                   ***    693c
  F9885(f9885dvr)                        228c   235c
  INIT(misc)                      ***    336c
  INITLOAD(bootdammodule)         ***    758c
  INITLOAD(mini)                         351d   357d   360d   361d   362d   363d
  INITLOAD(sysglobals)            ***    191d
  MIUI                            ***    120d
  TAPEBKUP                        ***   1051d
  TAPEBKUP(cs80tbdvr)                    573c   805c
```

```
znodevice
  AMIGO(amigodvr)                    470c   478c   483c   516c   519c   529c   533c   535c   567c   880c   892c
  AMIGO(csamigo)          ***        172c
  BUBBLES(bubble)                     46c    64c
  CS80(cs80dsr)                      677c   923c
  CS80(cs80dvr)                     1005c  1104c  1360c
  DISCHPIB(dischpib)                 222c   249c
  EPROMS(eproms)                      47c   102c
  F9885(f9885dvr)                     89c   217c   255c   262c
  INIT(initunits)                   2230c  2233c
  INIT(misc)              ***        320c
  INITLOAD(bootdammodule) ***        597c
  INITLOAD(mini)                     370d   483c   490c   493c
  INITLOAD(sysglobals)    ***        187d
  MINIT(qminit)           ***        958c
  MIUI                    ***        210c
  PRINTER(prtdvr)                    105c   140c   290c
  SRMAM(srmammodule)                 151c   159c   189c
  SRMDAM(srmdammodule)    ***       1645c
  SRM_DRV(srm)                      1152c  1470c
  TAPEBKUP(cs80tbdvr)                351c   557c   789c
znomedium
  AMIGO(amigodvr)         ***        893c
  CS80(cs80dvr)           ***       1238c
  F9885(f9885dvr)         ***        220c
  INIT(misc)              ***        338c
  INITLOAD(bootdammodule) ***        598c
  INITLOAD(mini)                     345d   346d   347d   348d   350d   367d
  INITLOAD(sysglobals)    ***        191d
  MINIT(hminit)           ***        481c
znosuchblk
  AMIGO(amigodvr)         ***        922c
  BUBBLES(bubble)         ***         44c
  CS80(cs80dsr)           ***        680c
  EPROMS(eproms)          ***         54c
  INIT(misc)              ***        319c
  INITLOAD(sysglobals)    ***        187d
  TAPEBKUP(cs80tbdvr)                560c   792c
znotready
  AMIGO(amigodvr)         ***        891c
  BUBBLES(bubble)                     71c    88c
  CS80(cs80dsr)           ***        688c
  INIT(misc)              ***        337c
  INITLOAD(bootdammodule) ***        599c
  INITLOAD(sysglobals)    ***        191d
  SRM_DRV(srm)            ***       1181c
  TAPEBKUP                          1401c  1403c
  TAPEBKUP(cs80tbdvr)                568c   800c
zprotected
  AMIGO(amigodvr)                    882c   897c
  CS80(cs80dsr)           ***        690c
  EPROMS(eproms)          ***         76c
  F9885(f9885dvr)         ***        224c
  INIT(misc)              ***        322c
  INITLOAD(mini)                     358d   359d
  INITLOAD(sysglobals)    ***        187d
  MINIT(hminit)           ***        482c
  TAPEBKUP(cs80tbdvr)                570c   802c
zstrangei
  BUBBLES(bubble)         ***         43c
  INIT(misc)              ***        323c
  INITLOAD(mini)                     365d   366d
  INITLOAD(sysglobals)    ***        188d

ztimeout
  BUBBLES(bubble)         ***         48c
  DISCHPIB(dischpib)      ***        225c
  INIT(misc)              ***        308c
  INITLOAD(sysglobals)    ***        184d
  PRINTER(prtdvr)                    132c   199c   316c
  SRM_DRV(srm)            ***       1366c
zuninitialized
  AMIGO(amigodvr)                    671c   899c
  CS80(cs80dsr)           ***        684c
  CS80(cs80dvr)           ***       1242c
  INIT(misc)              ***        335c
  INITLOAD(sysglobals)    ***        191d
  MIUI                    ***        121d
  TAPEBKUP                ***       1052d
```

A804XDVR

Description

A804XDVR contains the routines for talking to the 8041 and 8042 keyboard processors, as well as the procedure through which all interrupts are funneled and sent to the appropriate places.

Requirements

SYSGLOBALS, SYSDEVS, ISR, and ASM.

```
    1:S          (*
    2:S
    3:S           (c) Copyright Hewlett-Packard Company, 1983.
    4:S          All rights are reserved.  Copying or other
    5:S          reproduction of this program except for archival
    6:S          purposes is prohibited without the prior
    7:S          written consent of Hewlett-Packard Company.
    8:S
    9:S                                                          .
   10:S                       RESTRICTED RIGHTS LEGEND
   11:S
   12:S          Use, duplication, or disclosure by the Government
   13:S          is subject to restrictions as set forth in
   14:S          paragraph (b) (3) (B) of the Rights in Technical
   15:S          Data and Computer Software clause in
   16:S          DAR 7-104.9(a).
   17:S
   18:S          HEWLETT-PACKARD COMPANY
   19:D        0 Fort Collins, Colorado                          *)
   20:S
   21:D        0 $SYSPROG$
   22:D        0 $heap_dispose off$
   23:D        0 $iocheck off$
   24:D        0 $range off$ $ovflcheck off$
   25:D        0 $debug off$
   26:D        0 $stackcheck off$
   27:S
   28:D        0 PROGRAM A804XINIT;
   29:D        1 MODULE A804XDVR;
   30:D        1 $SEARCH 'INITLOAD','ASM','SYSDEVS','INIT'$
   31:D        1 IMPORT SYSGLOBALS,SYSDEVS,ISR,ASM;
   32:D        1 EXPORT
   33:D        1 TYPE
   34:D        1   DATAHOOKTYPE = PROCEDURE(DATA:BYTE);
   35:D        1 VAR
   36:D     -8 1   DATAHOOK : DATAHOOKTYPE;
   37:D    -16 1   STATUS5HOOK : KBDHOOKTYPE;
   38:D    -24 1   STATUS6HOOK : KBDHOOKTYPE;
   39:S
   40:D    -24 1   PROCEDURE SENDCMD(CMD:BYTE);
   41:D    -24 1   PROCEDURE SENDDATA(DATA:BYTE);
   42:D    -24 1   PROCEDURE CMD_READ_1(CMD:BYTE; VAR DATA:BYTE);
   43:S
   44:D    -24 1   FUNCTION  INITA804X:BOOLEAN;
   45:S
   46:D    -24 1 IMPLEMENT
   47:S
   48:D    -24 1 CONST
   49:D    -24 1   UP = TRUE; DOWN = FALSE;
   50:D    -24 1 TYPE
   51:D    -24 1   STATUSTYPE = PACKED RECORD
   52:D    -24 1                  CASE INTEGER OF
   53:D    -24 1                  0:(PAD1  :0..63;
   54:D    -24 1                     BUSY  :BOOLEAN;
   55:D    -24 1                     READY :BOOLEAN);
   56:D    -24 1                  1:(STATBYTE :CHAR);
   57:D    -24 1                END;
   58:D    -24 1   STRING3 = STRING[3];
   59:S
   60:D    -24 1 VAR
```



```
   61:D    -24 1   DATAREG    [HEX('428001')] : CHAR;
   62:D    -24 1   STATUSREG  [HEX('428003')] : CHAR;
   63:D    -24 1   CMDREG     [HEX('428003')] : CHAR;
   64:D    -44 1   ISRREC          : ISRIB;
   65:D    -46 1   MAXDATA         : SHORTINT;
   66:D    -48 1   HAVEDATA        : SHORTINT;
   67:D    -52 1   DATABUFFER      : WINDOWP;
   68:D    -54 1   EXTLEFT,EXTRIGHT: BOOLEAN;
   69:D    -56 1   MASK            : BYTE;
   70:S
   71:D        1   PROCEDURE DUMMYSTATUS56(VAR STATBYTE,DATABYTE:BYTE; VAR DONE:BOOLEAN);
   72:C        2   BEGIN END;
   73:S
   74:D        1   PROCEDURE STATUSSISR(VAR STATBYTE,DATABYTE:BYTE; VAR DONE:BOOLEAN);
   75:C        2   BEGIN { IF ANYTHING WRONG THEN RESET EXTENDCHAR BITS }
   76:C        2         { OTHERWISE IGNORE IT}
   77:C        2     IF DATABYTE>127 THEN BEGIN EXTLEFT:=FALSE; EXTRIGHT:=FALSE; END;
   78:C        2   END;
   79:S
   80:D        1   PROCEDURE DATAISR(DATA:BYTE);
   81:C        2   BEGIN
   82:C        2     IF HAVEDATA<MAXDATA THEN
   83:C        3     BEGIN DATABUFFER^[HAVEDATA]:=CHR(DATA);
   84:C        3           HAVEDATA:=HAVEDATA+1;
   85:C        3     END;
   86:C        2   END;
   87:S
   88:S
   89:D        1   PROCEDURE A804XISR(ISRIBPTR : PISRIB);
   90:D        2   VAR
   91:D     -4 2     STATUS,DATA : BYTE;
   92:D     -5 2     DOIT        : BOOLEAN;
   93:C        2   BEGIN
   94:C        2     STATUS := ORD(STATUSREG);   { READ STATUS REG }
   95:C        2     DATA   := ORD(DATAREG);     { READ DATA REG }
   96:C        2     DOIT   := TRUE;
   97:C        2     CASE STATUS DIV 16 OF
   98:C        3     0,7    : { UN IMPLEMENTED OPERATIONS };
   99:C        3              { 0 RESERVED; 7 REPORT ON POWERUP }
  100:C        3     1,2,3  : CALL(TIMERISRHOOK,STATUS,DATA,DOIT);
  101:C        3     4      : CALL(DATAHOOK,DATA);    { REQUESTED DATA OTHER THAN CARAVAN }
  102:C        3     5      : CALL(STATUSSHOOK,STATUS,DATA,DOIT); { CARAVAN STATUS CHANGE }
  103:C        3     6      : CALL(STATUS6HOOK,STATUS,DATA,DOIT); { CARAVAN RAW DATA }
  104:C        3     8..11  : BEGIN
  105:C        3                CALL(KBDTRANSHOOK,STATUS,DATA,DOIT);
  106:C        3                IF DOIT THEN CALL(KBDISRHOOK,STATUS,DATA,DOIT);
  107:C        3              END;
  108:C        3     OTHERWISE CALL(RPGISRHOOK,STATUS,DATA,DOIT);
  109:C        3     END;
  110:C        2   END;
  111:S
  112:S
  113:D        1   PROCEDURE POLLISR(WAIT:BOOLEAN);
  114:D     -2 2   VAR KBDSTATUS : STATUSTYPE;
  115:D     -6 2       ISRIBPTR  : PISRIB;
  116:C        2   BEGIN
  117:C        2     IF INTLEVEL>0 THEN
  118:C        3     BEGIN { POLL FOR INTERUPT OR JUST LEAVE }
  119:C        3       REPEAT KBDSTATUS.STATBYTE:=STATUSREG;
  120:C        4       UNTIL KBDSTATUS.READY OR NOT WAIT;
```

```
   121:C        3      IF KBDSTATUS.READY THEN
   122:C        4      BEGIN ISRIBPTR := ADDR(ISRREC); A804XISR(ISRIBPTR); END;
   123:C        3    END;
   124:C        2  END;
   125:S
   126:D        1  PROCEDURE SETUPREAD(COUNT:SHORTINT; ANYVAR BUFFER:WINDOW; OFFSET:SHORTINT);
   127:C        2  BEGIN
   128:C        2    MAXDATA:=COUNT; HAVEDATA:=0; DATABUFFER:=ADDR(BUFFER,OFFSET);
   129:C        2  END;
   130:S
   131:D        1  PROCEDURE WAIT4KBDREADY;
   132:D    -2  2  VAR KBDSTATUS : STATUSTYPE;
   133:C        2  BEGIN
   134:C        2    REPEAT KBDSTATUS.STATBYTE:=STATUSREG;
   135:C        3    UNTIL NOT KBDSTATUS.BUSY;
   136:C        2  END;
   137:S
   138:D        1  PROCEDURE SENDCMD(CMD:BYTE);
   139:C        2  BEGIN WAIT4KBDREADY; CMDREG:=CHR(CMD);
   140:C        2  END;
   141:S
   142:D        1  PROCEDURE SENDDATA(DATA:BYTE);
   143:C        2  BEGIN WAIT4KBDREADY; DATAREG:=CHR(DATA);
   144:C        2  END;
   145:S
   146:D        1  PROCEDURE CMD_READ_1(CMD:BYTE; VAR DATA:BYTE);
   147:C        2  BEGIN SETUPREAD(1,DATA,1); SENDCMD(CMD);
   148:C        2        WHILE HAVEDATA<MAXDATA DO POLLISR(TRUE);
   149:C        2  END;
   150:S
   151:D        1  PROCEDURE SENDBYTESLSF(N:SHORTINT; ANYVAR BUFFER:WINDOW; OFFSET:SHORTINT);
   152:D    -2  2  VAR I : SHORTINT;      { SEND LEAST SIGNIFICANT BYTE FIRST }
   153:C        2  BEGIN
   154:C        2    FOR I:=N-1 DOWNTO 0 DO SENDDATA(ORD(BUFFER[I+OFFSET]));
   155:C        2  END;
   156:S
   157:D        1  PROCEDURE BEEPOP(FREQUENCY,DURATION : BYTE);
   158:C        2  BEGIN
   159:C        2    SENDCMD(HEX('A3'));
   160:C        2    SENDDATA((256-DURATION) MOD 256); SENDDATA(FREQUENCY);
   161:C        2  END;
   162:S
   163:D        1  PROCEDURE MASKOPS(ENABLE,DISABLE:BYTE);
   164:C        2  BEGIN
   165:C        2    MASK:=IOR(DISABLE,IAND(MASK,-1-ENABLE));
   166:C        2    SENDCMD(HEX('40')+MASK);
   167:C        2  END;
   168:S
   169:D        1  PROCEDURE DO_RPGOPS(CMD : BYTE; VAR DATA : BYTE);
   170:C        2  BEGIN
   171:C        2    CASE CMD OF
   172:C        3    0: MASKOPS(KBDMASK,0);       { ENABLE RPG }
   173:C        3    1: MASKOPS(0,KBDMASK);       { DISABLE RPG }
   174:C        3    2: BEGIN SENDCMD(HEX('A6')); SENDDATA(DATA); END;    { SET RATE }
   175:C        3    3: BEGIN SENDCMD(HEX('26')); CMD_READ_1(HEX('17'),DATA) ; END;
   176:C        3                                 {FIX 4/12/84 SFB GET RATE}
   177:C        3    OTHERWISE
   178:C        3    END;
   179:C        2  END;
   180:S
```



```
   181:D        1  PROCEDURE DO_KBDOPS(CMD: BYTE; VAR DATA: BYTE);
   182:D        2  TYPE
   183:D        2    LANGITF=ARRAY[0..31] OF LANGTYPE;
   184:D        2    LANGNONITF=ARRAY[0..5] OF LANGTYPE;
   185:D        2  CONST
   186:D        2    WHICHITFLANG=LANGITF[NO_KBD,NO_KBD,NO_KBD,SWISS_FR_KBD,NO_KBD,NO_KBD,
   187:D        2                         NO_KBD,CDN_ENG_KBD,NO_KBD,NO_KBD,NO_KBD,ITALIAN_KBD,
   188:D        2                         NO_KBD,DUTCH_KBD,SWEDISH_KBD,GERMAN_KBD,
   189:D        2                         NO_KBD,NO_KBD,NO_KBD,SPANISH_EUR_KBD,NO_KBD,
   190:D        2                         BELGIAN_KBD,FINISH_KBD,UK_KBD,CDN_FR_KBD,
   191:D        2                         SWISS_GR_KBD,NORWEGIAN_KBD,FRENCH_KBD,DANISH_KBD,
   192:D        2                         KATAKANA_KBD,SPANISH_LATIN_KBD,US_KBD];
   193:D        2    WHICHNONITFLANG = LANGNONITF[US_KBD,FRENCH_KBD,GERMAN_KBD,SWEDISH_KBD,
   194:D        2                           SPANISH_KBD,KATAKANA_KBD];
   195:D    -2  2  VAR C:BYTE;
   196:C        2  BEGIN
   197:C        2    CASE CMD OF
   198:C        3    0: MASKOPS(KBDMASK,0);       { ENABLE KEYBOARD }
   199:C        3    1: MASKOPS(0,KBDMASK);       { DISABLE KEYBOARD }
   200:C        3    2,3: BEGIN
   201:C        3         IF CMD=2 THEN C:=HEX('A0') { SET AUTO DELAY }
   202:C        4                  ELSE C:=HEX('A2'); { SET REPEAT RATE }
   203:C        3         SENDCMD(C); SENDDATA(256-DATA);
   204:C        3        END;
   205:C        3    4,5: BEGIN
   206:C        3         IF CMD=4 THEN C:=HEX('20')  {GET DELAY}
   207:C        4                  ELSE C:=HEX('22'); {GET REPEAT}
   208:C        3         SENDCMD(C);                 { COPY DATA TO TIMER - SFB 3/20/84 }
   209:S                  CMD_READ_1(HEX('17'),DATA); DATA:=256-DATA; { READ byte from TIMER
   210:C        3                                                       SFB 3/20/84 }
   211:C        3        END;
   212:C        3    6: BEGIN    { SET_KBDTYPE }
   213:C        3         CMD_READ_1(HEX('11'),KBDCONFIG);
   214:C        3           IF ODD(KBDCONFIG DIV 32) THEN KBDTYPE:=ITFKBD
   215:C        4           ELSE
   216:C        4             IF ODD(KBDCONFIG) THEN KBDTYPE:=SMALLKBD
   217:C        5                               ELSE KBDTYPE:=LARGEKBD;
   218:C        3         DATA:=KBDCONFIG;
   219:C        3        END;
   220:C        3     7:BEGIN    { SET_KBDLANG }
   221:C        3         CMD_READ_1(HEX('12'),DATA);
   222:C        3         CASE KBDTYPE OF
   223:C        4           SMALLKBD,LARGEKBD : KBDLANG:=WHICHNONITFLANG[DATA];
   224:C        4           ITFKBD            : KBDLANG:=WHICHITFLANG[DATA MOD 32];
   225:C        4           OTHERWISE          KBDLANG:=NO_KBD;
   226:C        4         END;
   227:C        3        END;
   228:C        3    OTHERWISE
   229:C        3    END;
   230:C        2  END;
   231:S
   232:D    -4  1  PROCEDURE SEND_WAIT(L:STRING3);
   233:D    -8  2  VAR I,J : SHORTINT;    { SEND COMMANDS & WAIT FOR DATA }
   234:C        2  BEGIN
   235:C        2    FOR I:=1 TO STRLEN(L) DO
   236:C        3    BEGIN J:=HAVEDATA; SENDCMD(ORD(L[I])); WHILE J=HAVEDATA DO; END;
   237:C        2  END;
   238:S
   239:D        1  PROCEDURE CLOCKOPS(CMD:CLOCKOP; VAR THETIME:RTCTIME);
   240:C        2  BEGIN
```

```
241:C         2     IF CMD=CGET THEN
242:C         2     BEGIN {CGET} { READ THE TIME AND DATE }
243:C         3       THETIME.PACKEDTIME:=0; SETUPREAD(3,THETIME.PACKEDTIME,1);
244:C         3       SENDCMD(HEX('31')); SEND_WAIT(#21#20#19);
245:C         3       THETIME.PACKEDDATE:=0; SETUPREAD(2,THETIME.PACKEDDATE,2);
246:C         3       SEND_WAIT(#23#22);
247:C         3     END
248:C         3     ELSE
249:C         3     BEGIN {CPUT} { SET THE TIME AND DATE }
250:C         4       SENDCMD(HEX('AD')); SENDBYTESLSF(3,THETIME.PACKEDTIME,1);
251:C         4       SENDCMD(HEX('AF')); SENDBYTESLSF(2,THETIME.PACKEDDATE,2);
252:C         3     END;
253:C         2   END;
254:S
255:D         1   PROCEDURE TIMEROPS(TIMER:TIMERTYPES; OP:TIMEROPTYPE;
256:D         2                       VAR TD:TIMERDATA);
257:D    -4   2   VAR TDATA:INTEGER;
258:D    -6   2       C    : BYTE;
259:C         2   BEGIN
260:C         2     CASE OP OF
261:C         3     SETT:
262:C         3       CASE TIMER OF
263:C         4         DELAYT,CYCLICT:
264:C         4           BEGIN
265:C         4             IF TIMER=CYCLICT THEN C:=HEX('BA') ELSE C:=HEX('B7');
266:C         4             SENDCMD(C);
267:C         4             IF TD.COUNT=0 THEN SENDCMD(HEX('31')) { TO CANCEL }
268:C         5             ELSE
269:C         5             BEGIN TDATA:=16777216-TD.COUNT; SENDBYTESLSF(3,TDATA,1); END;
270:C         4           END;
271:C         4         PERIODICT:;     { DON'T DO ANY THING }
272:C         4         DELAY7T:   BEGIN
273:C         4                      SENDCMD(HEX('B2'));
274:C         4                      IF TD.COUNT=0 THEN SENDCMD(HEX('31')) { TO CANCEL }
275:C         5                      ELSE
276:C         5                      BEGIN TDATA:=65536-TD.COUNT; SENDBYTESLSF(2,TDATA,2); END;
277:C         4                    END;
278:C         4         MATCHT:    BEGIN
279:C         4                      TDATA:=(TD.MATCH.HOUR*360000)+(TD.MATCH.MINUTE*6000)+
280:C         4                             TD.MATCH.CENTISECOND;
281:C         4                      SENDCMD(HEX('B4'));
282:C         4                      IF TDATA=0 THEN SENDCMD(HEX('31')) { TO CANCEL }
283:C         5                                 ELSE SENDBYTESLSF(3,TDATA,1);
284:C         4                    END;
285:C         4         OTHERWISE
286:C         4       END; { CASE TIMER }
287:C         3     READT:
288:C         3       BEGIN
289:C         3         TDATA:=0;
290:C         3         CASE TIMER OF
291:C         4           DELAYT,CYCLICT:
292:C         4                      BEGIN
293:C         4                        SETUPREAD(3,TDATA,1);
294:C         4                        IF TIMER=CYCLICT THEN C:=HEX('3E') ELSE C:=HEX('3B');
295:C         4                        SENDCMD(C); SEND_WAIT(#21#20#19);
296:C         4                        TD.COUNT:=16777216-TDATA;
297:C         4                      END;
298:C         4           PERIODICT: TD.COUNT:=1;
299:C         4           DELAY7T:   BEGIN
300:C         4                        SETUPREAD(2,TDATA,2);
```



```
301:C         4                      SENDCMD(HEX('36')); SEND_WAIT(#20#19);
302:C         4                      TD.COUNT:= 65536-TDATA;
303:C         4                    END;
304:C         4           MATCHT:  BEGIN
305:C         4                      SETUPREAD(3,TDATA,1);
306:C         4                      SENDCMD(HEX('38')); SEND_WAIT(#21#20#19);
307:C         4                      TD.MATCH.HOUR:=TDATA DIV 360000;
308:C         4                      TD.MATCH.MINUTE:=TDATA-(TD.MATCH.HOUR*360000) DIV 6000;
309:C         4                      TD.MATCH.CENTISECOND:= TDATA MOD 6000;
310:C         4                    END;
311:C         4           OTHERWISE
312:C         4         END; { CASE TIMER }
313:C         3       END;
314:C         3     GETTINFO:
315:C         3       BEGIN
316:C         3         TD.RESOLUTION:=10000;
317:C         3         IF TIMER=DELAY7T THEN TD.RANGE:=65535
318:C         4         ELSE
319:C         4         IF TIMER=PERIODICT THEN TD.RANGE:=1
320:C         5                            ELSE TD.RANGE:=16777215;
321:C         3       END;
322:C         3     END; { CASE OP }
323:C         2   END;
324:S
325:S
326:D         1   PROCEDURE KEYTRANS(VAR STATBYTE,KEY: BYTE; VAR DOKEY: BOOLEAN);
327:D    -1   2   VAR EXTSTATE: BOOLEAN;
328:S
329:D         2     PROCEDURE SSET(VAR K:BOOLEAN);
330:C         2     BEGIN K:=TRUE; DOKEY:=TRANSMODE=KPASS_EXTC;
331:C         3           EXTSTATE:= DOKEY;
332:C         2     END;
333:D         2     PROCEDURE SCLEAR(VAR K:BOOLEAN);
334:C         3     BEGIN K:=FALSE; DOKEY:=FALSE;
335:C         3           IF TRANSMODE=KPASS_EXTC THEN EXTSTATE:= TRUE
336:C         4           ELSE EXTSTATE:= NOT (EXTLEFT OR EXTRIGHT);
337:C         3     END;
338:S
339:C         2   BEGIN {KEYTRANS}
340:C         2     STATBYTE := (STATBYTE DIV 16)*16; DOKEY:=TRUE;
341:C         2     IF KBDTYPE=ITFKBD THEN
342:C         3     BEGIN
343:C         3       IF TRANSMODE=KPASSTHRU THEN EXTSTATE:= UP
344:C         4       ELSE
345:C         4       BEGIN     { NOT PASSTHRU }
346:C         4         IF KEY=18 THEN SSET(EXTLEFT)
347:C         5         ELSE
348:C         5          IF KEY=19 THEN SSET(EXTRIGHT)
349:C         6          ELSE
350:C         6           IF KEY=146 THEN SCLEAR(EXTLEFT)
351:C         7           ELSE
352:C         7            IF KEY=147 THEN SCLEAR(EXTRIGHT)
353:C         8            ELSE
354:C         8            BEGIN
355:C         8              IF KBDSYSMODE THEN
356:C         9              CASE KEY OF
357:C        10                27: KEY:=26;  {F1=K0}
358:C        10                28: KEY:=42;  {F2=RECALL}
359:C        10                29: KEY:=51;  {F5=STEP}
360:C        10                30: KEY:=49;  {F6=ALPHA}
```

```
361:C       10              31: KEY:=50;  {F7=GRAPHICS}
362:C       10              32: KEY:=45;  {F3=CLR->END}
363:C       10              33: KEY:=58;  {F4=CONTINUE}
364:C       10              34,35: ; { UP AND DOWN ARROW KEYS }
365:C       10              36: KEY:=37;  {F8=K9}
366:C       10              OTHERWISE
367:C       10              END; { CASE KEY }
368:C        8              IF TRANSMODE=KSHIFT_EXTC THEN EXTSTATE:= NOT(EXTLEFT OR EXTRIGHT)
369:C        9                                       ELSE EXTSTATE:= UP;
370:C        8            END;
371:C        4       END;  { NOT PASSTHRU MODE }
372:C        3       STATBYTE:=STATBYTE+(ORD(EXTSTATE)*8)+7; { INCLUDE EXTCHAR STATUS }
373:C        3     END     { ITFK8D }
374:C        3     ELSE STATBYTE:=STATBYTE+15; { TURN ON ALL LOW ORDER BITS }
375:C        2   END;  {KEYTRANS}
376:S
377:D        1   FUNCTION INITA804X:BOOLEAN;
378:D        2   VAR
379:D    -2  2     TEMP : BYTE;
380:C        2   BEGIN
381:C        2     INITA804X:=FALSE;
382:C        2     TRY
383:C        3       TEMP := ORD(STATUSREG);   { IS THE 8041/8042 PRESENT ? }
384:C        3       MASK := 0;              { INITIALIZE MASK }
385:C        3       MASKOPS(0,KBDMASK+RESETMASK+TIMERMASK+PSIMASK+FHIMASK);
386:C        3       PERMISRLINK(A804XISR,ADDR(STATUSREG),1,1,1,ADDR(ISRREC));
387:C        3       HAVEDATA := 0;  MAXDATA := 0;
388:C        3       BFREQUENCY:=8;  BDURATION:=8;
389:C        3       BEEPERHOOK := BEEPOP;
390:C        3       RPGREQHOOK := DO_RPGOPS;
391:C        3       KBDREQHOOK := DO_KBDOPS;
392:C        3       KBDPOLLHOOK:= POLLISR;
393:C        3       CLOCKIOHOOK:= CLOCKOPS;
394:C        3       TIMERIOHOOK:= TIMEROPS;
395:C        3       DATAHOOK   := DATAISR;
396:C        3       DO_KBDOPS(SET_KBDTYPE,TEMP);
397:C        3       DO_KBDOPS(SET_KBDLANG,TEMP);
398:C        3       KBDSYSMODE  :=TRUE;
399:C        3       IF KBDTYPE=ITFKBD THEN SETSTATUS(6,'S');
400:C        3       TRANSMODE   := KPASSTHRU;
401:C        3       KBDALTLOCK  := FALSE; KBDCAPSLOCK:=TRUE;
402:C        3       EXTLEFT     := FALSE; EXTRIGHT := FALSE;
403:C        3       KBDTRANSHOOK:= KEYTRANS;
404:C        3       MASKOPSHOOK := MASKOPS;
405:C        3       STATUS5HOOK := DUMMYSTATUS56;
406:C        3       STATUS6HOOK := DUMMYSTATUS56;
407:C        3       INITA804X   := TRUE;
408:C        3     RECOVER
409:C        3       IF ESCAPECODE<>-12 THEN ESCAPE(ESCAPECODE);
410:C        2   END; {INITA804X}
411:S
412:C        1 END; {MODULE}
413:S
414:D        1 IMPORT A804XDVR,LOADER;
415:C        1 BEGIN
416:C        1   IF INITA804X THEN MARKUSER;
417:C        1 END. {PROGRAM}

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

Description

AMIGO is the transfer method (driver) for all supported Amigo family discs.

Usage

AMIGO supports the following discs:

- 8290x
- 9121
- 9895
- 913x

Requirements

DISCHPIB, DRVASM, IODECLARATIONS, HPIB or DISC_INTF

Optionally: DMA

Notes

AMIGO does not support the 7905, 7906, 7920 and 7925 hard discs. These discs may not be used with Series 200 computers.

```
    1:S                                                 (*
    2:S
    3:S           (c) Copyright Hewlett-Packard Company, 1983.
    4:S          All rights are reserved.  Copying or other
    5:S          reproduction of this program except for archival
    6:S          purposes is prohibited without the prior
    7:S          written consent of Hewlett-Packard Company.
    8:S
    9:S
   10:S                       RESTRICTED RIGHTS LEGEND
   11:S
   12:S          Use, duplication, or disclosure by the Government
   13:S          is subject to restrictions as set forth in
   14:S          paragraph (b) (3) (B) of the Rights in Technical
   15:S          Data and Computer Software clause in
   16:S          DAR 7-104.9(a).
   17:S
   18:S          HEWLETT-PACKARD COMPANY
   19:D        0 Fort Collins, Colorado                        *)
   20:S
   21:S
   22:D        0 $modcal$
   23:D        0 $debug off, range off, ovflcheck off$
   24:D        0 $stackcheck off, iocheck off$
   25:S
   26:D        0 $search 'DRVASM', 'DISCHPIB', 'IOLIB:KERNEL'$
   27:S
```



```
   28:D        0 $page$
   29:D        0 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'$
   30:S
   31:S
   32:D        0 program AMIGOinit;
   33:S
   34:D        1 module CSamigo; {amigo command set}
   35:S
   36:D        1 import
   37:D        1   sysglobals, bkgnd, discHPIB;
   38:S
   39:D        1 export
   40:D        1
   41:D        1   type
   42:S
   43:D        1     amigo_dev_type =  {enumerated supported amigo devices}
   44:D        1       (HP9895, HP8290X, HP913X_A, HP913X_B, HP913X_C, HP7905, HP7906, HP7920, HP7925);
   45:D        1
   46:D        1     command_type = {commands supported by the issue_cmd procedure}
   47:D        1       ( req_status,       { request status              }
   48:D        1         req_syndrome,     { request syndrome            }
   49:D        1         req_log_addr,     { request logical address     }
   50:D        1         seek_cmd,         { seek                        }
   51:D        1         addr_record_cmd,  { address record              }
   52:D        1         recalibrate_cmd,  { recalibrate                 }
   53:D        1         unbuf_read_cmd,   { unbuffered read             }
   54:D        1         verify_cmd,       { verify                      }
   55:D        1         unbuf_write_cmd,  { unbuffered write            }
   56:D        1         init_d_cmd,       { initialize, setting D bits  }
   57:D        1         format_cmd,       { format                      }
   58:D        1         buf_read_cmd,     { buffered read               }
   59:D        1         buf_write_cmd );  { buffered write              }
   60:D        1
   61:D        1     ftcb_type = {first two command bytes -  structure for most commands}
   62:D        1       packed record
   63:D        1         opcode: byte;
   64:D        1         unit: byte;
   65:D        1       end;
   66:D        1
   67:D        1     s1_type =  {enumerated status 1 values}
   68:D        1         (
   69:D        1     normal_completion                       , illegal_opcode                        ,
   70:D        1     unit_available                          , illegal_drive_type                    ,
   71:D        1     s1_4                                    , s1_5                                  ,
   72:D        1     s1_6                                    , cylinder_compare_error                ,
   73:D        1     uncorrectable_data_error                , head_sector_compare_error             ,
   74:D        1     io_program_error                        , s1_11                                 ,
   75:D        1     end_of_cylinder                         , sync_bit_not_received_in_time         ,
   76:D        1     overrun                                 , possibly_correctable_data_error       ,
   77:D        1     illegal_access_to_spare_track           , defective_track                       ,
   78:D        1     access_not_ready_during_data_operation  , status_2_error                        ,
   79:D        1     s1_20                                   , s1_21                                 ,
   80:D        1     attempt_to_write_on_protected_track     , unit_unavailable                      ,
   81:D        1     s1_24                                   , s1_25                                 ,
   82:D        1     s1_26                                   , s1_27                                 ,
   83:D        1     s1_28                                   , s1_29                                 ,
   84:D        1     s1_30                                   , drive_attention
   85:D        1         );
```

```
    86:D        1 $page$
    87:D        1
    88:D        1     tva_type = {three vector address}
    89:D        1       packed record
    90:D        1         cyl: shortint;    { cylinder address }
    91:D        1         head: byte;       { head address     }
    92:D        1         sect: byte;       { sector address   }
    93:D        1       end;
    94:D        1
    95:D        1     status_type = {4 bytes of returned status}
    96:D        1       packed record
    97:D        1         { stat 1 - from previous operation }
    98:D        1         s: boolean;       { 15       spare track bit        }
    99:D        1         p: boolean;       { 14       protected track bit    }
   100:D        1         d: boolean;       { 13       defective track bit    }
   101:D        1         s1: s1_type;      { 12-8     last operation status  }
   102:D        1         unit: byte;       { 7-0      unit number            }
   103:D        1           { stat 2 - from specified drive }
   104:D        1         star: boolean;    { 15       conditions *'ed below  }
   105:D        1         xx: 0..3;         { 14-13    undefined              }
   106:D        1         tttt: 0..15;      { 12-9     disc type bits         }
   107:D        1         r: boolean;       { 8        reserved               }
   108:D        1         a: boolean;       { 7        drive attention        }
   109:D        1         w: boolean;       { 6        write protected        }
   110:D        1         fmt: boolean;     { 5        format switch          }
   111:D        1         e: boolean;       { 4        *drive fault           }
   112:D        1         f: boolean;       { 3        first status bit       }
   113:D        1         c: boolean;       { 2        *seek check            }
   114:D        1         ss: 0..3;         { 1,0      *drive ready status    }
   115:D        1       end;
   116:D        1
   117:D        1     syndrome_type = {14 bytes of returned syndrome information}
   118:D        1       packed record
   119:D        1         sb_pad1: 0..7;
   120:D        1         sb_s1: s1_type;
   121:D        1         sb_pad2: byte;
   122:D        1         sb_tva: tva_type;
   123:D        1         sb_offset: shortint;
   124:D        1         sb_correction_bytes: packed array[0..5] of char;
   125:D        1       end;
   126:D        1
   127:D        1     map_type = {media addressing parameters}
   128:D        1       record
   129:D        1         cyl_per_med: shortint;  { number of cylinders per medium  }
   130:D        1         trk_per_cyl: shortint;  { number of tracks per cylinder   }
   131:D        1         sec_per_trk: shortint;  { number of sectors per track     }
   132:D        1       end;
   133:D        1
   134:D        1     unsigned16 = 0..65535;
```



```
   135:D        1   $page$
   136:D        1
   137:D        1   function  device             (uep: uep_type): amigo_dev_type;
   138:D        1   function  MI_controller      (uep: uep_type): boolean;
   139:D        1   function  surface_mode       (uep: uep_type): boolean;
   140:D        1   procedure get_map            (uep: uep_type; var map: map_type);
   141:D        1   function  records_per_medium (uep: uep_type): integer;
   142:D        1   function  decoded_addr       (uep: uep_type; tva: tva_type): integer;
   143:D        1
   144:D        1   procedure issue_cmd          (uep: uep_type; command: command_type; var cmd_buffer: ftcb_type);
   145:D        1   function  dsj                (uep: uep_type): byte;
   146:D        1   procedure set_file_mask      (uep: uep_type);
   147:D        1   procedure recalibrate        (uep: uep_type);
   148:D        1   procedure status             (uep: uep_type; var status_bytes: status_type);
   149:D        1   procedure syndrome           (uep: uep_type; var syndrome_bytes: syndrome_type);
   150:D        1   procedure seek               (uep: uep_type; record_addr: integer);
   151:D        1   procedure addr_record        (uep: uep_type; record_addr: integer);
   152:D        1   function  logical_addr       (uep: uep_type): integer;
   153:S
   154:D        1 implement {CSamigo}
   155:S
   156:D        1 var
   157:D    -4  1   most_recent_status: status_type;  {for post-mortem diagnostic purposes only!!!}
   158:S
   159:S
   160:D        1 function device(uep: uep_type): amigo_dev_type;
   161:C        2   begin {device}
   162:C        2     case uep^.letter of
   163:C        3       'H':        device := HP9895;
   164:C        3       'N':        device := HP8290X;
   165:C        3       'U':        device := HP913X_A;
   166:C        3       'V':        device := HP913X_B;
   167:C        3       'W':        device := HP913X_C;
   168:C        3       'Y':        device := HP7905;
   169:C        3       'C':        device := HP7906;
   170:C        3       'P':        device := HP7920;
   171:C        3       'X':        device := HP7925;
   172:C        3       otherwise  ioresc_bkgnd(uep, znodevice);
   173:C        3     end {case}
   174:C        3   end; {device}
   175:S
   176:S
   177:D        1 function MI_controller(uep: uep_type): boolean;
   178:C        2   begin {MI_controller}
   179:C        2     MI_controller := device(uep) in [HP7905, HP7906, HP7920, HP7925];
   180:C        2   end; {MI_controller}
   181:D    -4  1
   182:S
   183:D        1 function surface_mode(uep: uep_type): boolean;
   184:C        2   begin {surface_mode}
   185:C        2     surface_mode := device(uep) in [HP7905, HP7906];
   186:C        2   end; {surface_mode}
```

```
187:D     -4  1 $page$
188:S
189:D         1 procedure get_map(uep: uep_type; var map: map_type);
190:D         2   type
191:D         2     device_maps_type = array[HP8290X..HP7925] of map_type;
192:D         2   const
193:D         2     DS9895_map = map_type[ cyl_per_med:  75,  trk_per_cyl: 2,  sec_per_trk: 30];
194:D         2     SS9895_map = map_type[ cyl_per_med:  73,  trk_per_cyl: 1,  sec_per_trk: 30];
195:D         2     device_maps = device_maps_type
196:D         2      [(HP8290X)  map_type[ cyl_per_med:  33,  trk_per_cyl: 2,  sec_per_trk: 16],
197:D         2       (HP913X_A) map_type[ cyl_per_med: 152,  trk_per_cyl: 4,  sec_per_trk: 31],
198:D         2       (HP913X_B) map_type[ cyl_per_med: 305,  trk_per_cyl: 4,  sec_per_trk: 31],
199:D         2       (HP913X_C) map_type[ cyl_per_med: 305,  trk_per_cyl: 6,  sec_per_trk: 31],
200:D         2       (HP7905)   map_type[ cyl_per_med: 400,  trk_per_cyl: 2,  sec_per_trk: 48],
201:D         2       (HP7906)   map_type[ cyl_per_med: 400,  trk_per_cyl: 2,  sec_per_trk: 48],
202:D         2       (HP7920)   map_type[ cyl_per_med: 800,  trk_per_cyl: 5,  sec_per_trk: 48],
203:D         2       (HP7925)   map_type[ cyl_per_med: 800,  trk_per_cyl: 9,  sec_per_trk: 64]  ];
204:D         2   var
205:D     -2  2     this_device: amigo_dev_type;
206:C         2   begin {get_map}
207:C         2     this_device := device(uep);
208:C         2     if this_device=HP9895 then {use single/double-sided flag set by status routine}
209:C         3       case uep^.devid of
210:C         4         1:  map := SS9895_map;
211:C         4         2:  map := DS9895_map;
212:C         4         otherwise ioresc_bkgnd(uep, zcatchall);
213:C         4       end {case}
214:C         4     else
215:C         3       map := device_maps[this_device];
216:C         2 end; {get_map}
217:S
218:S
219:D         1 function records_per_medium(uep: uep_type): integer;
220:D         2   var
221:D     -6  2     map: map_type;
222:C         2   begin {records_per_medium}
223:C         2     get_map(uep, map);
224:C         2     with map do
225:C         3       records_per_medium := sec_per_trk*trk_per_cyl*cyl_per_med;
226:C         2   end; {records_per_medium}
227:S
228:S
229:D         1 function dsj(uep: uep_type): byte;
230:D         2   var
231:D     -2  2     dsj_byte: packed record  b: byte;  end;
232:D     -2  2   const
233:D     -2  2     dsj_sec = 16;
234:C         2   begin {dsj}
235:C         2     HPIBshort_msge_in(uep, dsj_sec, addr(dsj_byte), sizeof(dsj_byte));
236:C         2     dsj := dsj_byte.b;
237:C         2   end; {dsj}
```



```
238:D     -4  1 $page$
239:D     -4  1
240:D         1 procedure issue_cmd(uep: uep_type; command: command_type; var cmd_buffer: ftcb_type);
241:D         2   type
242:D         2     ctet = {command table entry type}
243:D         2       packed record
244:D         2         sec: shortint; { secondary command     }
245:D         2         oc: byte;      { opcode                }
246:D         2         nb: byte;      { number of data bytes  }
247:D         2       end;
248:D         2     command_table_type = packed array[command_type] of ctet;
249:D         2   const
250:D         2     command_table = command_table_type
251:D         2       [ (req_status     )  ctet[sec:  8, oc: 03, nb: 2],
252:D         2         (req_syndrome   )  ctet[sec:  8, oc: 13, nb: 2],
253:D         2         (req_log_addr   )  ctet[sec:  8, oc: 20, nb: 2],
254:D         2         (seek_cmd       )  ctet[sec:  8, oc: 02, nb: 6],
255:D         2         (addr_record_cmd)  ctet[sec:  8, oc: 12, nb: 6],
256:D         2         (recalibrate_cmd)  ctet[sec:  8, oc: 01, nb: 2],
257:D         2         (unbuf_read_cmd )  ctet[sec:  8, oc: 05, nb: 2],
258:D         2         (verify_cmd     )  ctet[sec:  8, oc: 07, nb: 4],
259:D         2         (unbuf_write_cmd)  ctet[sec:  8, oc: 08, nb: 2],
260:D         2         (init_d_cmd     )  ctet[sec:  8, oc: 43, nb: 2],
261:D         2         (format_cmd     )  ctet[sec: 12, oc: 24, nb: 5],
262:D         2         (buf_read_cmd   )  ctet[sec: 10, oc: 05, nb: 2],
263:D         2         (buf_write_cmd  )  ctet[sec:  9, oc: 08, nb: 2]  ];
264:C         2   begin {issue_cmd}
265:C         2     with cmd_buffer, command_table[command] do
266:C         3       begin
267:C         3         opcode := oc;
268:C         3         unit := uep^.du;
269:C         3         HPIBshort_msge_out(uep, sec, addr(cmd_buffer), nb);
270:C         3       end; {with}
271:C         2   end; {issue_cmd}
272:S
273:S
274:D         1 procedure set_file_mask(uep: uep_type);
275:D         2   type
276:D         2     sfm_cmd_type =  {set file mask command}
277:D         2       packed record
278:D         2         oc: byte;
279:D         2         mask: byte;
280:D         2       end;
281:D         2     sfm_cmd_array_type = array[boolean] of sfm_cmd_type;
282:D         2   const
283:D         2     sfm_sec = 8;        {secondary}
284:D         2     sfm_oc = 15;        {op code}
285:D         2     sfm_cmd_array = sfm_cmd_array_type
286:D         2       [ {false: cylinder mode} sfm_cmd_type[ oc: sfm_oc, mask: 7 ],
287:D         2         {true:  surface mode } sfm_cmd_type[ oc: sfm_oc, mask: 5 ]  ];
288:D         2   var
289:D     -2  2     sfm_cmd: sfm_cmd_type;
290:C         2   begin {set_file_mask}
291:C         2     sfm_cmd := sfm_cmd_array[surface_mode(uep)];
292:C         2     HPIBshort_msge_out(uep, sfm_sec, addr(sfm_cmd), sizeof(sfm_cmd));
293:C         2   end; {set_file_mask}
```

```
294:D    -4  1 $page$
295:S
296:D        1 procedure recalibrate(uep: uep_type);
297:D        2   var
298:D    -2  2     recalibrate_cmd_buf: ftcb_type;
299:C        2   begin {recalibrate}
300:C        2     issue_cmd(uep, recalibrate_cmd, recalibrate_cmd_buf);
301:C        2   end; {recalibrate}
302:D    -4  1
303:S
304:D        1 procedure status(uep: uep_type; var status_bytes: status_type);
305:D        2   var
306:D    -2  2     status_cmd_buf: ftcb_type;
307:D    -2  2   const
308:D    -2  2     send_sts_sec = 8;
309:C        2   begin {status}
310:C        2     issue_cmd(uep, req_status, status_cmd_buf);
311:C        2     if not MI_controller(uep) then HPIBwait_for_ppol(uep);
312:C        2     HPIBshort_msge_in(uep, send_sts_sec, addr(status_bytes), sizeof(status_bytes));
313:C        2     most_recent_status := status_bytes;  {for post-mortem diagnostic purposes only!}
314:C        2     with uep^ do
315:C        3       case device(uep) of
316:C        4         HP9895:  {use the otherwise undefined devid field to indicate...}
317:C        4           if status_bytes.tttt in [5,6]
318:C        5             then devid := 2         {double-sided disc}
319:C        5             else devid := 1;        {single-sided disc}
320:C        4         HP8290X: {use the otherwise undefined devid field to indicate...}
321:C        4           devid := ord(status_bytes.r);  {Sparrow (1) versus Chinook (0)}
322:C        4         otherwise
323:C        4           {do nothing};
324:C        4       end; {case}
325:C        2   end; {status}
326:S
327:D    -4  1
328:D        1 procedure syndrome(uep: uep_type; var syndrome_bytes: syndrome_type);
329:D        2   var
330:D    -2  2     syndrome_cmd_buf: ftcb_type;
331:D    -2  2   const
332:D    -2  2     send_syn_sec = 8;
333:C        2   begin {syndrome}
334:C        2     issue_cmd(uep, req_syndrome, syndrome_cmd_buf);
335:C        2     HPIBshort_msge_in(uep, send_syn_sec, addr(syndrome_bytes), sizeof(syndrome_bytes));
336:C        2   end; {syndrome}
```



```
337:D    -4  1 $page$
338:D    -4  1
339:D        1 function coded_addr(uep: uep_type; record_addr: integer): tva_type;
340:D        2   var
341:D    -6  2     map: map_type;
342:D   -10  2     track: integer;
343:C        2   begin {coded_addr}
344:C        2     get_map(uep, map);
345:C        2     with map do
346:C        3       begin
347:C        3         coded_addr.sect  := record_addr mod sec_per_trk;
348:C        3         track            := record_addr div sec_per_trk;
349:C        3         if surface_mode(uep) then
350:C        4           begin  {select proper 7905/06 logical "volume"}
351:C        4             coded_addr.head := track div cyl_per_med + 2*uep^.dv;
352:C        4             coded_addr.cyl  := track mod cyl_per_med;
353:C        4           end {then}
354:C        4         else
355:C        4           begin
356:C        4             coded_addr.head := track mod trk_per_cyl;
357:C        4             coded_addr.cyl  := track div trk_per_cyl;
358:C        4           end; {else}
359:C        3       end; {with}
360:C        2   end; {coded_addr}
361:S
362:S
363:D        1 function decoded_addr(uep: uep_type; tva: tva_type): integer;
364:D        2   var
365:D    -6  2     map: map_type;
366:D   -10  2     track: integer;
367:C        2   begin {decoded_addr}
368:C        2     get_map(uep, map);
369:C        2     with tva, map do
370:C        3       begin
371:C        3         if surface_mode(uep)
372:C        4           then track := (head-2*uep^.dv)*cyl_per_med+cyl
373:C        4           else track := cyl*trk_per_cyl+head;
374:C        3         decoded_addr := track*sec_per_trk+sect;
375:C        3       end; {with}
376:C        2   end; {decoded_addr}
```

```
377:D    -4  1 $page$
378:S
379:D        1 procedure seek(uep: uep_type; record_addr: integer);
380:D        2   var
381:D        2     seek_cmd_buf:
382:D        2       packed record
383:D        2         ftcb: ftcb_type;
384:D        2         tva: tva_type;
385:D    -6  2       end;
386:C        2   begin {seek}
387:C        2     seek_cmd_buf.tva := coded_addr(uep, record_addr);
388:C        2     issue_cmd(uep, seek_cmd, seek_cmd_buf.ftcb);
389:C        2   end; {seek}
390:S
391:S
392:D        1 procedure addr_record(uep: uep_type; record_addr: integer);
393:D        2   var
394:D        2     addr_record_cmd_buf:
395:D        2       packed record
396:D        2         ftcb: ftcb_type;
397:D        2         tva: tva_type;
398:D    -6  2       end;
399:C        2   begin {addr_record}
400:C        2     addr_record_cmd_buf.tva := coded_addr(uep, record_addr);
401:C        2     issue_cmd(uep, addr_record_cmd, addr_record_cmd_buf.ftcb);
402:C        2   end; {addr_record}
403:S
404:S
405:D        1 function logical_addr(uep: uep_type): integer;
406:D        2   var
407:D    -2  2     ladd_cmd_buf: ftcb_type;
408:D    -6  2     tva: tva_type;
409:D    -6  2   const
410:D    -6  2     send_addr_sec = 8;
411:C        2   begin {logical_addr}
412:C        2     issue_cmd(uep, req_log_addr, ladd_cmd_buf);
413:C        2     if not MI_controller(uep) then HPIBwait_for_ppol(uep);
414:C        2     HPIBshort_msge_in(uep, send_addr_sec, addr(tva), sizeof(tva));
415:C        2     logical_addr := decoded_addr(uep, tva);
416:C        2   end; {logical_addr}
417:S
418:S
419:C        1 end; {CSamigo}
```



```
420:D        1 $page$
421:S
422:D        1 module amigodvr;
423:S
424:D        1 import
425:D        1   sysglobals, drvasm, bkgnd, discHPIB, CSamigo;
426:S
427:D        1 export
428:D        1   procedure get_letter(uep: uep_type; ident: shortint; var letter: char);
429:D        1
430:D        1   procedure amigoio(fp: fibp; request: amrequesttype;
431:D        2                     anyvar buffer: window; length, position: integer);
432:S
433:D        1 implement {amigodvr}
434:S
435:S
436:S
437:S       procedure used by CTABLE for self-configuring
438:D        1 }
439:D        1 procedure get_letter(uep: uep_type; ident: shortint; var letter: char);
440:D        2   const
441:D        2     ident_table_entries = 7;
442:D        2   var
443:D    -2  2     index: shortint;
444:D    -6  2     status_bytes: status_type;
445:D    -6  2   type
446:D    -6  2     device_table_type = array[0..3] of char;
447:D    -6  2     itet = {ident_table_entry_type}
448:D    -6  2       record
449:D    -6  2         ident: shortint;
450:D    -6  2         letter: char;
451:D    -6  2       end;
452:D    -6  2     ident_table_type = array[1..ident_table_entries] of itet;
453:D    -6  2   const
454:D    -6  2     device_table = device_table_type
455:D    -6  2       [ {0} 'C', {1} 'P', {2} 'V', {3} 'X' ];
456:D    -6  2     ident_table = ident_table_type
457:D    -6  2       [ {HP9895 }   itet[ ident: 0*256+129 ($0081), letter: 'H' ],
458:D    -6  2         {HP8290X}   itet[ ident: 1*256+  4 ($0104), letter: 'N' ],
459:D    -6  2         {HP913X_A}  itet[ ident: 1*256+  6 ($0106), letter: 'U' ],
460:D    -6  2         {HP913X_B}  itet[ ident: 1*256+ 10 ($010A), letter: 'V' ],
461:D    -6  2         {HP913X_C}  itet[ ident: 1*256+ 15 ($010F), letter: 'W' ],
462:D    -6  2         {MAC}       itet[ ident: 0*256+  2 ($0002), letter: 'X' ],
463:D    -6  2         {IDC}       itet[ ident: 0*256+  3 ($0003), letter: 'X' ] ];
464:C        2   begin {get_letter}
465:S
466:C        2     letter := chr(255); {initially undefined}
467:C        2     for index := 1 to ident_table_entries do
468:C        3       if ident=ident_table[index].ident then
469:C        4         letter := ident_table[index].letter;
470:C        2     if letter=chr(255) then ioresc_bkgnd(uep, znodevice);
471:S
472:C        2     uep^.letter := letter;  {for determining ppol wait in status routine}
473:C        2     if dsj(uep)<>0 then {don't worry about it};
474:C        2     HPIBamigo_clear(uep);
475:C        2     HPIBwait_for_ppol(uep);
476:C        2     status(uep, status_bytes);
477:C        2     if dsj(uep)<>0 then ioresc_bkgnd(uep, zcatchall);
478:C        2     if status_bytes.ss=2 then ioresc_bkgnd(uep, znodevice);  {"unit not present or power off"}
479:S
```

```
480:C        2     if letter='X' then  {determine which 7906 family member it really is}
481:C        3       begin
482:C        3         if not (status_bytes.tttt in [0..3]) then
483:C        4           ioresc_bkgnd(uep, znodevice); {unrecognized unit type}
484:C        3         letter := device_table[status_bytes.tttt];
485:C        3       end; {if}
486:S
487:C        2   end; {get_letter}
488:S
489:S
490:D        1 procedure clear_unit(uep:uep_type);
491:D        2   type
492:D        2     device_ident_type = array[HP9895..HP913X_C] of shortint;
493:D        2     device_table_type = array[0..3] of amigo_dev_type;
494:D        2   const
495:D        2     device_ident = device_ident_type
496:D        2       [ {HP9895 }   0*256+129,  {$0081}
497:D        2         {HP8290X}   1*256+  4,  {$0104}
498:D        2         {HP913X_A}  1*256+  6,  {$0106}
499:D        2         {HP913X_B}  1*256+ 10,  {$010A}
500:D        2         {HP913X_C}  1*256+ 15   {$010F} ];
501:D        2     MAC_ident = 0*256+2;  {$0002}
502:D        2     IDC_ident = 0*256+3;  {$0003}
503:D        2     device_table = device_table_type
504:D        2       [ {0} HP7906, {1} HP7920, {2} HP7905, {3} HP7925 ];
505:D        2   var
506:D    -2  2     dev: amigo_dev_type;
507:D    -4  2     ident: shortint;
508:D    -6  2     dummy_dsj: byte;
509:D   -10  2     status_bytes: status_type;
510:C        2   begin {clear_unit}
511:C        2     dev := device(uep);
512:C        2     ident := HPIBamigo_identify(uep);
513:C        2     if MI_controller(uep) then  {check for MAC or IDC controller}
514:C        3       begin
515:C        3         if not ((ident=MAC_ident) or ((ident=IDC_ident) and Simon_DMA(uep))) then
516:C        4           ioresc_bkgnd(uep, znodevice);
517:C        3       end {then}
518:C        3     else  {require EXACT device/ident match}
519:C        3       if ident<>device_ident[dev] then ioresc_bkgnd(uep, znodevice);
520:C        2     if dev=HP8290X then {avoid the amigo clear; it takes too much time!}
521:C        3       dummy_dsj := dsj(uep)  {just remove the power-on holdoff}
522:C        3     else  {go ahead and do the hard clear}
523:C        3       begin
524:C        3         HPIBamigo_clear(uep);
525:C        3         HPIBwait_for_ppol(uep);
526:C        3       end; {else}
527:C        2     status(uep, status_bytes);
528:C        2     if dsj(uep)<>0 then ioresc_bkgnd(uep, zcatchall);
529:C        2     if status_bytes.ss=2 then ioresc_bkgnd(uep, znodevice);     {"unit not present or power off"}
530:C        2     if MI_controller(uep) then  {we need to check the exact type of THIS particular unit}
531:C        3       begin
532:C        3         if not (status_bytes.tttt in [0..3]) then
533:C        4           ioresc_bkgnd(uep, znodevice); {unrecognized unit type}
534:C        3         if dev<>device_table[status_bytes.tttt] then
535:C        4           ioresc_bkgnd(uep, znodevice); {wrong unit type}
536:C        3       end; {then}
537:C        2   end; {clear_unit}
```



```
538:D        1 $page$
539:S
540:S          {
541:S            procedures in the background transfer chain
542:D        1 }
543:D        1 procedure initial_seek          (uep: uep_type);  forward;
544:D        1 procedure enter_transfer_chain  (uep: anyptr);    forward;
545:D        1 procedure issue_transfer_request (uep: uep_type); forward;
546:D        1 procedure initiate_data_transfer (uep: anyptr);   forward;
547:D        1 procedure upon_data_transfer_comp(uep: anyptr);   forward;
548:D        1 procedure check_dsj             (uep: anyptr);    forward;
549:S
550:S
551:S          {
552:S            main driver procedure
553:D        1 }
554:D        1 procedure amigoio;
555:D        2
556:D        2   var
557:D    -4  2     uep: uep_type;
558:D    -6  2     ident: shortint;
559:D    -7  2     asynchronous: boolean;
560:D    -7  2
561:C        2   begin {amigoio}
562:C        2
563:C        2     uep := addr(unitable^[fp^.funit]);
564:C        2     asynchronous := (request=startread) or (request=startwrite);
565:C        2
566:C        2     if uep^.offline then
567:C        3       ioresult := ord(znodevice)
568:C        3     else
569:C        3       try
570:C        4
571:C        4         ioresult := ord(inoerror);
572:C        4
573:C        4         case request of
574:C        5
575:C        5           clearunit:
576:C        5             begin
577:C        5               uep^.umediavalid := false;
578:C        5               unit_wait(uep);
579:C        5               ioresult := ord(inoerror);  {forget any previous error}
580:C        5               allocate_bkgnd_info(uep);
581:C        5               HPIBcheck_sc(uep);
582:C        5               clear_unit(uep);
583:C        5               deallocate_bkgnd_info(uep);
584:C        5             end;
585:C        5
586:C        5           unitstatus:
587:C        5             fp^.fbusy := unit_busy(uep);
588:C        5
589:C        5           flush:
590:C        5             {do nothing};
591:S
592:C        5           readbytes, writebytes, startread, startwrite:
593:C        5             begin {transfer operations}
594:C        5               unit_wait(uep);
595:C        5               ioresult := ord(inoerror);  {forget any previous error}
596:C        5               allocate_bkgnd_info(uep);
597:S
```

```
598:C        5               with bip_type(uep^.dvrtemp)^ do
599:C        6                 begin
600:C        6                   feot := fp^.feot;       {end of transfer procedure}
601:C        6                   fibptr := fp;           {parmeter to the eot procedure}
602:C        6                   async := asynchronous;  {determines whether or not to call eot proc}
603:S
604:C        6                   if Simon_no_dma(uep) then
605:C        7                     ioresc_bkgnd(uep, zbaddma);
606:C        6                   if uep^.ureportchange and not uep^.umediavalid then
607:C        7                     ioresc_bkgnd(uep, zmediumchanged);
608:C        6                   if position mod 256<>0 then
609:C        7                     ioresc_bkgnd(uep, zbadmode);
610:C        6                   if (position<0) or (length<0) or (position+length>fp^.fpeof) then
611:C        7                     ioresc_bkgnd(uep, ieof);
612:S
613:C        6                   HPIBcheck_sc(uep);
614:C        6                   ident := HPIBamigo_identify(uep);  {confirm device present}
615:C        6                   if dsj(uep)<>0 then {do nothing};  {remove power-on holdoff if any}
616:S
617:C        6                   if length=0 then
618:C        7                     deallocate_bkgnd_info(uep)  {nothing to transfer}
619:C        7                   else
620:C        7                     begin
621:C        7                       read_operation      :=  (request=readbytes) or (request=startread);
622:C        7                       xfr_chain_semaphore :=  false;
623:C        7                       bx_tries            :=  0;
624:C        7                       bx_strt_rcrd        :=  (position+fp^.fileid+uep^.byteoffset) div 256;
625:C        7                       bx_bufptr           :=  addr(buffer);
626:C        7                       bx_length           :=  length;
627:C        7                       initial_seek(uep);  {initiate the transfer}
628:C        7                     end; {else}
629:C        6                 end; {with}
630:S
631:C        5               if not asynchronous then
632:C        6                 begin
633:C        6                   unit_wait(uep);
634:C        6                   uep^.dvrtemp := ord(inoerror);  {report synchronous errors only once}
635:C        6                 end; {if}
636:C        5             end; {transfer operations}
637:C        5
638:C        5           otherwise  {unrecognized request}
639:C        5             ioresult := ord(ibadrequest);
640:C        5
641:C        5         end; {cases}
642:C        4
643:C        4       recover
644:C        4         begin
645:C        4           abort_bkgnd_process(uep);
646:C        4           ioresult := uep^.dvrtemp;
647:C        4           if not asynchronous then
648:C        5             uep^.dvrtemp := ord(inoerror);  {report synchronous errors only once}
649:C        4         end; {recover}
650:C        2
651:C        2   end; {amigoio}
```



```
652:D        1 $page$
653:S
654:D        1 procedure initial_seek(uep: uep_type);
655:D        2   var
656:D    -4  2     status_bytes: status_type;
657:C        2   begin {initial_seek}
658:C        2
659:C        2     if device(uep)=HP9895 then  {read status to determine single or double-sided}
660:C        3       begin
661:C        3         status(uep, status_bytes);
662:C        3         with status_bytes do  {specifically disallow non HP-formatted discs}
663:C        4           begin
664:C        4             if f then
665:C        5               begin
666:C        5                 uep^.umediavalid := false;
667:C        5                 if uep^.ureportchange then
668:C        6                   ioresc_bkgnd(uep, zmediumchanged);
669:C        5               end; {if}
670:C        4             if (ss=0) and not (tttt in [2,6]) then
671:C        5               ioresc_bkgnd(uep, zuninitialized)
672:C        5           end; {with}
673:C        3       end; {if}
674:C        2
675:C        2     with bip_type(uep^.dvrtemp)^ do
676:C        3       begin
677:C        3
678:C        3         if MI_controller(uep) then
679:C        4           begin
680:C        4             set_file_mask(uep);
681:C        4             buffered_transfer := not Simon_DMA(uep);
682:C        4             HPIBwait_for_ppol(uep);  {shouldn't take very long!}
683:C        4           end {then}
684:C        4         else
685:C        4           if device(uep)=HP8290X
686:C        5             then buffered_transfer := uep^.devid=0  {Sparrow (1) versus Chinook (0)}
687:C        5             else buffered_transfer := false;
688:C        3
689:C        3         if device(uep) in [HP913X_A..HP913X_C]
690:C        4           then addr_record(uep, bx_strt_rcrd)   {allows greater overlapping with 9914}
691:C        4           else seek       (uep, bx_strt_rcrd);
692:C        3
693:C        3         HPIBupon_ppol_resp(uep, enter_transfer_chain);
694:C        3
695:C        3       end; {with}
696:C        2
697:C        2   end; {initial_seek}
698:S
699:S
700:D        1 procedure enter_transfer_chain(uep: anyptr);
701:C        2   begin {enter_transfer_chain}
702:C        2     with bip_type(uep_type(uep)^.dvrtemp)^ do
703:C        3       if not test_and_toggle(xfr_chain_semaphore) then
704:C        4         repeat
705:C        5           issue_transfer_request(uep);
706:C        5         until test_and_toggle(xfr_chain_semaphore);
707:C        2   end; {enter_transfer_chain}
```

```
708:D        1 $page$
709:S
710:D        1 procedure issue_transfer_request(uep: uep_type);
711:D        2   const
712:D        2     sect_per_surf = 400*48;  {only valid for 7905/06!!!}
713:D        2   var
714:D     -2 2     transfer_command: command_type;
715:D     -4 2     transfer_cmd_buf: ftcb_type;
716:D     -8 2     max_tfr_length: integer;
717:D    -12 2     remaining_surf_bytes: integer;
718:D    -13 2     wait_for_ppol: boolean;
719:C        2   begin {issue_transfer_request}
720:C        2     with bip_type(uep^.dvrtemp)^ do
721:C        3       try
722:C        4         if buffered_transfer then
723:C        5           begin
724:C        5             if read_operation
725:C        6               then transfer_command := buf_read_cmd
726:C        6               else transfer_command := buf_write_cmd;
727:C        5             max_tfr_length := 256;
728:C        5           end {then}
729:C        5         else
730:C        5           begin
731:C        5             if read_operation
732:C        6               then transfer_command := unbuf_read_cmd
733:C        6               else transfer_command := unbuf_write_cmd;
734:C        5             if MI_controller(uep) then
735:C        6               begin
736:C        6                 max_tfr_length := 65536;  {max DMA burst length}
737:C        6                 if surface_mode(uep) then {don't try to cross a surface boundary}
738:C        7                   begin
739:C        7                     remaining_surf_bytes := (sect_per_surf-bx_strt_rcrd mod sect_per_surf)*256;
740:C        7                     if remaining_surf_bytes<max_tfr_length then
741:C        8                       max_tfr_length := remaining_surf_bytes;
742:C        7                   end; {then}
743:C        6               end {then}
744:C        6             else
745:C        6               max_tfr_length := maxint;
746:C        5           end; {else}
747:C        4
748:C        4         if bx_length<=max_tfr_length
749:C        5           then bx_tfr_length := bx_length
750:C        5           else bx_tfr_length := max_tfr_length;
751:C        4
752:C        4         wait_for_ppol := buffered_transfer or (device(uep)=HP8290X);
753:C        4         issue_cmd(uep, transfer_command, transfer_cmd_buf);
754:C        4         if wait_for_ppol  {computed above because of critical MAC/IDC timing!}
755:C        5           then HPIBupon_ppol_resp(uep, initiate_data_transfer)
756:C        5           else initiate_data_transfer(uep);
757:C        4       recover
758:C        4         abort_bkgnd_process(uep);
759:C        2   end; {issue_transfer_request}
```



```
760:D        1 $page$
761:S
762:D        1 procedure initiate_data_transfer(uep: anyptr);
763:D        2   const
764:D        2     tfr_data_sec = 0;
765:C        2   begin {initiate_data_transfer}
766:C        2     with bip_type(uep_type(uep)^.dvrtemp)^ do
767:C        3       try
768:C        4         HPIBupon_dxfr_comp(uep, tfr_data_sec, bx_bufptr, bx_tfr_length, upon_data_transfer_comp);
769:C        4       recover
770:C        4         abort_bkgnd_process(uep);
771:C        2   end; {initiate_data_transfer}
772:S
773:S
774:D        1 procedure upon_data_transfer_comp(uep: anyptr);
775:C        2   begin {upon_data_transfer_comp}
776:C        2     try
777:C        3       if bip_type(uep_type(uep)^.dvrtemp)^.iores<>inoerror then escape(-10);
778:C        3       HPIBupon_ppol_resp(uep, check_dsj);
779:C        3     recover
780:C        3       abort_bkgnd_process(uep);
781:C        2   end; {upon_data_transfer_comp}
782:S
783:D        1
784:D        1 procedure check_dsj(uep: anyptr);
785:D        2   var
786:D     -1 2     transfer_successful: boolean;
787:D     -1 2   const
788:D     -1 2     maxtries = 10;
789:D     -1 2
790:D        2   procedure process_errors(uep: uep_type);
791:D        3     var
792:D     -4 3       status_bytes: status_type;
793:D    -18 3       syndrome_bytes: syndrome_type;
794:D    -22 3       cb_ptr: charptr;
795:D    -24 3       cb_index: shortint;
796:D    -28 3       e_rcrd: integer;
797:D    -32 3       possible_bytes_transferred: integer;
798:C        3     begin {process_errors}
799:C        3       with bip_type(uep^.dvrtemp)^ do
800:C        4         begin
801:C        4           status(uep, status_bytes);
802:C        4           if dsj(uep)<>0 then ioresc_bkgnd(uep, zcatchall);
803:C        4           with status_bytes do
804:C        5             case s1 of
805:C        6               {retryable errors}
806:C        6                 cylinder_compare_error,
807:C        6                 uncorrectable_data_error,
808:C        6                 head_sector_compare_error,
809:C        6                 end_of_cylinder,
810:C        6                 sync_bit_not_received_in_time,
811:C        6                 overrun,
812:C        6                 possibly_correctable_data_error,
813:C        6                 illegal_access_to_spare_track,
814:C        6                 defective_track,
815:C        6                 access_not_ready_during_data_operation:
816:C        6                   begin  {retryable errors case}
817:C        6                     if s1=overrun
818:C        7                       then e_rcrd := bx_strt_rcrd  {addr untrustworthy after overrun}
819:C        7                       else e_rcrd := logical_addr(uep);
```

```
820:C        6
821:C        6                    if s1=cylinder_compare_error then
822:C        7                      if MI_controller(uep) then  {recalibrate & retry}
823:C        8                        begin
824:C        8                          recalibrate(uep);
825:C        8                          HPIBwait_for_ppol(uep); {shouldn't happen very often!}
826:C        8                        end {then}
827:C        9                      else if e_rcrd=bx_strt_rcrd then  {don't retry...}
828:C        9                        ioresc_bkgnd(uep, znoblock);  {Chinook takes 25 secs/retry!!!}
829:C        6
830:C        6                    if e_rcrd<=bx_strt_rcrd then  {careful with MAC/IDC verify address!}
831:C        7                      begin
832:C        7                        bx_tries := bx_tries+1;
833:C        7                        transfer_successful := false;  {unless correctable below}
834:C        7                        if (s1=possibly_correctable_data_error) and (bx_tries>5) then
835:C        8                          with syndrome_bytes do
836:C        9                            begin
837:C        9                              syndrome(uep, syndrome_bytes);
838:C        9                              if (sb_s1=possibly_correctable_data_error)  and
839:C       10                                 (decoded_addr(uep, sb_tva)=e_rcrd)       and
840:C       10                                 (sb_offset>=0)                           and
841:C       10                                 (sb_offset<=125)                         then {it's correctable!}
842:C       10                                begin
843:C       10                                  cb_ptr := addr(bx_bufptr^,2*sb_offset);
844:C       10                                  cb_index := 0;
845:C       10                                  while (cb_index<6) and
846:C       11                                        (integer(cb_ptr)<integer(bx_bufptr)+bx_tfr_length) do
847:C       11                                    begin
848:C       11                                      eor(sb_correction_bytes[cb_index], cb_ptr);
849:C       11                                      cb_ptr := addr(cb_ptr^, 1);
850:C       11                                      cb_index := cb_index+1;
851:C       11                                    end; {while}
852:C       10                                  e_rcrd := e_rcrd+1;  {this record has been corrected!}
853:C       10                                  bx_tries := 0;  {no attempts made on the next record yet}
854:C       10                                  transfer_successful := true;  {at least partially!}
855:C       10                                end; {then}
856:C        9                            end; {with}
857:C        7                      end {then}
858:C        7                    else
859:C        7                      begin
860:C        7                        bx_tries := 1;  {first attempt on this record}
861:C        7                        transfer_successful := true;  {at least partially!}
862:C        7                      end; {else}
863:C        6
864:C        6                    if transfer_successful then
865:C        7                      begin
866:C        7                        possible_bytes_transferred := (e_rcrd-bx_strt_rcrd)*256;
867:C        7                        if bx_tfr_length>possible_bytes_transferred then
868:C        8                          bx_tfr_length := possible_bytes_transferred;
869:C        7                      end {then}
870:C        7                    else
871:C        7                      if bx_tries>=maxtries then
872:C        8                        if s1 in [uncorrectable_data_error, possibly_correctable_data_error]
873:C        9                          then ioresc_bkgnd(uep, zbadblock)
874:C        9                          else ioresc_bkgnd(uep, znoblock);
875:C        6                  end;  {retryable errors case}
876:C        6
877:C        6                {immediate escape errors}
878:C        6                  illegal_drive_type,
879:C        6                  unit_unavailable:
```



```
880:C        6                    ioresc_bkgnd(uep, znodevice);
881:C        6                  attempt_to_write_on_protected_track:
882:C        6                    ioresc_bkgnd(uep, zprotected);
883:C        6
884:C        6                {errors requiring status 2 processing}
885:C        6                  status_2_error:
886:C        6                    begin
887:C        6                      if f then
888:C        7                        uep^.umediavalid := false;
889:C        6                      if e then ioresc_bkgnd(uep, zbadhardware);
890:C        6                      case ss of
891:C        7                        1: ioresc_bkgnd(uep, znotready);
892:C        7                        2: ioresc_bkgnd(uep, znodevice);
893:C        7                        3: ioresc_bkgnd(uep, znomedium);
894:C        7                        otherwise {test further conditions below};
895:C        7                      end; {case}
896:C        6                      if not read_operation and w then
897:C        7                        ioresc_bkgnd(uep, zprotected);
898:C        6                      if not MI_controller(uep) and not (tttt in [2,6]) then
899:C        7                        ioresc_bkgnd(uep, zuninitialized);
900:C        6                      if f then
901:C        7                        begin
902:C        7                          if uep^.ureportchange then
903:C        8                            ioresc_bkgnd(uep, zmediumchanged);
904:C        7                          bx_tries := bx_tries+1;
905:C        7                          if bx_tries>1 then
906:C        8                            ioresc_bkgnd(uep, zcatchall);
907:C        7                          transfer_successful := false;
908:C        7                        end {then}
909:C        7                      else
910:C        7                        ioresc_bkgnd(uep, zcatchall);
911:C        6                    end;
912:C        6                  drive_attention:
913:C        6                    begin
914:C        6                      if e then ioresc_bkgnd(uep, zbadhardware);
915:C        6                      if c then
916:C        7                        begin
917:C        7                          e_rcrd := logical_addr(uep);
918:C        7                          if e_rcrd>(bx_strt_rcrd+(bx_tfr_length-1)div 256) then
919:C        8                            transfer_successful := true  {already transferred enough bytes}
920:C        8                          else
921:C        8                            if e_rcrd>=records_per_medium(uep)
922:C        9                              then ioresc_bkgnd(uep, znosuchblk)
923:C        9                              else ioresc_bkgnd(uep, znoblock);
924:C        7                        end {then}
925:C        7                      else
926:C        7                        ioresc_bkgnd(uep, zcatchall);
927:C        6                    end;
928:C        6
929:C        6                {other errors}
930:C        6                  otherwise
931:C        6                    ioresc_bkgnd(uep, zcatchall);
932:C        6                end; {case}
933:C        4              end; {with}
934:C        3          end; {process_errors}
```

```
   935:D    -1  2 $page$
   936:D    -1  2
   937:C        2   begin {check_dsj}
   938:C        2     with bip_type(uep_type(uep)^.dvrtemp)^ do
   939:C        3       try
   940:C        4
   941:C        4         if dsj(uep)=0 then
   942:C        5           if bdx_pre_eoi
   943:C        6             then ioresc_bkgnd(uep, zcatchall)  {unresolved premature eoi!}
   944:C        6             else transfer_successful := true
   945:C        6         else
   946:C        5           process_errors(uep);  {will set/clear transfer_successful, or escape}
   947:C        4
   948:C        4         if transfer_successful then
   949:C        5           begin
   950:C        5             bx_strt_rcrd := bx_strt_rcrd+bx_tfr_length div 256;
   951:C        5             bx_bufptr := addr(bx_bufptr^, bx_tfr_length);
   952:C        5             bx_length := bx_length-bx_tfr_length;
   953:C        5           end; {then}
   954:C        4
   955:C        4         if bx_length>0 then
   956:C        5           if MI_controller(uep) or not transfer_successful then
   957:C        6             begin
   958:C        6               if device(uep) in [HP913X_A..HP913X_C, HP7905..HP7925]
   959:C        7                 then addr_record(uep, bx_strt_rcrd)
   960:C        7                 else seek       (uep, bx_strt_rcrd);
   961:C        6               HPIBupon_ppol_resp(uep, enter_transfer_chain);
   962:C        6             end {then}
   963:C        6           else
   964:C        6             enter_transfer_chain(uep)
   965:C        6         else
   966:C        5           deallocate_bkgnd_info(uep);
   967:C        4
   968:C        4       recover
   969:C        4         abort_bkgnd_process(uep);
   970:C        2   end; {check_dsj}
   971:S
   972:S
   973:C        1 end; {amigodvr}
   974:S
   975:S
   976:S
   977:D        1 { program AMIGOinit }
   978:S
   979:D        1 import
   980:D        1   loader;
   981:S
   982:C        1 begin {AMIGOinit}
   983:C        1   markuser;
   984:C        1 end. {AMIGOinit}
   985:S
   986:C        1

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

ASCII

Description

ASCII contains the access method (AM) used to read and write text to a file of type ASC. ASCII also contains the routine necessary to install itself in the system.

Usage

ASCII is used by the file system to read and write .ASC files.

Requirements

SYSGLOBALS and ASM.

Notes

ASCII can do "direct access" reads on record boundaries. This is required by the Compiler to allow INCLUDE files.

```
     1:D        0 $SYSPROG$ $DEBUG OFF, RANGE OFF, OVFLCHECK OFF, STACKCHECK OFF$
     2:D        0 program installascii;
     3:D        1 module asciimodule;
     4:D        1 import sysglobals, asm;
     5:D        1 export
     6:D        1   procedure asciiam(fp : fibp; request: amrequesttype;
     7:D        2                         anyvar buffer:window; bufsize,position:integer);
     8:D        1 implement
     9:D        1 const
    10:D        1   buflength = fblksize;
    11:D        1   sectorsize= 256;
    12:S
    13:D        1 procedure asciiam(fp: fibp; request: amrequesttype;
    14:D        2                       anyvar buffer: window; bufsize,position: integer);
    15:D        2 var
    16:D    -4  2   bufend : integer;
    17:S
    18:D        2   procedure bufparams;
    19:C        3   begin
    20:C        3     with fp^ do
    21:C        4     if (flastpos+buflength)>fpeof then bufend := fpeof
    22:C        5                                   else bufend := flastpos + buflength;
    23:C        3   end;
    24:S
    25:D        2   procedure rendfile;
    26:C        3   begin
    27:C        3     if request<>readtoeol then ioresult := ord(ieof);
    28:C        3     fp^.fpos := fp^.fleof;        { fix fpos }
    29:C        3     escape(0);
    30:C        3   end;
    31:S
    32:D        2   procedure flushbuffer;
    33:C        3   begin
    34:C        3     ioresult := 0;
    35:C        3     with fp^, unitable^[funit] do
    36:C        4     begin     { write out the buffer }
    37:C        4       call(tm,fp,WRITEBYTES,fbuffer,bufend-flastpos,flastpos);
    38:C        4       if ioresult<>0 then escape(0);
    39:C        4       fbufchanged := false;
    40:C        4     end;
    41:C        3   end;
    42:S
    43:D        2   procedure loadbuffer(posit: integer);
    44:C        3   begin
    45:C        3     ioresult:=0;
    46:C        3     with fp^, unitable^[funit]  do
    47:C        4     begin
    48:C        4       if fbufchanged then flushbuffer;
    49:C        4       flastpos := (posit div sectorsize) * sectorsize; bufparams;
    50:C        4       call(tm,fp,READBYTES,fbuffer,bufend-flastpos,flastpos);
    51:C        4       if ioresult<>0 then escape(0);
    52:C        4     end;
    53:C        3   end;
    54:S
    55:D        2   procedure seekposit(posit: integer);
    56:C        3   begin
    57:C        3     with fp^ do
    58:C        4     begin
    59:C        4       if (posit<flastpos) or (posit>=bufend) then loadbuffer(posit);
    60:C        4       fpos:=posit;
```



```
    61:C        4     end;
    62:C        3   end;
    63:S
    64:D        2   procedure wendbuffer;
    65:C        3   begin { append access only }
    66:C        3     with fp^ do
    67:C        4     begin
    68:C        4         { write all but last sector }
    69:C        4       fbufchanged := true;
    70:C        4       if (bufend-flastpos)>sectorsize then
    71:C        5       begin
    72:C        5         bufend := bufend - sectorsize;
    73:C        5         flushbuffer;
    74:C        5         moveleft(fbuffer[bufend-flastpos],fbuffer[0],sectorsize);
    75:C        5       end
    76:C        5       else flushbuffer;
    77:C        4       fleof := fpos + 1;            fmodified := true;
    78:C        4       if fleof>fpeof then
    79:C        5       begin   { move the physical end of file }
    80:C        5         fpos := ((fleof + 256) div 256) * 256;
    81:C        5         call(unitable^[funit].dam,fp^,funit,STRETCHIT);
    82:C        5         if fleof>fpeof then begin ioresult := ord(ieof); escape(0); end;
    83:C        5         fpos := fleof - 1;
    84:C        5       end;
    85:C        4       flastpos := bufend; bufparams;
    86:C        4     end;
    87:C        3   end;
    88:S
    89:D        2   procedure wnextbyte(c:char);
    90:C        3   begin { append access only }
    91:C        3     with fp^ do
    92:C        4     begin
    93:C        4       if fpos>=bufend then wendbuffer;
    94:C        4       fbuffer[fpos - flastpos] := c;     fbufchanged := true;
    95:C        4       fpos := fpos + 1;  fleof := fpos;  fmodified   := true;
    96:C        4     end;
    97:C        3   end;      { wnextbyte }
    98:S
    99:D        2   procedure writeendline;
   100:D        3   var
   101:D    -8  3     tposit, j : integer;
   102:C        3   begin
   103:C        3     with fp^ do
   104:C        4     begin
   105:C        4       if freptcnt=0 then
   106:C        5       begin     { zero length record }
   107:C        5         if odd(fpos) then wnextbyte(' ');
   108:C        5         wnextbyte(chr(0)); wnextbyte(chr(0));
   109:C        5       end
   110:C        5       else
   111:C        5       begin     { have some data }
   112:C        5         tposit := fpos; j := tposit - freptcnt - 2;
   113:C        5                 { rewrite the record size }
   114:C        5         if j<flastpos then loadbuffer(j);
   115:C        5         fbuffer[j - flastpos] := chr(freptcnt div 256);
   116:C        5         fbuffer[(j+1) - flastpos] := chr(freptcnt mod 256);
   117:C        5         fbufchanged := true;
   118:S
   119:C        5         if tposit>=bufend then loadbuffer(tposit);
   120:C        5         fpos := tposit;   freptcnt := 0;
```

```
   121:C        5      end;
   122:C        4    end;
   123:C        3  end; { writeendline }
   124:S
   125:D        2  procedure wendfile;
   126:C        3  begin
   127:C        3    with fp^ do
   128:C        4    begin
   129:C        4      if freptcnt>0 then writeendline;
   130:C        4      { write logical end of file marker }
   131:C        4      if odd(fpos) then  wnextbyte(' '); { pad to even position }
   132:C        4      if fleof<fpeof then
   133:C        5      begin
   134:C        5        wnextbyte(chr(255)); wnextbyte(chr(255));
   135:C        5      end;
   136:C        4      if fbufchanged then flushbuffer;
   137:C        4      call(unitable^[funit].tm,fp,flush,ioresult,0,0);
   138:C        4    end;
   139:C        3  end;
   140:S
   141:D        2  function min(v1,v2,v3:integer):integer;
   142:C        3  begin
   143:C        3    if v1<v2 then
   144:C        4    begin       { v1 or v3 }
   145:C        4      if v1<v3 then min := v1 else min := v3;
   146:C        4    end
   147:C        4    else        { v2 or v3 }
   148:C        4    begin
   149:C        4      if v2<v3 then min := v2 else min := v3;
   150:C        4    end;
   151:C        3  end;
   152:S
   153:D        2  function rnextbyte:char;
   154:C        3  begin
   155:C        3    with fp^ do
   156:C        4    begin
   157:C        4      if fpos>=fleof then rendfile;
   158:C        4      if (fpos>=bufend) then loadbuffer(fpos);
   159:C        4      rnextbyte := fbuffer[fpos - flastpos];
   160:C        4      fpos      := fpos + 1;
   161:C        4    end;
   162:C        3  end;
   163:S
   164:D        2  procedure getrecsize;
   165:C        3  begin
   166:C        3    with fp^ do
   167:C        4    begin
   168:C        4      if odd(fpos) then freptcnt := ord(rnextbyte);
   169:C        4      freptcnt := ord(rnextbyte);
   170:C        4      if freptcnt>127 then rendfile;
   171:C        4      freptcnt := (freptcnt * 256) + ord(rnextbyte);
   172:C        4    end;
   173:C        3  end;
   174:S
   175:D        2  procedure readchars;
   176:D        3  var
   177:D    -8  3    count, i     : integer;
   178:C        3  begin       { readchars }
   179:C        3    with fp^ do
   180:C        4    begin
```



```
   181:C        4      i:=0;
   182:C        4      if bufsize=1 then
   183:C        5      begin   { single character read }
   184:C        5        if freptcnt=0 then
   185:C        6        begin buffer[0] := ' '; freptcnt := -1; end
   186:C        6        else
   187:C        6        begin buffer[0] := rnextbyte; freptcnt := freptcnt - 1; end;
   188:C        5      end
   189:C        5      else    { multi character read }
   190:C        5      while i<bufsize do
   191:C        6      begin
   192:C        6        if freptcnt=0 then
   193:C        7        begin   { end of record }
   194:C        7          buffer[i] := ' ';   i := i + 1;
   195:C        7          if i<bufsize then getrecsize
   196:C        8                       else freptcnt := -1;
   197:C        7        end
   198:C        7        else
   199:C        7        begin   { move data bytes }
   200:C        7          if fpos>=fleof then rendfile
   201:C        8                         else seekposit(fpos);
   202:C        7          count := min(bufsize-i, bufend-fpos, freptcnt);
   203:C        7          moveleft(fbuffer[fpos-flastpos],buffer[i],count);
   204:C        7          i := i + count; freptcnt := freptcnt - count; fpos := fpos + count;
   205:C        7        end;
   206:C        6      end;       { while }
   207:C        4      feoln := (freptcnt<0);
   208:C        4      if not feoln then fpos := -fpos;
   209:C        4    end;
   210:C        3  end;         { readchars }
   211:S
   212:D        2  procedure readstring;
   213:D    -8  3  var   i, count : integer;
   214:C        3  begin
   215:C        3    i := 0;
   216:C        3    with fp^ do
   217:C        4    begin
   218:C        4      if freptcnt>0 then
   219:C        5      while i<bufsize do
   220:C        6      begin       { read data bytes }
   221:C        6        if fpos>=fleof then rendfile    else seekposit(fpos);
   222:C        6        count := min(bufsize-i, buferd-fpos, freptcnt);
   223:C        6        moveleft(fbuffer[fpos-flastpos],buffer[i+1],count);
   224:C        6        freptcnt := freptcnt - count; i := i + count; fpos := fpos + count;
   225:C        6        buffer[0] := chr(i);
   226:C        6        if freptcnt=0 then i := bufsize
   227:C        7      end;
   228:C        4      fpos := -fpos;
   229:C        4    end;
   230:C        3  end;
   231:S
   232:D        2  procedure writechars;
   233:D        3  var
   234:D    -8  3    i, count  : integer;
   235:S
   236:C        3  begin       { writechars }
   237:C        3    with fp^ do
   238:C        4    begin
   239:C        4      if (freptcnt = 0) and (bufsize>0) then
   240:C        5      begin          { start a new record }
```

```
241:C  5         if odd(fpos) then wnextbyte(' ');        { pad to even size }
242:C  5         wnextbyte(chr(255)); wnextbyte(chr(255)); { dummy count field }
243:C  5       end;
244:C  4       i:=0;
245:C  4       while i<bufsize do
246:C  5       begin          { write data character(s) }
247:C  5         if fpos>=bufend then wendbuffer;
248:C  5         count := min(bufsize-i, bufend-fpos, 32767-freptcnt);
249:C  5         if count<=0 then { too many characters for the record }
250:C  6           begin ioresult := ord(ibadformat); escape(0); end;
251:C  5         moveleft(buffer[i],fbuffer[fpos-flastpos],count);
252:C  5         fpos := fpos + count; freptcnt := freptcnt + count; i := i + count;
253:C  5         fbufchanged := true;
254:C  5       end;    { while }
255:C  4       fleof := fpos;
256:C  4     end;      { with }
257:C  3   end;        { write chars }
258:S
259:C  2   begin { asciiam }
260:C  2     ioresult:=0;
261:C  2     try
262:C  3       with fp^ do
263:C  4       begin
264:C  4         if flastpos<0 then      { force buffer load }
265:C  5           begin flastpos := -buflength; fbufchanged := false; end;
266:C  4         bufparams;  fpos := abs(position);
267:C  4         case request of
268:C  5           readbytes, readtoeol:
269:C  5             begin
270:C  5               if request=readtoeol then buffer[0] := chr(0);
271:C  5               if fbufchanged then
272:C  6               begin     { close last record }
273:C  6                 fpos := fleof; wendfile;
274:C  6                 fpos := abs(position);  { restore fpos }
275:C  6               end;
276:C  5               if position<0 then
277:C  6               begin     { seqential read }
278:C  6                 if request=readbytes then readchars
279:C  7                                      else readstring;
280:C  6               end       { seqential read }
281:C  6               else
282:C  6               begin     { positioned read }
283:C  6                 if position>fp^.fleof then rendfile
284:C  7                                       else seekposit(position);
285:C  6                 getrecsize;
286:C  6                 if request=readbytes then readchars
287:C  7                                      else readstring;
288:C  6               end;      { positioned read }
289:C  5             end;
290:C  5           writebytes:
291:C  5             begin
292:C  5               if position<>0 then writechars    { normal write }
293:C  6               else
294:C  6               begin       { rewrite }
295:C  6                 flastpos := 0; bufparams; freptcnt := 0; writechars;
296:C  6               end;
297:C  5             end;
298:C  5           flush: if fbufchanged then
299:C  6                    begin fpos := fleof; wendfile; end;
300:C  5           writeeol:     { end the line }
```



```
301:C  5             begin
302:C  5               if position<>0 then writeendline
303:C  6               else
304:C  6               begin     { zero length record at start of file }
305:C  6                 flastpos := 0; bufparams; freptcnt := 0;
306:C  6                 wnextbyte(chr(0)); wnextbyte(chr(0));
307:C  6               end;
308:C  5             end;
309:C  5           otherwise ioresult := ord(ibadrequest);
310:C  5         end;    { case }
311:C  4       end;      { with }
312:C  3     recover
313:C  3       if escapecode<0 then escape(escapecode);
314:C  2   end;  { asciiam }
315:C  1 end;    { ascii module }
316:S
317:D  1 import asciimodule,sysglobals,loader;
318:C  1 begin   { installascii }
319:C  1   amtable^[ASCIIFILE]     := asciiam;
320:C  1   suffixtable^[ASCIIFILE] := 'ASC';
321:C  1   efttable^[ASCIIFILE]    := 1;
322:C  1   markuser;
323:C  1 end. { rev 18 A }
```

No errors. No warnings.
***** Nonstandard language features enabled *****

Description

BAT provides the initialization for the battery backup (power fail) option.

Usage

BAT can read the date and time from the battery option, also set the timeout for power fail.

Requirements

SYSGLOBALS and SYSDEVS.

Notes

Also see module SYSDEVS.

```
Pascal [Rev 3.0M  6/ 4/84] BAT.TEXT                26-Dec-84 21:09:44 Page 1   *** File name: BAT ***

     1:S                                                        (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1983.
     4:S          All rights are reserved.  Copying or other
     5:S          reproduction of this program except for archival
     6:S          purposes is prohibited without the prior
     7:S          written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                        RESTRICTED RIGHTS LEGEND
    11:S
    12:S          Use, duplication, or disclosure by the Government
    13:S          is subject to restrictions as set forth in
    14:S          paragraph (b) (3) (B) of the Rights in Technical
    15:S          Data and Computer Software clause in
    16:S          DAR 7-104.9(a).
    17:S
    18:S          HEWLETT-PACKARD COMPANY
    19:D        0 Fort Collins, Colorado                        *)
    20:S
    21:S
    22:D        0 $modcal$
    23:D        0 $heap_dispose off$
    24:D        0 $iocheck off$
    25:D        0 $range off$ $ovflcheck off$
    26:D        0 $debug off$
    27:D        0 $STACKCHECK OFF$
    28:D        0 $SEARCH  'INITLOAD','ASM','INIT','SYSDEVS'$
    29:S
    30:D        0 program initbat(OUTPUT);
    31:S
    32:D        1 module bat;
    33:D        1 import sysglobals, sysdevs;
    34:D        1 export
    35:D        1     procedure batinit;
    36:S
    37:D        1 implement
    38:D        1 type
    39:D        1     statustype = packed record
    40:D        1                  case integer of
    41:D        1                  0:(pad1  :0..63;
    42:D        1                     busy  :boolean;
    43:D        1                     ready :boolean);
    44:D        1                  1:(statbyte :byte);
    45:D        1                END;
    46:D        1
    47:D        1 var bat8041statusreg[4554785 ( 458001 ) ]: char;
    48:D        1     bat8041cmdreg   [4554785 ( 458001 ) ]: char;
    49:D        1     bat8041datareg  [4554753 ( 458021 ) ]: char;
    50:S
    51:D        1 procedure wait4batready;
    52:D    -2  2 var batstatus: statustype;
    53:C        2 begin
    54:C        2   repeat
    55:C        3     batstatus.statbyte:=ord(bat8041statusreg);
    56:C        3   until not batstatus.busy;
    57:C        2 end;
    58:S
    59:D        1 procedure wait4batreadready;
    60:D    -2  2 var batstatus: statustype;
```

```
Pascal [Rev 3.0M  6/ 4/84] BAT.TEXT                26-Dec-84 21:09:44 Page 2   *** File name: BAT ***

    61:C        2 begin
    62:C        2   repeat
    63:C        3     batstatus.statbyte:=ord(bat8041statusreg);
    64:C        3   until batstatus.ready;
    65:C        2 end;
    66:S
    67:D        1 procedure dobatcommand(cmd: byte; numdata: integer; b1, b2, b3, b4, b5: byte);
    68:D        2
    69:D        2   procedure batdataout(d: byte);
    70:C        3   begin wait4batready; bat8041datareg := chr(d); end;
    71:D        2
    72:C        2 begin
    73:C        2   if batterypresent then
    74:C        3   begin
    75:C        3     wait4batready;
    76:C        3     bat8041cmdreg := chr(cmd);
    77:C        3     if numdata >= 1 then batdataout(b1);
    78:C        3     if numdata >= 2 then batdataout(b2);
    79:C        3     if numdata >= 3 then batdataout(b3);
    80:C        3     if numdata >= 4 then batdataout(b4);
    81:C        3     if numdata >= 5 then batdataout(b5);
    82:C        3   end;
    83:C        2 end;
    84:S
    85:D        1 procedure batreadbyte(var data: byte);
    86:C        2 begin
    87:C        2   wait4batreadready; data := ord(bat8041datareg);
    88:C        2 end;
    89:S
    90:D        1 procedure batinit;
    91:C        2 begin
    92:C        2   batcommand(167,2,23,112,0,0,0); {set power fail to 60 seconds}
    93:C        2   batcmdhook := dobatcommand;
    94:C        2   batreadhook:= batreadbyte;
    95:C        2 end;
    96:S
    97:C        1 end;
    98:S
    99:D        1 import bat, loader, sysdevs;
   100:S
   101:C        1 begin
   102:C        1   if batterypresent THEN
   103:C        2   begin batinit; markuser; end;
   104:C        1 end.
   105:S

No errors. No warnings.
          ***** Nonstandard language features enabled *****
```

BUBBLES

Description

BUBBLES is the TM for bubble memory mass storage.

Requirements

BUBDVR, SYSGLOBALS, IODECLARATIONS, and ISR.

```
     1:S                                                (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1983.
     4:S           All rights are reserved.  Copying or other
     5:S           reproduction of this program except for archival
     6:S           purposes is prohibited without the prior
     7:S           written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                       RESTRICTED RIGHTS LEGEND
    11:S
    12:S           Use, duplication, or disclosure by the Government
    13:S           is subject to restrictions as set forth in
    14:S           paragraph (b) (3) (B) of the Rights in Technical
    15:S           Data and Computer Software clause in
    16:S           DAR 7-104.9(a).
    17:S
    18:S           HEWLETT-PACKARD COMPANY
    19:D         0 Fort Collins, Colorado                          *)
    20:S
    21:D         0 $SYSPROG$
    22:D         0 $DEBUG OFF$$RANGE OFF$
    23:D         0 program bubbles;
    24:D         1   module bubble;
    25:D         1   $SEARCH 'BUB_DVR'$
    26:D         1   import bub_dvr,sysglobals,iodeclarations,isr;
    27:D         1   export
    28:S
    29:D         1     procedure bub_tm(fp:fibp; request:amrequesttype;
    30:D         2                      anyvar buffer : window; bufsize, position : integer);
    31:D         1
    32:D         1     procedure bub_isr(isrib : pisrib);
    33:S
    34:D         1     procedure bub_init;
    35:S
    36:D         1   IMPLEMENT
    37:D         1     procedure xlate_errors(error : berrortype);
    38:C         2     begin
    39:C         2       case error of
    40:C         3         bnoerror :   ioresult := ord(inoerror);
    41:C         3         btimeout :   ioresult := ord(zbadhardware);
    42:C         3         bopfailed :  ioresult := ord(zbadhardware);
    43:C         3         bbadinterupt:ioresult := ord(zstrangei);
    44:C         3         bbadsector : ioresult := ord(znosuchblk);
    45:C         3         bbadcount :  ioresult := ord(zbadmode);
    46:C         3         bnotbubble : ioresult := ord(znodevice);
    47:C         3         bbaddata :   ioresult := ord(zbadblock);
    48:C         3         biofail :    ioresult := ord(ztimeout);
    49:C         3       otherwise   ioresult := ord(zcatchall);
    50:C         3       end;
    51:C         2     end;
    52:S
    53:D         1     procedure bub_tm(fp:fibp; request:amrequesttype;
    54:D         2                      anyvar buffer : window; bufsize, position : integer);
    55:D         2     var
    56:D     -4  2       card : anyptr;
    57:D     -8  2       info : infoptr;
    58:S
    59:C         2     begin
    60:C         2       ioresult := ord(inoerror);
```



```
    61:C         2       with fp^ , unitable^[funit] do
    62:C         3       begin
    63:C         3         card := isc_table[sc].card_ptr;
    64:C         3         if isc_table[sc].io_tmp_ptr=NIL then ioresult := ord(znodevice)
    65:C         4         else
    66:C         4         begin
    67:C         4           info := addr(isc_table[sc].io_tmp_ptr^.drv_misc);
    68:C         4           with info^ do
    69:C         5           case request of
    70:C         6             startread,
    71:C         6             readbytes:    if runstate<>b_idle then ioresult := ord(znotready)
    72:C         7                           else
    73:C         7                           begin
    74:C         7                             if (position+bufsize)<=fpeof then
    75:C         8                             begin          { is inside the file }
    76:C         8                               bbuffer.brec := addr(buffer);
    77:C         8                               bstart := fileid + byteoffset + position;
    78:C         8                               bcount := bufsize;
    79:C         8                               bretry := 3;
    80:C         8                               if bcount<>0 then bubdoread(card,info);
    81:C         8                               if request=readbytes then
    82:C         9                                  while runstate<>b_idle do; { wait for idle }
    83:C         8                               if runstate=b_idle then xlate_errors(errorcode);
    84:C         8                             end
    85:C         8                             else ioresult := ord(ieof);
    86:C         7                           end;
    87:C         6             startwrite,
    88:C         6             writebytes:   if runstate<>b_idle then ioresult := ord(znotready)
    89:C         7                           else
    90:C         7                           begin
    91:C         7                             if (position+bufsize)<=fpeof then
    92:C         8                             begin
    93:C         8                               bbuffer.brec := addr(buffer);
    94:C         8                               bstart := fileid + byteoffset + position;
    95:C         8                               if bufsize>0
    96:C         9                               then bcount := (bufsize + 255) DIV 256 * 256
    97:C         9                               else bcount := bufsize;
    98:C         8                               bretry := 3;
    99:C         8                               if bcount<>0 then bubdowrite(card,info);
   100:C         8                               if request=writebytes then
   101:C         9                                  while runstate<>b_idle do;   { wait for idle }
   102:C         8                               if runstate=b_idle then xlate_errors(errorcode);
   103:C         8                             end
   104:C         8                             else ioresult := ord(ieof);
   105:C         7                           end;
   106:C         6             flush:;   { NO_OP }
   107:C         6             clearunit:    begin
   108:C         6                             bubdoreset(card,info);
   109:C         6                             xlate_errors(info^.errorcode);
   110:C         6                             if ioresult=ord(inoerror)
   111:C         7                             then umaxbytes := maxbytes      { set device size }
   112:C         7                             else umaxbytes := 0;
   113:C         6                           end;
   114:C         6             unitstatus:   begin
   115:C         6                             fbusy := runstate<>b_idle;
   116:C         6                             if not fbusy then xlate_errors(errorcode);
   117:C         6                           end;
   118:C         6             otherwise  ioresult := ord(ibadrequest);
   119:C         6           end;  { case }
   120:C         4         end;    { if nounit }
```

```
   121:C        3       end;      { with }
   122:C        2     end; { bub_tm }
   123:D        1 $DEBUG OFF$
   124:D        1     procedure bub_isr(isrib : pisrib);
   125:D        2     var
   126:D    -4  2       temps : pio_tmp_ptr;
   127:D    -8  2       info  : infoptr;
   128:C        2     begin
   129:C        2       temps := addr(isrib^);
   130:C        2       info  := addr(temps^.drv_misc);
   131:C        2       bubdoisr(temps^.card_addr,info);
   132:C        2     end; { bub_isr }
   133:S
   134:D        1     procedure bub_init;
   135:D        2     const
   136:D        2       intreg = 3;       { interupt reg offset }
   137:D        2     type
   138:D        2       cardrec = packed record
   139:D        2                   pad : byte;
   140:D        2                   id  : byte;
   141:D        2                 end;
   142:D        2     var
   143:D    -4  2       sc        : integer;
   144:D    -8  2       isrinfo   : pisrib;
   145:D   -12  2       info      : infoptr;
   146:D   -16  2       card      : ^cardrec;
   147:C        2     begin
   148:C        2       { scan selectcode table for bubble devices }
   149:C        2       for sc:= iominisc to iomaxisc do
   150:C        3       with isc_table[sc] do
   151:C        4       begin
   152:C        4         { fix isc_table to work for system 2.0, 2.1 or 2.2 }
   153:C        4         if ((card_type=1) and (card_id=0)) or
   154:C        5            ((card_type=8) and (card_id=30)) then
   155:C        5         begin
   156:C        5           card    := card_ptr;
   157:C        5           if card^.id=30 then
   158:C        6           begin
   159:C        6             card_type := 8; { bubble memory card }
   160:C        6             card_id   := 30;
   161:C        6             isrinfo := addr(io_tmp_ptr^);
   162:C        6             info    := addr(io_tmp_ptr^.drv_misc);
   163:C        6             bubgetinfo(card,info);
   164:C        6             if info^.errorcode=bnoerror then
   165:C        7             begin
   166:C        7               permisrlink(bub_isr,         { ISR PROC }
   167:C        7                       addr(card^.intreg), { INTERUPT REG ADDRESS }
   168:C        7                       hex('C0'),hex('C0'),{ MASK and VALUE }
   169:C        7                       info^.priority,     { INTERUPT PRIORITY }
   170:C        7                       isrinfo);           { ISR INFO POINTER  }
   171:C        7             end;
   172:C        6           end;
   173:C        5         end;
   174:C        4       end;
   175:C        2     end; {bub_init}
   176:D        1
   177:C        1 end; { bub_ops }
   178:D        1 {       bubbles installation program    }
   179:D        1 import bubble,loader;
   180:S
```



```
   181:C        1 begin
   182:C        1   bub_init;
   183:C        1   markuser;
   184:C        1 end.

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

C_HOOK

Description

C_HOOK initializes the graphics CRT state variables for the Series 200 Model 236C color computer and provides Model 236C dump graphics procedure.

Requirements

SYSDEVS and SYSGLOBALS.

```
     1:D        0 {                                                                                 }
     2:D        0 { HP 9836C graphics hooks, and workstation init module                            }
     3:D        0 {                                                                                 }
     4:D        0 { Module    = C_HOOK                                                              }
     5:D        0 { Programer = BJS                                                                 }
     6:D        0 { Date      = 2-7-83                                                              }
     7:D        0 {                                                                                 }
     8:D        0 { Purpose: To initialize the graphics crt state variables for the HP 9836C.      }
     9:S
    10:D        0 { Rev history                                                                     }
    11:D        0 {  Created  - 02-07-83                                                            }
    12:D        0 {  Modified - 01-18-84 jws                                                        }
    13:S
    14:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    15:S                All rights are reserved.  Copying or other
    16:S                reproduction of this program except for archival
    17:S                purposes is prohibited without the prior
    18:S                written consent of Hewlett-Packard Company.
    19:S
    20:S
    21:S                            RESTRICTED RIGHTS LEGEND
    22:S
    23:S                Use, duplication, or disclosure by the Government
    24:S                is subject to restrictions as set forth in
    25:S                paragraph (b) (3) (B) of the Rights in Technical
    26:S                Data and Computer Software clause in
    27:S                DAR 7-104.9(a).
    28:S
    29:S                HEWLETT-PACKARD COMPANY
    30:D        0       Fort Collins, Colorado                              }
    31:S
    32:D        0 $modcal$
    33:D        0 program c_hook(output);
    34:S
    35:D        1 import sysdevs,sysglobals;
    36:S
    37:D        1 type
    38:D        1   sysflag_def = packed record
    39:D        1                   bit7,bit6,hpib,crt_config,
    40:D        1                   kbd,high,big_graph,alpha50 : boolean;
    41:D        1                 end;
    42:S
    43:D        1   crt_reg_def = packed record
    44:D        1                   selfinit,bit14,bit13,top1,top2,highlite,
    45:D        1                   graph,alpha,
    46:D        1                   bit7,bit6,bit5,bit4,bit3,bit2,bit1,bit0 : boolean;
    47:D        1                 end;
    48:S
    49:D        1   c_map_def   = packed array [0..15] of shortint;
    50:S
    51:D        1 var
    52:D        1   my_sysflag [ hex('fffed2') ] : sysflag_def;
    53:D        1   crt_reg [ hex('51fffe') ]    : crt_reg_def;
    54:D        1   c_map [ hex('51fb00') ]      : c_map_def;
    55:D        1   g_on [ hex ('51fffc') ] : shortint;
    56:D     -8 1   old_toggle_hook : procedure;
    57:D    -16 1   old_dump_hook   : procedure;
    58:D    -16 1   graphics_base ['GRAPHICSBASE'] : integer;
    59:S
    60:D        1 procedure dump_c;
```



```
    61:S
    62:D        2 { Purpose:  To dump 'C' bit map }
    63:S
    64:D        2 label 1;
    65:S
    66:D        2 const
    67:D        2   gwidthb = 64;
    68:D        2   gmaxheight = 512;
    69:D        2   gbuffersize = gwidthb + 6;
    70:S
    71:D        2 type
    72:D        2   gbyte = 0..255;
    73:D        2   row_def = packed array [0..(512*390)-1] of gbyte;
    74:S
    75:D        2 var
    76:D     -4 2   row : ^row_def;
    77:S
    78:D    -74 2   gbuffer : packed array [1..gbuffersize] of char;
    79:D    -82 2   i,j : integer;
    80:D    -86 2   index : integer;
    81:D    -90 2   bit_mask : integer;
    82:D    -94 2   result : integer;
    83:S
    84:C        2 begin
    85:S
    86:C        2   row := anyptr(graphics_base);
    87:S
    88:C        2   gbuffer[1] := chr(27); { escape sequence for graphics }
    89:C        2   gbuffer[2] := '*';
    90:C        2   gbuffer[3] := 'b';
    91:C        2   gbuffer[4] := '6';
    92:C        2   gbuffer[5] := '4';
    93:C        2   gbuffer[6] := 'W';
    94:S
    95:C        2   for j := 0 to 389 do
    96:C        3     begin
    97:C        3       for i := 0 to 63 do
    98:C        4         begin
    99:C        4           result := 0;
   100:C        4           index := j*512+i*8;
   101:C        4           bit_mask := 256;
   102:C        4           for index := index to index+7 do
   103:C        5             begin
   104:C        5               bit_mask := bit_mask div 2;
   105:C        5               if row^[index] <> 0 then result := bit_mask+result;
   106:C        5             end;
   107:C        4           gbuffer[i+7] := chr(result);
   108:C        4         end;
   109:C        3       write(gfiles[4]^,gbuffer:gwidthb+6);
   110:C        3       if ioresult <> ord(inoerror) then goto 1;
   111:C        3     end;
   112:S
   113:C        2   write(gfiles[4]^,#27'*rB');   { terminate graphics sequence }
   114:C        2 1:
   115:C        2 end;
   116:S
   117:D        1 procedure toggle_c;
   118:S
   119:C        2 begin
   120:C        2   graphicstate := not graphicstate;
```

```
   121:C        2   if graphicstate then g_on := 1
   122:C        3   else                 g_on := 0;
   123:C        2 end;
   124:S
   125:D        1 procedure init_c;
   126:S
   127:D        2 var
   128:D    -4  2   i : integer;
   129:S
   130:C        2 begin
   131:C        2   g_on      := 0;                    { turn off graphics screen }
   132:C        2   c_map[0] := -1;
   133:C        2   for i := 1 to 15 do
   134:C        3     c_map[i] := 0;                   { init color map is b&w }
   135:C        2   graphics_base      := hex('520000');
   136:C        2   togglegraphicshook := toggle_c;
   137:C        2   dumpgraphicshook   := dump_c;
   138:C        2 end;
   139:S
   140:D   -16  1 $partial_eval on$  { check for a HP 9836C }
   141:C        1 begin
   142:C        1   if (currentcrt = alphatype) and            { jws 1/18/84 }
   143:C        2     my_sysflag.big_graph and my_sysflag.crt_config and
   144:C        2     (not crt_reg.top1) and (crt_reg.top2) then
   145:C        2     init_c;
   146:C        1 end.
```

No errors. No warnings.
***** Nonstandard language features enabled *****

CLOCK

Description

CLOCK reads and sets the time using the keyboard clock.

Requirements

SYSGLOBALS, ASM, MISC, and SYSDEVS.

Notes

See also module SYSDEVS, and SYSGLOBALS for definitions.

```
                                                                (*
                (c) Copyright Hewlett-Packard Company, 1983.
               All rights are reserved.  Copying or other
               reproduction of this program except for archival
               purposes is prohibited without the prior
               written consent of Hewlett-Packard Company.


                           RESTRICTED RIGHTS LEGEND

               Use, duplication, or disclosure by the Government
               is subject to restrictions as set forth in
               paragraph (b) (3) (B) of the Rights in Technical
               Data and Computer Software clause in
               DAR 7-104.9(a).

               HEWLETT-PACKARD COMPANY
               Fort Collins, Colorado                           *)


             $modcal$
             $heap_dispose off$
             $iocheck off$
             $range off$ $ovflcheck off$
             $debug off$
             $stackcheck off$
             $search  'INITLOAD','ASM','INIT','SYSDEVS'$

             program clockinit;

             module clock;
             import sysglobals, asm, misc, sysdevs;
             export procedure initclock;

             implement

             type trickint = packed record
                                case integer of
                              0: ( ipart: integer );
                              1: ( byte3: byte;
                                   byte2: byte;
                                   byte1: byte;
                                   byte0: byte  )
                             end;

             var  boottype[-576]: shortint;

             procedure dosysdate(var thedate: daterec);
             var yr,dd,mm,k,k1,k2: integer;
                 ltime:  rtctime;
             begin
               CALL(CLOCKIOHOOK,CGET,LTIME);
               k:=ltime.packeddate+1;
               k1:= k*4-1;
               yr:= k1 div 1461;
               dd:= (k1-(1461*yr)+4) div 4;
               k2:=(5*dd-3);
               mm:=k2 div 153;
               dd:=k2-153*mm;
```



```
               dd:=(dd+5) div 5;
               if  mm<10 then mm:=mm+3
               else
                 begin mm:=mm-9;yr:=yr+1; end;
               with thedate do
                 begin
                   year:=yr mod 100;{to protect our file}
                   month:=mm;
                   day:=dd;
                 end;
             end;

             procedure dosystime(var thetime: timerec);
             var t: integer;
             begin
               t:=sysclock mod (24*360000);
               with thetime do
                 begin
                   hour := t div 360000;
                   minute := (t-(hour*360000)) div 6000;
                   centisecond := t mod 6000;
                 end;
             end;

             procedure setrtctime(thetime: rtctime);
             const
               cmmdb7=183;
               cmmd40=64;
             var
               t1,t2: trickint;
               TTIME: RTCTIME;
             begin
               TTIME:=THETIME; CALL(CLOCKIOHOOK,CSET,TTIME);
               t1.ipart := thetime.packeddate;
               t2.ipart := thetime.packedtime;
               batcommand(cmmdb7,5,t1.byte1,t1.byte0,t2.byte2,t2.byte1,t2.byte0);
               batcommand(cmmd40,0,0,0,0,0,0);
             end;    { setrtctime }

             procedure dosetsysdate(thedate: daterec);
             var   ltime: rtctime;  yr,mth,dy: integer;
             begin
               CALL(CLOCKIOHOOK,CGET,LTIME);
               with ltime,thedate do
                 begin
                   yr := year;  mth := month;  dy := day;
                   if mth>2 then mth:=mth-3
                            else begin  mth:=mth+9; yr:=yr-1; end;
                   packeddate:=((1461* yr) div 4 +(153*mth+2) div 5)+dy-1;
                 end;
               setrtctime(ltime);
             end;

             procedure dosetsystime(thetime: timerec);
             var   ltime: rtctime;  hr,min,ctsec: integer;
             begin
               CALL(CLOCKIOHOOK,CGET,LTIME);
               with ltime, thetime do
                 begin
                   hr := hour;  min := minute;  ctsec := centisecond;
```

```
   121:C        3       packedtime:=((hr*3600)+min*60)*100+ctsec;
   122:C        3     end;
   123:C        2   setrtctime(ltime);
   124:C        2 end;
   125:S
   126:D        1 procedure inittime;
   127:D    -8  2 var thetime:rtctime;
   128:D    -8  2 const
   129:D    -8  2   cmmd41=65;                    {65 hex to load timer output buffer with time}
   130:D    -8  2   cmmdf7=247;                   {F7 hex to load data buffer with first byte}
   131:D    -8  2   cmmdf6=246;                   {F6 hex to load data buffer with second byte}
   132:D    -8  2   cmmdf5=245;                   {F5 hex to load data buffer with third  byte}
   133:D    -8  2   cmmdf4=244;                   {F4 hex to load data buffer with fourth byte}
   134:D    -8  2   cmmdf3=243;                   {F3 hex to load data buffer with fifth  byte}
   135:S            cmmdf2=242;                   {F2 hex to load a letter 'B',or'P', or'H' for
   136:D    -8  2                                          Basic, or Pascal, or HPL respectively}
   137:D   -12  2 var  t: trickint;
   138:C        2 begin {inittime}
   139:C        2   thetime.packedtime := 0;
   140:C        2   thetime.packeddate := 0;
   141:C        2   if batterypresent then
   142:C        3    with t do
   143:C        4     begin
   144:C        4     setintlevel(2);
   145:S
   146:C        4     ipart := 0;
   147:C        4     batcommand(cmmd41,0,0,0,0,0,0);
   148:C        4     batcommand(cmmdf7,0,0,0,0,0,0);   byte1 := batbytereceived;
   149:C        4     batcommand(cmmdf6,0,0,0,0,0,0);   byte0 := batbytereceived;
   150:C        4     thetime.packeddate := ipart;
   151:S
   152:C        4     ipart := 0;
   153:C        4     batcommand(cmmdf5,0,0,0,0,0,0);   byte2 := batbytereceived;
   154:C        4     batcommand(cmmdf4,0,0,0,0,0,0);   byte1 := batbytereceived;
   155:C        4     batcommand(cmmdf3,0,0,0,0,0,0);   byte0 := batbytereceived;
   156:C        4     thetime.packedtime := ipart;
   157:S
   158:C        4     setintlevel(0);{lower cpu int level}
   159:C        4     end;
   160:C        2   setrtctime(thetime);
   161:C        2 end;    { inittime }
   162:S
   163:D        1 PROCEDURE DOCLOCKOPS(CMD:CLOCKFUNC; ANYVAR DATA: CLOCKDATA);
   164:C        2 BEGIN
   165:C        2   CASE CMD OF
   166:C        3   CGETDATE: DOSYSDATE(DATA.DATETYPE);
   167:C        3   CGETTIME: DOSYSTIME(DATA.TIMETYPE);
   168:C        3   CSETDATE: DOSETSYSDATE(DATA.DATETYPE);
   169:C        3   CSETTIME: DOSETSYSTIME(DATA.TIMETYPE);
   170:C        3   END;
   171:C        2 END;
   172:S
   173:D        1 procedure initclock;
   174:C        2 begin
   175:C        2   if boottype = 0 {powerup} then inittime;
   176:C        2   boottype := 18;
   177:C        2   CLOCKREQHOOK:=DOCLOCKOPS;
   178:C        2 end;
   179:S
   180:C        1 end; { module clock }
```



```
   181:S
   182:D        1 import clock, loader;
   183:S
   184:C        1 begin
   185:C        1   initclock;
   186:C        1   markuser;
   187:C        1 end.
   188:S

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

CONVERT

Description

CONVERT translates to or from a memory image of a WS1.0-format page of text when reading or writing any kind of text file.

Usage

CONVERT is used by the Compiler, Assembler, and Editor.

```
     1:S                                                    (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1983.
     4:S          All rights are reserved.  Copying or other
     5:S          reproduction of this program except for archival
     6:S          purposes is prohibited without the prior
     7:S          written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                       RESTRICTED RIGHTS LEGEND
    11:S
    12:S          Use, duplication, or disclosure by the Government
    13:S          is subject to restrictions as set forth in
    14:S          paragraph (b) (3) (B) of the Rights in Technical
    15:S          Data and Computer Software clause in
    16:S          DAR 7-104.9(a).
    17:S
    18:S          HEWLETT-PACKARD COMPANY
    19:D        0 Fort Collins, Colorado                          *)
    20:S
    21:S
    22:D        0 $modcal$
    23:D        0 $debug off, range off, ovflcheck off, stackcheck off, iocheck off$
    24:S
    25:D        0 module convert_text;
    26:S
    27:D        1 import sysglobals, misc, asm;
    28:S
    29:D        1 export
    30:S
    31:D        1 const pagesize = 1024;
    32:S
    33:D        1 type  pagebuftype = packed array[0..pagesize-1] of char;
    34:S
    35:D        1 procedure any_to_UCSD(var T: text; anyvar pagebuffer: pagebuftype);
    36:D        1 procedure UCSD_to_any(anyvar pagebuffer: pagebuftype; var T: text);
    37:S
    38:D        1 implement
    39:S
    40:D        1 procedure any_to_UCSD(var T: text; anyvar pagebuffer: pagebuftype);
    41:D        2 label 1;
    42:D        2 const strsize = 120;                    {arbitrary choice}
    43:D        2 type str = string[strsize];
    44:D        2      strptr = ^str;
    45:D        2      charptr = ^ char;
    46:D  -134  2 var i,j: integer;          s: str;          sp: strptr;
    47:D  -135  2     endline: boolean;
    48:S
    49:D        2   procedure iocheck;
    50:C        3   begin
    51:C        3   if ioresult <> ord(inoerror) then
    52:C        4     begin
    53:C        4     if ioresult = ord(ieof) then ioresult := ord(inoerror);
    54:C        4     goto 1;
    55:C        4     end;
    56:C        3   end;
    57:S
    58:C        2 begin
    59:C        2   i := 0; endline := false;
    60:C        2   while (i < pagesize - strsize) do
```



```
    61:C        3     begin
    62:C        3     if endline then
    63:C        4       begin
    64:C        4       sp := addr(pagebuffer[i],-1);
    65:C        4       read(T, sp^);
    66:C        4       j := strlen(sp^);
    67:C        4       charptr(sp)^ := eol;      iocheck;
    68:C        4       i := i + j;
    69:C        4       end
    70:C        4     else
    71:C        4       begin
    72:C        4       read(T, s); iocheck;
    73:C        4       moveleft(s[1], pagebuffer[i], strlen(s));
    74:C        4       i := i + strlen(s);
    75:C        4       end;
    76:C        3     endline := eoln(T); iocheck;
    77:C        3     if endline then begin
    78:C        4                     pagebuffer[i] := eol; i := i + 1;
    79:C        4                     readln(T); iocheck;
    80:C        4                     end;
    81:C        3     end;
    82:C        2   1: for i := i to pagesize-1 do pagebuffer[i] := nullchar;
    83:C        2 end;
    84:S
    85:D        1 $iocheck on$
    86:S
    87:D        1 procedure UCSD_to_any(anyvar pagebuffer: pagebuftype; var T: text);
    88:D        2 label 1;
    89:D        2 type pageptr = ^pagebuftype;
    90:D    -8  2 var i, j: integer;
    91:D    -9  2     c: char;
    92:C        2 begin
    93:C        2 try
    94:C        3 i := 0;
    95:C        3 repeat
    96:C        4   c := pagebuffer[i];
    97:C        4   if c = chr(dle) then
    98:C        5     begin
    99:C        5     i := i + 1;
   100:C        5     if i = pagesize then begin write(T, c); goto 1; end
   101:C        6     else
   102:C        6       begin
   103:C        6       c := pagebuffer[i];
   104:C        6       if c > ' ' then write(T, ' ':ord(c)-ord(' '));
   105:C        6       i := i + 1;
   106:C        6       if i = pagesize then goto 1;
   107:C        6       c := pagebuffer[i];
   108:C        6       end;
   109:C        5     end;
   110:C        4   j := i;
   111:C        4   while (c<>eol) and (c<>nullchar) do
   112:C        5     begin
   113:C        5     j := j + 1;
   114:C        5     if j = pagesize then c := nullchar
   115:C        6     else c := pagebuffer[j];
   116:C        5     end;
   117:C        4   write(T, pageptr(addr(pagebuffer[i]))^:j-i);
   118:C        4   if c = nullchar then goto 1;
   119:C        4   writeln(T);
   120:C        4   i := j + 1;
```

```
   121:C         4 until i = pagesize;
   122:C         3 1:
   123:C         3 recover if escapecode <> -10 then escape(escapecode);
   124:C         2 end;
   125:S
   126:C         1 end. {module convert_text}
   127:S
   128:S

No errors. No warnings.
             ***** Nonstandard language features enabled *****
```

Description

CRT is called as a transfer method (TM) by the file system, and contains the debugger window handler.

Requirements

SYSGLOBALS, ASM, MISC, and SYSDEVS.

Notes

See also SYSDEVS.

```
                                                        (*

         (c) Copyright Hewlett-Packard Company, 1984.
        All rights are reserved.  Copying or other
        reproduction of this program except for archival
        purposes is prohibited without the prior
        written consent of Hewlett-Packard Company.


                     RESTRICTED RIGHTS LEGEND

        Use, duplication, or disclosure by the Government
        is subject to restrictions as set forth in
        paragraph (b) (3) (B) of the Rights in Technical
        Data and Computer Software clause in
        DAR 7-104.9(a).

        HEWLETT-PACKARD COMPANY
      0 Fort Collins, Colorado                          *)

      0 $UCSD$
      0 $modcal$
      0 $heap_dispose off$
      0 $iocheck off$
      0 $range off$ $ovflcheck off$
      0 $debug off$
      0 $stackcheck off$
      0 $search  'INITLOAD','ASM','INIT','SYSDEVS'$

      0 program initcrt;

      1 module crt;
      1 import sysglobals, asm, misc, sysdevs;
      1 export
      1 function alphacrt: boolean;

      1 implement
      1
      1 const
      1 minkana = 161;
      1 maxkana = 223;
      1 yenromlocation = 128; { location of Yen symbol in old CRT rom }

      1 type


      1 kanatocrtlookuptype = packed array [minkana..maxkana] of 128..255;
      1 romtokanatype = packed array[#128..#238] of 0..255;

      1 crtregtype = 0..15;
      1 crtcmdwrd = packed record case integer of
      1                   0: (topbyte, botbyte: byte);
      1                   1: (longword: shortint);
      1                   2: (p1,p2, textfield, softfield: boolean);
      1                 end;

      1  crtscreen = array[0..maxint] of crtword;
      1  scrptr = ^crtscreen;
```



```
      1 const
      1 kanatocrtlookup = kanatocrtlookuptype [
      1     { code 161 }      129,130,131,132,133,134,135,
      1     { code 168 }  136,137,138,139,140,141,142,143,
      1     { code 176 }  144,145,146,147,148,149,150,151,
      1     { code 184 }  152,153,154,155,156,157,158,159,
      1     { code 192 }  160,161,162,163,164,165,166,167,
      1     { code 200 }  173,174,177,178,180,188,190,191,
      1     { code 208 }  224,225,226,227,228,229,230,231,
      1     { code 216 }  232,233,234,235,236,237,238,179  ];

      1   romtokanamap = romtokanatype          [  92, 161, 162,
      1       163, 164, 165, 166, 167, 168, 169, 170, 171, 172,
      1       173, 174, 175, 176, 177, 178, 179, 180, 181, 182,
      1       183, 184, 185, 186, 187, 188, 189, 190, 191, 192,
      1       193, 194, 195, 196, 197, 198, 199, 168, 169, 170,
      1       171, 172, 200, 201, 175, 176, 202, 203, 223, 204,
      1       181, 182, 183, 184, 185, 186, 187, 205, 189, 206,
      1       207, 192, 193, 194, 195, 196, 197, 198, 199, 200,
      1       201, 202, 203, 204, 205, 206, 207, 208, 209, 210,
      1       211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
      1       221, 222, 223, 208, 209, 210, 211, 212, 213, 214,
      1       215, 216, 217, 218, 219, 220, 221, 222];


      1   b9826info=crtirec[
      1            width :80,height:24,
      1            crtmemaddr:5316608          { + 416},
      1            crtcontroladdr:5341185,
      1            keybufferaddr: 5320448          {  + 416},
      1            progstateinfoaddr: 5320592          {  + 416},
      1            keybuffersize: 72,
      1            crtcon:
      1            crtconsttype [114,80,76,7,26,10,25,25,0,14,76,13],
      1            right(FS):chr(28),
      1            left(BS):chr(8),
      1            down(LF):chr(10),     up(US):chr(31),
      1            badch(?):chr(63),
      1            chardel(BS):chr(8),stop(DC3) :chr(19),
      1            break(DLE):chr(16),
      1            flush(ACK):chr(6),   eof(ETX):chr(3),
      1            altmode(ESC):chr(27),
      1            linedel(DEL):chr(127),
      1            backspace(BS):chr(8),
      1            etx:chr(3),prefix:chr(0),
      1            prefixed:b14[14 of false],
      1            cursormask : 0,     spare : 0];


      1   environc=environ[miscinfo:crtfrec[
      1                                   nobreak:false,
      1                                   stupid :false,
      1                                   slowterm:false,
      1                                   hasxycrt:true,
      1                                   haslccrt:true,  (?)
      1                                   hasclock:true,
      1                                   canupscroll:true,
      1                                   candownscroll:true],
      1                   crttype:0,
      1                   crtctrl:crtcrec[
```

```
   121:D        1                                    rlf:chr(31),
   122:D        1                                    ndfs:chr(28),
   123:D        1                                    eraseeol:chr(9),
   124:D        1                                    eraseeos:chr(11),
   125:D        1                                    home:chr(1),
   126:D        1                                    escape:chr(0),
   127:D        1                                    backspace:chr(8),
   128:D        1                                    fillcount:10,
   129:D        1                                    clearscreen:chr(0),
   130:D        1                                    clearline:chr(0),
   131:D        1                                    prefixed:b9[9 of false]],
   132:D        1                        crtinfo:crtirec[
   133:D        1                                    width :50,height:24,
   134:D        1                                    crtmemaddr:5316608,
   135:D        1                                    crtcontroladdr:5308417,
   136:D        1                                    keybufferaddr: 5319008,
   137:D        1                                    progstateinfoaddr: 5319092,
   138:D        1                                    keybuffersize: 42,
   139:D        1                                    crtcon: crtconsttype [64,50,49,10,25,9,25,
   140:D        1                                                          25,0,11,74,11],
   141:D        1                                    right(FS):chr(28),
   142:D        1                                    left(BS):chr(8),
   143:D        1                                    down(LF):chr(10),    up(US):chr(31),
   144:D        1                                    badch(?):chr(63),
   145:D        1                                    chardel(BS):chr(8),stop(DC3) :chr(19),
   146:D        1                                    break(DLE):chr(16),
   147:D        1                                    flush(ACK):chr(6),   eof(ETX):chr(3),
   148:D        1                                    altmode(ESC):chr(27),
   149:D        1                                    linedel(DEL):chr(127),
   150:D        1                                    backspace(BS):chr(8),
   151:D        1                                    etx:chr(3),prefix:chr(0),
   152:D        1                                    prefixed:b14[14 of false],
   153:D        1                                    cursormask : 0,     spare : 0]];
   154:S
   155:S
   156:D        1 var
   157:S
   158:D    -4  1 lptr:  scrptr;
   159:D    -8  1 screenwidth: integer;
   160:D   -12  1 screenheight: integer;
   161:S
   162:S
   163:D   -18  1 maxx,maxy,screensize:shortint;
   164:D   -22  1 screen:scrptr;
   165:D   -24  1 defaulthighlight: shortint;
   166:S
   167:D   -26  1 highlight:  shortint;
   168:D   -27  1 hascolor: boolean;
   169:D   -32  1 pm6845addrreg:^char;
   170:D   -36  1 pm6845comdreg:^char;
   171:D   -37  1 nomap:  boolean;                          { 3.0 bug fix jws 3/20/84 }
   172:S
   173:D   -37  1 crtidreg[hex('51FFFE')]: packed record
   174:D   -37  1       b15,b14,b13: boolean;
   175:D   -37  1       colorinfo: (cinfo0, cinfo1, cinfo2, cinfo3);
   176:D   -37  1       b10,b9,b8,b7,b6,b5,b4,b3,b2,b1,b0: boolean;
   177:D   -37  1       end;
   178:S
   179:S
   180:D        1 procedure dumpa;
```



```
   181:D        2 label 1;
   182:D    -8  2 var   row, column:integer;
   183:D    -9  2       c: char;
   184:D  -110  2       line: string[100];
   185:C        2 begin with syscom^.crtinfo do
   186:C        3   begin
   187:C        3   setstrlen(line, width);
   188:C        3   for row := 0 to height-1 do
   189:C        4     begin
   190:C        4     for column := 0 to width-1 do
   191:C        5       begin
   192:C        5       c := screen^[row*width+column].character;
   193:C        5       if (c >= #128) and (c <= #238) then c := chr(romtokanamap[c]);
   194:C        5       line[column+1] := c;
   195:C        5       end;
   196:C        4     column := width;
   197:C        4     while (column > 1) and (line[column]= ' ') do column := column - 1;
   198:C        4     writeln(gfiles[4]^, line:column);
   199:C        4     if ioresult <> ord(inoerror) then goto 1;
   200:C        4     end;
   201:C        3   end;
   202:C        2 1: end;
   203:S
   204:D        1 procedure toggleg;
   205:D        2 var gon [5439488{530000 HEX}]:shortint;
   206:D        2     goff[5472256{538000 HEX}]:shortint;
   207:D        2     gbase['GRAPHICSBASE']: ^shortint;
   208:S
   209:C        2 begin
   210:C        2   graphicstate:=not graphicstate;
   211:C        2   if graphicstate then gbase:=addr(gon)
   212:C        3                       else gbase:=addr(goff);
   213:C        2   gbase^ := gbase^;
   214:C        2 end;
   215:S
   216:D        1 procedure dumpg;
   217:D        2 label 1;
   218:D        2 const gheight = 300;    gheightb = 390;
   219:D        2       gwidth = 50;      gwidthb  = 64;
   220:D        2       gbuffersize=gwidthb+6;
   221:D        2 type  gword=packed record
   222:D        2               dummy,growbyte:char;
   223:D        2               end;
   224:D        2 gdotrow=packed array[1..gwidth] of gword;
   225:D        2 type gmemtype =  packed array [1..gheight] of gdotrow;
   226:D        2      gmembtype = packed array [1..gheightb, 1..gwidthb] of char;
   227:D        2      gmem =  ^gmemtype;
   228:D        2      gmemb = ^gmembtype;
   229:D        2 var   graphicsbase['GRAPHICSBASE']:  anyptr;
   230:D  -70   2       gbuffer:packed array[1..gbuffersize] of char;
   231:D  -90   2       i,j,rows,buffersize,pindex:integer;
   232:D  -91   2       busy:boolean;
   233:C        2 begin
   234:C        2   gbuffer[1]:=chr(esc) (escape sequence for graphics);
   235:C        2   gbuffer[2]:='*';
   236:C        2   gbuffer[3]:='b';
   237:C        2   gbuffer[6]:='L';
   238:C        2   if sysflag.biggraphics then
   239:C        3       begin
   240:C        3       gbuffer[4]:='6';
```

```
   24:C         3        gbuffer[5]:='4';
   25:C         3        rows := gheightb;
   26:C         3        buffersize := gwidthb+6;
   27:C         3        end
   28:C         3    else begin
   29:C         3        gbuffer[4]:='5';
   30:C         3        gbuffer[5]:='0';
   31:C         3        rows := gheight;
   32:C         3        buffersize := gwidth+6;
   33:C         3        end;
   34:C         2    for i:= 1 to rows do
   35:C         3      begin
   36:C         3      if sysflag.biggraphics then
   37:C         4       for j:=1 to gwidthb do gbuffer[j+6]:=gmemb(graphicsbase)^[i,j] else
   38:C         4         for j:=1 to gwidth  do gbuffer[j+6]:=gmem(graphicsbase)^[i,j].growbyte;
   39:C         3      write(gfiles[4]^, gbuffer:buffersize);
   40:C         3      if ioresult <> ord(inoerror) then goto 1;
   41:C         3      end;
   42:C         2    write(gfiles[4]^, #27'*rB'); {terminate graphics sequence};
   43:C         2  1:
   44:C         2  end;
   45:S
   46:D         1  procedure crtcommand(reg: crtregtype; data: byte);
   47:C         2  begin
   48:C         2      pm6845addrreg^ := chr(reg);
   49:C         2      pm6845comdreg^ := chr(data);
   50:C         2  end;
   51:D   -37   1
   52:D         1  procedure doupdatecursor;
   53:D    -2   2  var cursaddr: crtcmdwrd;
   54:C         2  begin
   55:C         2    cursaddr.longword:=integer(screen) mod 8192 div 2 + ypos*screenwidth+xpos;
   56:C         2    cursaddr.textfield := alphastate;
   57:C         2    cursaddr.softfield:=alphastate;
   58:C         2    crtcommand(14, cursaddr.topbyte);
   59:C         2    crtcommand(15, cursaddr.botbyte);
   60:C         2  end;
   61:S
   62:D         1  procedure togglea;
   63:D    -2   2  var   lcursaddr:crtcmdwrd;
   64:S
   65:C         2  begin
   66:C         2    alphastate:=not(alphastate);
   67:C         2    lcursaddr.longword:=integer(screen) mod 8192 div 2;
   68:C         2    lcursaddr.textfield:=alphastate;
   69:C         2    lcursaddr.softfield:=alphastate;
   70:C         2    crtcommand(12, lcursaddr.topbyte);
   71:C         2    crtcommand(13, lcursaddr.botbyte);
   72:C         2    doupdatecursor;
   73:C         2  end;
   74:S
   75:S
   76:D         1  procedure getxy(var x,y: integer);
   77:C         2  begin
   78:C         2  x := xpos;      y := ypos;
   79:C         2  end;
   80:S
   81:D         1  procedure setxy(x, y: shortint);
   82:C         2  begin
   83:C         2    if x>=screenwidth then xpos:=maxx
```



```
   84:C         3    else if x<0 then xpos:=0
   85:C         4    else xpos := x;
   86:C         2    if y>=screenheight then ypos:=maxy
   87:C         3    else if y<0 then ypos:=0
   88:C         4    else ypos := y;
   89:C         2  end;
   90:S
   91:D         1  procedure gotoxy(x,y: integer);
   92:C         2  begin
   93:C         2    setxy(x,y);
   94:C         2    doupdatecursor;
   95:C         2  end;
   96:S
   97:S
   98:D         1  procedure clear(number: shortint);
   99:D    -4   2  var x,y: shortint;
  100:C         2  begin
  101:C         2    x:=xpos; y:=ypos;
  102:C         2    while number>0 do
  103:C         3      begin
  104:C         3        screen^[y*screenwidth+x].wholeword:= ord(' ');
  105:C         3        number:=number-1;
  106:C         3        if x<maxx then x:=x+1
  107:C         4        else begin x:=0; if y<maxy then y:=y+1 end;
  108:C         3      end;
  109:C         2  end;
  110:S
  111:D         1  procedure scrollup;
  112:D    -2   2  var i: shortint;
  113:C         2  begin
  114:C         2    moveleft(screen^[screenwidth{1, 0}],
  115:C         2             screen^[0{0, 0}],
  116:C         2             (screensize-screenwidth)*2);
  117:C         2    for i:=0 to maxx do
  118:C         3      screen^[maxy*screenwidth+i].wholeword:=ord(' ');
  119:C         2  end;
  120:S
  121:D         1  procedure scrolldown;{new  4/30/81}
  122:D    -2   2  var i: shortint;
  123:C         2  begin
  124:C         2    moveright(screen^[0{0, 0}],
  125:C         2             screen^[screenwidth{1, 0}],
  126:C         2             (screensize-screenwidth)*2);
  127:C         2    for i:=0 to maxx do screen^[i].wholeword := ord(' ');
  128:C         2  end;
  129:S
  130:S
  131:S
  132:S
  133:S
  134:D         1  function maptocrt(c:char):char;
  135:S
  136:D         2  const illegalchar = #223;
  137:D         2                    { char to disp for illegal internal codes; looks like hp }
  138:D         2  procedure mapromextocrt;
  139:D         3  const
  140:D         3        minromex = 168; { lookup table ranges }
  141:D         3        maxromex = 255;
  142:D         3  type  romexsettype = set of minromex..maxromex;
  143:D         3  const romexset = romexsettype [168..172,175,176,179,181..187,189,192..222,255];
  144:D         3                                                    { legal Romex codes }
```

```
361:C        3 begin
362:C        3     if (ord(c) < 128) or (ord(c) in romexset)
363:C        4                       or nomap  then    {  3.0 bug fix  jws 3/20/84}
364:C        4       maptocrt:=c
365:C        4     else
366:C        4       if ord(c)=188  { ROMAN8 yen char}                  { jws 3/1/84 }
367:C        5         then maptocrt:=chr(yenromlocation)               { jws 3/1/84 }
368:C        5       else
369:C        5         maptocrt:=illegalchar;
370:C        3 end;
371:S
372:S
373:D        2 procedure mapkanatocrt ;
374:S
375:D        3 const yencode = 92;
376:S
377:S
378:S          { Converts Katakana codes to their correct "old" CRT rom location codes; also,
379:S            converts "illegal" Kana chars to the "hp" char.  Note that the Yen symbol
380:D        3   overlays the USASCII backslash (\), and that code 255 is left unconverted. }
381:S
382:S
383:C        3 begin
384:C        3   if nomap then maptocrt:=c
385:C        4   else begin
386:C        4     if ord(c) = yencode then maptocrt := chr(yenromlocation)
387:C        5     else if (ord(c) < 128) or (ord(c) = 255) then maptocrt:= c
388:C        6     else begin
389:C        6       if (ord(c) < minkana) or (ord(c) > maxkana) then maptocrt := illegalchar
390:C        7       else maptocrt := chr(kanatocrtlookup[ord(c)]);
391:C        6     end;
392:C        4   end;
393:C        3 end; { mapkanatocrt }
394:S
395:C        2 begin
396:C        2   if kbdlang = katakana_kbd then mapkanatocrt
397:C        3   else mapromextocrt;
398:C        2 end;
399:S
400:S
401:S
402:D        1 procedure docrtio(fp: fibp; request: amrequesttype; anyvar buffer: window;
403:D        2                                                     length, position: integer);
404:D    -1  2 var c: char;
405:D    -4  2     s: string[1];
406:D    -8  2     buf: charptr;
407:C        2 begin
408:C        2  ioresult := ord(inoerror);
409:C        2  buf := addr(buffer);
410:C        2  case request of
411:C        3   {uwait: ;                }
412:C        3   setcursor: gotoxy(fp^.fxpos, fp^.fypos);
413:C        3   getcursor: getxy (fp^.fxpos, fp^.fypos);
414:C        3   flush:  {do nothing};
415:C        3   unitstatus:  kbdio(fp, request, buffer, length, position);
416:C        3   clearunit: highlight := defaulthighlight;
417:C        3   readtoeol:
418:C        3     begin
419:C        3     buf := addr(buf^, 1);
420:C        3     buffer[0] := chr(0);
```



```
421:C        3     while length>0 do
422:C        4       begin
423:C        4       kbdio(fp, readtoeol,  s, 1, 0);
424:C        4       if     strlen(s)=0     then length := 0
425:C        5     { else if s[1] = chr(etx) then length := 0 }
426:C        5       else  begin
427:C        5             length := length - 1;
428:C        5             crtio(fp, writebytes, s[1], 1, 0);
429:C        5             buf := addr(buf^, 1);
430:C        5             buffer[0] := chr(ord(buffer[0])+1);
431:C        5             end;
432:C        4       end;
433:C        3     end;
434:C        3   startread,
435:C        3   readbytes:
436:C        3     begin
437:C        3     while length>0 do
438:C        4       begin
439:C        4       kbdio(fp, readbytes,  buf^, 1, 0);
440:C        4       if buf^ = chr(etx) then length := 0
441:C        5                          else length := length - 1;
442:C        4       if buf^ = eol then crtio(fp, writeeol,   buf^, 1, 0)
443:C        5                     else crtio(fp, writebytes, buf^, 1, 0);
444:C        4       buf := addr(buf^, 1);
445:C        4       end;
446:C        3     if request = startread then call(fp^.feot, fp);
447:C        3     end;
448:C        3   writeeol:   begin
449:C        3                 if ypos=maxy then scrollup;
450:C        3                 gotoxy(0, ypos+1);
451:C        3               end;
452:C        3   startwrite,
453:C        3   writebytes:
454:C        3     begin
455:C        3     while length>0 do
456:C        4      begin
457:C        4        c:=buf^; buf:=addr(buf^,1); length:=length-1;
458:S
459:C        4        case c of
460:C        5        homechar:   setxy(0,0);
461:C        5        leftchar:   if (xpos = 0) and (ypos>0) then setxy(maxx, ypos-1)
462:C        6                    else setxy(xpos-1, ypos);
463:C        5        rightchar:  if (xpos = maxx) and (ypos<maxy) then setxy(0, ypos+1)
464:C        6                    else setxy(xpos+1, ypos);
465:C        5        upchar:     begin if ypos <= 1  then scrolldown;
466:C        5                          if ypos>0 then setxy(xpos, ypos-1);
467:C        5                    end;
468:C        5        downchar:   if ypos=maxy then scrollup
469:C        6                    else setxy(xpos, ypos+1);
470:C        5        bellchar:   beep;
471:C        5        cteos:      clear(screensize-(ypos*screenwidth+xpos));
472:C        5        cteol:      clear(screenwidth-xpos);
473:C        5        clearscr:   begin setxy(0,0); clear(screensize); end;
474:C        5        eol:        setxy(0, ypos);
475:C        5        chr(etx):   length:=0;
476:C        5        otherwise   if (ord(c)>=128) and (ord(c)< 144) then
477:C        6                      if hascolor then
478:C        7                        if ord(c) >= 136 then highlight :=
479:C        8                                              highlight mod 2048 + (ord(c)-136)*4096
480:C        8                        else highlight :=
```

```
481:C        8                                  (highlight div 2048 * 8 + (ord(c)-128))*256
482:C        8                        else highlight := (ord(c)-128)*256
483:C        7                      else with screen^[ypos*screenwidth+xpos] do
484:C        7                        begin
485:C        7                         wholeword:=highlight+ ord(maptocrt(c));
486:C        7                         if xpos = maxx then
487:C        8                           begin
488:C        8                             if ypos = maxy then scrollup;
489:C        8                             setxy(0, ypos+1);
490:C        8                           end
491:C        8                         else setxy(xpos+1, ypos);
492:C        7                        end;
493:C        5          end;
494:C        4       doupdatecursor;
495:C        4       end; {while}
496:C        3      if request = startwrite then call(fp^.feot, fp);
497:C        3      end;
498:C        3    otherwise ioresult := ord(ibadrequest);
499:C        3   end; {case}
500:C        2 end;
501:S
502:D        1 procedure lineops(op: crtllops; anyvar position: integer; c:char);
503:D        2 var
504:D    -4  2   i: integer;
505:D    -8  2   sptr: ^string255;
506:S
507:C        2 begin
508:C        2 case op of
509:C        3   cllput: lptr^[position].wholeword:=ord(maptocrt(c));
510:S
511:C        3   cllshiftl:
512:C        3      begin
513:C        3        for i:=0 to (maxx-8) do lptr^[i]:=lptr^[i+1];
514:C        3        lptr^[maxx-8].wholeword:=ord(' ');
515:C        3      end;
516:S
517:C        3   cllshiftr:
518:C        3      begin
519:C        3        for i:=0 to (maxx-9) do lptr^[maxx-8-i]:=lptr^[maxx-9-i];
520:C        3        lptr^[0].wholeword:=ord(' ');
521:C        3      end;
522:S
523:C        3   cllclear:
524:C        3      for i:=0 to (maxx-8) do lptr^[i].wholeword:=ord(' ');
525:S
526:C        3   clldisplay:
527:C        3      begin
528:C        3        sptr:=addr(position);
529:C        3        for i:=1 to length(sptr^) do
530:C        4          lptr^[i-1].wholeword:=ord(maptocrt(sptr^[i]));
531:C        3        for i:=length(sptr^) to (maxx-8) do
532:C        4          lptr^[i].wholeword:=ord(' ');
533:C        3      end;
534:S
535:C        3   putstatus:
536:C        3      begin { position should be in range 0..7 }
537:C        3        lptr^[maxx-7+position].wholeword:=ord(c);
538:C        3      end;
539:S
540:C        3 end; { case}
```



```
541:S
542:C        2 end; { lineops }
543:S
544:D        1 procedure crtdebug(op: dbcrtops; var dbrec: dbcinfo );
545:S
546:D        2 type iptr = ^iarray;
547:D        2      iarray = array[0..maxint] of shortint;
548:S
549:D        2 var
550:D    -4  2   xtemp, ytemp: shortint;
551:D   -10  2   i,j,k: shortint;
552:D   -12  2   len: shortint;
553:D   -14  2   inc: shortint;
554:D  -174  2   temp: array[0..79] of shortint;
555:S
556:S
557:C        2 begin
558:C        2 with dbrec do begin
559:C        3   case op of
560:S
561:C        4     dbinfo: savesize:=(xmax-xmin+1)*(ymax-ymin+1)*2;
562:S
563:C        4     dbgotoxy:
564:C        4       begin
565:C        4         xtemp:=xpos; ytemp:=ypos;
566:C        4         xpos:=cursx; ypos:=cursy;
567:C        4         doupdatecursor;
568:C        4         xpos:=xtemp; ypos:=ytemp;
569:C        4       end;
570:S
571:C        4     dbscrollup,dbscrolldn:
572:C        4       begin
573:C        4         len:=(xmax-xmin+1)*2;
574:C        4         if op=dbscrollup then begin
575:C        5           j:=ymin;
576:C        5           inc:=screenwidth;
577:C        5         end
578:C        5         else begin
579:C        5           j:=ymax;
580:C        5           inc:=-screenwidth;
581:C        5         end;
582:C        4         j:=j*screenwidth+xmin;
583:C        4         for i:=(ymin+1) to ymax do begin
584:C        5           k:=j; j:=j+inc;
585:C        5           moveleft(screen^[j], screen^[k], len);
586:C        5         end;
587:C        4         for i:=0 to (xmax-xmin) do
588:C        5           screen^[j+i].wholeword:=ord(' ');
589:C        4       end;
590:S
591:C        4     dbscrolll,dbscrollr:
592:C        4       begin
593:C        4         len:=(xmax-xmin+1)*2-2;     { fixed  4/13/84 }
594:C        4         j:=(ymin-1)*screenwidth+xmin;
595:C        4         if op=dbscrolll then begin
596:C        5           j:=j+1;
597:C        5           k:=xmax-xmin-1;
598:C        5         end
599:C        5         else begin
600:C        5           k:=0;
```

```
   601:C        5          end;
   602:C        4          for i:=ymin to ymax do begin
   603:C        5           j:=j+screenwidth;
   604:C        5           if op=dbscrolll then
   605:C        6             moveleft(screen^[j],screen^[j-1], len)
   606:C        6           else
   607:C        6             moveright(screen^[j],screen^[j+1], len);
   608:C        5           screen^[j+k].wholeword:=ord(' ');
   609:C        5          end;
   610:C        4        end;
   611:S
   612:C        4      dbhighl:
   613:C        4        begin
   614:C        4          i:=cursy*screenwidth+cursx;
   615:C        4          screen^[i].wholeword:=ord(screen^[i].character)+(ord(c)-128)*256;
   616:C        4        end;
   617:S
   618:S
   619:C        4      dbput: screen^[cursy*screenwidth+cursx].wholeword:=ord(c);
   620:S
   621:C        4      dbclear:
   622:C        4        for i:=ymin to ymax do
   623:C        5          for j:=xmin to xmax do
   624:C        6            screen^[i*screenwidth+j].wholeword:=ord(' ');
   625:S
   626:C        4      dbcline:
   627:C        4        for i:=cursx to xmax do
   628:C        5          screen^[cursy*screenwidth+i].wholeword:=ord(' ');
   629:S
   630:C        4      dbinit:
   631:C        4        begin
   632:C        4          for i:=0 to (savesize div 2)-1 do
   633:C        5            iptr(savearea)^[i]:=ord(' ');
   634:C        4          cursx:=xmin; cursy:=ymin;
   635:C        4          areaisdbcrt:=true;
   636:C        4        end;
   637:S
   638:C        4      dbexcg:
   639:C        4        begin
   640:C        4          k:=xmax-xmin+1;
   641:C        4          for i:=ymin to ymax do begin
   642:C        5            moveleft(screen^[i*screenwidth+xmin], temp, k*2);
   643:C        5            moveleft(iptr(savearea)^[(i-ymin)*k],
   644:C        5                     screen^[i*screenwidth+xmin], k*2);
   645:C        5            moveleft(temp, iptr(savearea)^[(i-ymin)*k], k*2);
   646:C        5          end;
   647:C        4          if areaisdbcrt then begin
   648:C        5            xtemp:=xpos; ytemp:=ypos;
   649:C        6            xpos:=cursx; ypos:=cursy;
   650:C        5            doupdatecursor;
   651:C        6            xpos:=xtemp; ypos:=ytemp;
   652:C        5          end
   653:C        5          else doupdatecursor;
   654:C        4          areaisdbcrt:=not areaisdbcrt;
   655:C        4        end;
   656:S
   657:S
   658:C        4
   659:C        4    end; { of case }
   660:C        3  end; { of with }
```



```
   661:C        2 end; { procedure crtdebug }
   662:S
   663:D        1 procedure alphacrtinit;
   664:D   -10  2  var cursaddr: crtcmdwrd; i,k: integer;
   665:C        2  begin
   666:C        2    with syscom^.crtinfo do
   667:C        3    begin
   668:C        3      screer:=anyptr(crtmemaddr);
   669:C        3      screerwidth:=width;
   670:C        3      screerheight:=height;
   671:C        3      maxx:=width-1;
   672:C        3      maxy:=height-1;
   673:C        3      screensize:=width*height;
   674:S
   675:C        3      for i:=0 to screensize-1 do screen^[i].wholeword:=ord(' ');  {clear screen}
   676:C        3      pm6845addrreg:=anyptr(crtcontroladdr);
   677:C        3      pm6845comdreg:=anyptr(crtcontroladdr+2);
   678:C        3      cursaddr.longword:=integer(screen) mod 8192 div 2;
   679:C        3      cursaddr.texifield:=alphastate;
   680:C        3      cursaddr.softfield:=alphastate;
   681:C        3      crtcommand(12, cursaddr.topbyte);
   682:C        3      crtcommand(13, cursaddr.botbyte);
   683:C        3      defaulthighlight := 0; highlight := 0;
   684:S
   685:C        3      idle:=250;
   686:C        3      nomap:=false;
   687:C        3      if sysflag.crtconfigreg then begin
   688:C        4        if crtidreg.b13 then begin        { 3.0 bug jws 3/20/84 }
   689:C        5          nomap:=true;                 { 3.0 bug jws 3/20/84 }
   690:C        5          idle:=245;                      { 3.0 bug jws 3/20/84 }
   691:C        5        end;                              { 3.0 bug jws 3/20/84 }
   692:C        4        hascolor := crtidreg.colorinfo > cinfo0;
   693:C        4      end
   694:C        4      else hascolor := false;
   695:S
   696:C        3      gotoxy(0,0);
   697:C        3      dumpalphahook := dumpa;
   698:C        3      dumpgraphicshook := dumpg;
   699:C        3      togglealphahook := togglea;
   700:C        3      togglegraphicshook := toggleg;
   701:C        3      updatecursorhook:=doupdatecursor;
   702:C        3      crtiohook:=docrtio;
   703:C        3      crtllhook:=lineops;
   704:C        3      dbcrthook:=crtdebug;
   705:C        3      crtinithook:=alphacrtinit;
   706:C        3      lptr:=anyptr(keybufferaddr);
   707:C        3      keybuffer^.maxsize:=maxx-8;
   708:C        3      currentcrt:=alphatype;
   709:C        3    end;
   710:C        2  end;
   711:D   -37  1
   712:D        1 function alphacrt:boolean;
   713:D        2 var i[hex('512000')]:shortint;
   714:D    -2  2     j: shortint;
   715:C        2 begin
   716:C        2 alphacrt:=true;  { assume we have alpha screen }
   717:C        2 try
   718:C        3   j:=i;   { attempt read from alpha screen ram }
   719:C        3   syscom^:=environc;  { setup for my kind of environment }
   720:C        3   if not sysflag.alpha50 then syscom^.crtinfo:=b9826info;
```

```
721:C        3   alphacrtinit;
722:C        3 recover
723:C        3   if escapecode=-12 then
724:C        4     alphacrt:=false  { bus error -- no alpha screen }
725:C        4   else escape(escapecode);
726:C        2 end;
727:S
728:C        1 end; {  of module }
729:S
730:S
731:D        1 import crt, loader;
732:S
733:C        1 begin
734:C        1   if alphacrt then markuser;
735:C        1 end.
736:S

No errors. No warnings.
             ***** Nonstandard language features enabled *****
```

Description

CS80 is a transfer method (TM) for all supported Command Set '80 and Subset '80 discs.

Usage

CS80 supports the following CS/80 discs:

- 7908
- 7911
- 7912
- 7914
- 7935
- 7941
- 7944 (tape drive only; no disc)
- 7945

CS80 supports the following SS/80 discs:

- 9133
- 9134

Requirements

DISCHPIB, DRVASM, IODECLARATIONS and DISC_INTF

Optionally: DMA

Notes

CS80 does support the 7933 discs (see AMIGO).

```
    :S                                                  (*
    :S
    :S          (c) Copyright Hewlett-Packard Company, 1983.
    :S          All rights are reserved.  Copying or other
    :S          reproduction of this program except for archival
    :S          purposes is prohibited without the prior
    :S          written consent of Hewlett-Packard Company.
    :S
    :S
    :S                      RESTRICTED RIGHTS LEGEND
    :S
    :S          Use, duplication, or disclosure by the Government
    :S          is subject to restrictions as set forth in
    :S          paragraph (b) (3) (B) of the Rights in Technical
    :S          Data and Computer Software clause in
    :S          DAR 7-104.9(a).
    :S
    :S          HEWLETT-PACKARD COMPANY
    :D        0 Fort Collins, Colorado                          *)
    :S
    :S
    :D        0 $modcal$
    :D        0 $debug off, range off, ovflcheck off$
    :D        0 $stackcheck off, iocheck off$
    :D        0 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'$
    :S
    :D        0 $search 'DRVASM', 'DISCHPIB', 'IOLIB:KERNEL'$
    :S
    :S
    :D        0 program CS80init;

    :D        1 module tapebuf;

    :S
    :D        1 import
    :D        1   sysglobals, bkgnd;
    :S
    :D        1 export
    :D        1   const
    :D        1     tapebuf_maxsize = 1024;
    :D        1   type
    :D        1     tapebuf_type = packed array[0..tapebuf_maxsize-1] of byte;
    :D        1     tapebuf_state_type = (undefined, unmodified, modified);
    :D        1   var
    :D   -4   1     tapebuf_ptr: ^tapebuf_type;
    :D   -6   1     tapebuf_state: tapebuf_state_type;
    :D  -10   1     tapebuf_uep: uep_type;
    :D  -14   1     tapebuf_block: integer;
    :D  -18   1     tapebuf_size: integer;
    :D  -18   1   procedure init_tapebuf;
    :S
    :D  -18   1 implement {tapebuf}
    :S
    :D        1 procedure init_tapebuf;
    :C        2   begin {init_tapebuf}
    :C        2     if tapebuf_ptr=nil then
    :C        3       new(tapebuf_ptr);
    :C        2     tapebuf_state := undefined;
    :C        2   end; {init_tapebuf}
    :S
    :C        1 end; {tapebuf}
```



```
    :D        1 $page$
    :S
    :D        1 module CS80; {Command Set '80}
    :S
    :D        1 import
    :D        1   sysglobals, bkgnd, discHPIB;
    :S
    :D        1 export

    :S
    :D        1   type
    :D        1     signed16 = -32768..32767;
    :D        1     signed8  = -128..127;
    :D        1
    :D        1     unsgn24  = 0..16777215;
    :D        1     unsgn8   = 0..255;
    :D        1     unsgn4   = 0..15;
    :D        1
    :D        1     ct_type =  {controller type field in describe}
    :D        1       packed record
    :D        1         b7, b6, b5, b4, b3:  boolean;
    :D        1         subset80:            boolean;
    :D        1         multiport:           boolean;
    :D        1         multiunit:           boolean;
    :D        1       end;
    :S
    :D        1     sva_type = {single-vector address (6 bytes)}
    :D        1       packed record
    :D        1         utb: signed16;  {upper two bytes}
    :D        1         lfb: integer;   {lower four bytes (all we manage internally)}
    :D        1       end;
    :D        1
    :D        1     describe_type = {info returned by describe of unit other than controller}
    :D        1       packed record
    :D        1                           {CONTROLLER DESCRIPTION FIELD}
    :D        1         iu: signed16;                   {installed unit word: 1 bit per unit}
    :D        1         mitr: signed16;                 {max instantaneous xfr rate (Kbytes)}
    :D        1         ct: ct_type;                    {controller type}
    :D        1                           {UNIT DESCRIPTION FIELD}
    :D        1         dt: signed8;                    {generic device type}
    :D        1         dn: unsgn24;                    {device number (6 BCD digits)}
    :D        1         nbpb: signed16;                 {# of bytes per block}
    :D        1         nbb: unsgn8;                    {# of blocks which can be buffered}
    :D        1         rbs: unsgn8;                    {recommended burst size}
    :D        1         blocktime: signed16;            {block time in microseconds}
    :D        1         catr: signed16;                 {continuous avg xfr rate (Kbytes)}
    :D        1         ort: signed16;                  {optimal retry time in centiseconds}
    :D        1         atp: signed16;                  {access time parameter in centiseconds}
    :D        1         mif: unsgn8;                    {maximum interleave factor}
    :D        1         fvb: unsgn8;                    {fixed volume byte: 1 bit/volume}
    :D        1         rvb: unsgn8;                    {removeable volume byte: 1 bit/vol}
    :D        1                           {VOLUME DESCRIPTION FIELD}
    :D        1         maxcadd: unsgn24;               {maximum cylinder address}
    :D        1         maxhadd: unsgn8;                {maximum head address}
    :D        1         maxsadd: signed16;              {maximum sector address}
    :D        1         maxsvadd: sva_type;             {maximum single-vector address}
    :D        1         currentif: unsgn8;              {current interleave factor}
    :D        1       end;
```

```
118:D  1 $page$
119:D  1
120:D  1     evu_type = (encoded volume/unit (1 byte) - status & copy commands)
121:D  1       packed record case integer of
122:D  1         0: (vvvv: unsgn4;       {volume number}
123:D  1             uuuu: unsgn4};      {unit number}
124:D  1         1: (evu_byte: signed8); {for -1 test}
125:D  1       end;
126:D  1
127:S
128:D  1     errorbit_type = (error bit assignments in status & status mask)
129:D  1       (
130:D  1                     {REJECT ERRORS FIELD}
131:D  1         ( 0)  eb0,
132:D  1         ( 1)  eb1,
133:D  1         ( 2)  channel_parity_error,
134:D  1         ( 3)  eb3,
135:D  1         ( 4)  eb4,
136:D  1         ( 5)  illegal_opcode,
137:D  1         ( 6)  module_addressing,
138:D  1         ( 7)  address_bounds,
139:D  1         ( 8)  parameter_bounds,
140:D  1         ( 9)  illegal_parameter,
141:D  1         (10)  message_sequence,
142:D  1         (11)  eb11,
143:D  1         (12)  message_length,
144:D  1         (13)  eb13,
145:D  1         (14)  eb14,
146:D  1         (15)  eb15,
147:D  1                     {FAULT ERRORS FIELD}
148:D  1         (16)  eb16,
149:D  1         (17)  cross_unit,
150:D  1         (18)  eb18,
151:D  1         (19)  controller_fault,
152:D  1         (20)  eb20,
153:D  1         (21)  eb21,
154:D  1         (22)  unit_fault,
155:D  1         (23)  eb23,
156:D  1         (24)  diagnostic_result,
157:D  1         (25)  eb25,
158:D  1         (26)  operator_release_required,
159:D  1         (27)  diagnostic_release_required,
160:D  1         (28)  internal_maintenance_required,
161:D  1         (29)  eb29,
162:D  1         (30)  power_fail,
163:D  1         (31)  retransmit,
164:D  1                     {ACCESS ERRORS FIELD}
165:D  1         (32)  illegal_parallel_operation,
166:D  1         (33)  uninitialized_media,
167:D  1         (34)  no_spares_available,
168:D  1         (35)  not_ready,
169:D  1         (36)  write_protect,
170:D  1         (37)  no_data_found,
171:D  1         (38)  eb38,
172:D  1         (39)  eb39,
173:D  1         (40)  unrecoverable_data_overflow,
174:D  1         (41)  unrecoverable_data,
175:D  1         (42)  eb42,
176:D  1         (43)  end_of_file,
177:D  1         (44)  end_of_volume,
```



```
178:D  1         (45)  eb45,
179:D  1         (46)  eb46,
180:D  1         (47)  eb47,
181:D  1                     {INFORMATION ERRORS FIELD}
182:D  1         (48)  operator_request,
183:D  1         (49)  diagnostic_request,
184:D  1         (50)  internal_maintenance_request,
185:D  1         (51)  media_wear,
186:D  1         (52)  latency_induced,
187:D  1         (53)  eb53,
188:D  1         (54)  eb54,
189:D  1         (55)  auto_sparing_invoked,
190:D  1         (56)  eb56,
191:D  1         (57)  recoverable_data_overflow,
192:D  1         (58)  marginal_data,
193:D  1         (59)  recoverable_data,
194:D  1         (60)  eb60,
195:D  1         (61)  maintenance_track_overflow,
196:D  1         (62)  eb62,
197:D  1         (63)  eb63
198:D  1       );
199:D  1
200:D  1
201:D  1     status_mask_type =
202:D  1       packed array[errorbit_type] of boolean;
203:D  1
204:D  1
205:D  1     status_type =
206:D  1       packed record
207:D  1                     {IDENTIFICATION FIELD}
208:D  1         current_vu: evu_type;                          {current volume/unit}
209:D  1         requesting_unit: signed8;                      {unit requesting service}
210:D  1                     {ERROR REPORTING FIELDS}
211:D  1         errorbits: status_mask_type;
212:D  1                     {PARAMETER FIELD}
213:D  1         case integer of
214:D  1           {positive cases correspond to error bits}
215:D  1           -1: (nta: sva_type;                           {new target address}
216:D  1                faultlog: integer);                      {fault log}
217:D  1           -2: (aaa: sva_type;                           {affected area address}
218:D  1                afl: integer);                           {affected field length}
219:D  1            17: (uee: packed array[1..6] of signed8);   {units experiencing errors}
220:D  1            24: (dor: packed array[1..6] of unsgn8);    {diagnostic results}
221:D  1            38: (ta: sva_type);                         {target address}
222:D  1            41: (bba: sva_type);                        {bad block address}
223:D  1        48..50: (urr: packed array[1..6] of signed8);   {units requesting release}
224:D  1            58: (btbs: sva_type);                       {block to be spared}
225:D  1            59: (rba: sva_type);                        {recoverable block address}
226:D  1         end; {case}
```

```
227:D        1 $page$
228:D        1
229:D        1     t_type = {'T' parameter in SET_RELEASE}
230:D        1       (   allow_release_timeout,          {power-on default}
231:D        1         suppress_release_timeout);
232:D        1
233:D        1
234:D        1     z_type = {'Z' parameter in SET_RELEASE}
235:D        1       (disable_auto_release,              {power-on default}
236:D        1         enable_auto_release);
237:S
238:D        1
239:D        1     CMD_type = (enumerated opcodes for device commands)
240:D        1 (CMDlocate_and_read, CMD1              , CMDlocate_and_wrt , CMD3              ,
241:D        1  CMDlocate_and_ver , CMD5              , CMDspare_block    , CMD7              ,
242:D        1  CMDcopy_data      , CMD9              , CMDcold_load_read , CMD11             ,
243:D        1  CMD12             , CMDrequest_status , CMDrelease        , CMDrelease_denied ,
244:D        1  CMDset_address_1V , CMDset_address_3V , CMDset_block_disp , CMD19             ,
245:D        1  CMD20             , CMD21             , CMD22             , CMD23             ,
246:D        1  CMDset_length     , CMD25             , CMD26             , CMD27             ,
247:D        1  CMD28             , CMD29             , CMD30             , CMD31             ,
248:D        1  CMDset_unit_0     , CMDset_unit_1     , CMDset_unit_2     , CMDset_unit_3     ,
249:D        1  CMDset_unit_4     , CMDset_unit_5     , CMDset_unit_6     , CMDset_unit_7     ,
250:D        1  CMDset_unit_8     , CMDset_unit_9     , CMDset_unit_10    , CMDset_unit_11    ,
251:D        1  CMDset_unit_12    , CMDset_unit_13    , CMDset_unit_14    , CMDset_unit_15    ,
252:D        1  CMDinit_util_NEM  , CMDinit_util_REM  , CMDinit_util_SEM  , CMDinit_diagnostic,
253:D        1  CMDno_op          , CMDdescribe       , CMD54             , CMDinit_media     ,
254:D        1  CMDset_options    , CMDset_rps        , CMDset_retry_time , CMDset_release    ,
255:D        1  CMDset_burst_LBO  , CMDset_burst_ABT  , CMDset_status_mask, CMD63             ,
256:D        1  CMDset_vol_0      , CMDset_vol_1      , CMDset_vol_2      , CMDset_vol_3      ,
257:D        1  CMDset_vol_4      , CMDset_vol_5      , CMDset_vol_6      , CMDset_vol_7      ,
258:D        1  CMDset_retadd_mode, CMDwrite_file_mark, CMDunload         , CMD75             ,
259:D        1  CMD76             , CMD77             , CMD78             , CMD79             ,
260:D        1  CMD80             , CMD81             , CMD82             , CMD83             ,
261:D        1  CMD84             , CMD85             , CMD86             , CMD87             ,
262:D        1  CMD88             , CMD89             , CMD90             , CMD91             ,
263:D        1  CMD92             , CMD93             , CMD94             , CMD95             ,
264:D        1  CMD96             , CMD97             , CMD98             , CMD99             ,
265:D        1  CMD100            , CMD101            , CMD102            , CMD103            ,
266:D        1  CMD104            , CMD105            , CMD106            , CMD107            ,
267:D        1  CMD108            , CMD109            , CMD110            , CMD111            ,
268:D        1  CMD112            , CMD113            , CMD114            , CMD115            ,
269:D        1  CMD116            , CMD117            , CMD118            , CMD119            ,
270:D        1  CMD120            , CMD121            , CMD122            , CMD123            ,
271:D        1  CMD124            , CMD125            , CMD126            , CMD127            ,
272:D        1  CMD128 {the field width is forced to 8 bits for packing considerations}  );
273:D        1
274:D        1   const
275:D        1     transparent_sec = 18;
276:D        1     command_sec = 5;
277:D        1     execution_sec = 14;
278:D        1     reporting_sec = 16;
```



```
279:D        1 $page$
280:D        1
281:D        1     errorbits_owning_parmfield = (errorbits which set the parameter field)
282:D        1       [
283:D        1                 {REJECT ERRORS FIELD}
284:D        1         ( 0)  eb0,                                  {unknown, but assumed}
285:D        1         ( 1)  eb1,                                  {unknown, but assumed}
286:D        1         ( 3)  eb3,                                  {unknown, but assumed}
287:D        1         ( 4)  eb4,                                  {unknown, but assumed}
288:D        1         (11)  eb11,                                 {unknown, but assumed}
289:D        1         (13)  eb13,                                 {unknown, but assumed}
290:D        1         (14)  eb14,                                 {unknown, but assumed}
291:D        1         (15)  eb15,                                 {unknown, but assumed}
292:D        1                 {FAULT ERRORS FIELD}
293:D        1         (16)  eb16,                                 {unknown, but assumed}
294:D        1         (17)  cross_unit,
295:D        1         (18)  eb18,                                 {unknown, but assumed}
296:D        1         (20)  eb20,                                 {unknown, but assumed}
297:D        1         (21)  eb21,                                 {unknown, but assumed}
298:D        1         (23)  eb23,                                 {unknown, but assumed}
299:D        1         (24)  diagnostic_result,
300:D        1         (25)  eb25,                                 {unknown, but assumed}
301:D        1         (29)  eb29,                                 {unknown, but assumed}
302:D        1                 {ACCESS ERRORS FIELD}
303:D        1         (38)  eb38,                                 {unknown, but assumed}
304:D        1         (39)  eb39,                                 {unknown, but assumed}
305:D        1         (41)  unrecoverable_data,
306:D        1         (42)  eb42,                                 {unknown, but assumed}
307:D        1         (45)  eb45,                                 {unknown, but assumed}
308:D        1         (46)  eb46,                                 {unknown, but assumed}
309:D        1         (47)  eb47,                                 {unknown, but assumed}
310:D        1                 {INFORMATION ERRORS FIELD}
311:D        1         (48)  operator_request,
312:D        1         (49)  diagnostic_request,
313:D        1         (50)  internal_maintenance_request,
314:D        1         (53)  eb53,                                 {unknown, but assumed}
315:D        1         (54)  eb54,                                 {unknown, but assumed}
316:D        1         (56)  eb56,                                 {unknown, but assumed}
317:D        1         (58)  marginal_data,
318:D        1         (59)  recoverable_data,
319:D        1         (60)  eb60,                                 {unknown, but assumed}
320:D        1         (62)  eb62,                                 {unknown, but assumed}
321:D        1         (63)  eb63                                  {unknown, but assumed}
322:D        1       ];
323:D        1
324:S
325:D        1     errorbits_requesting_release = (errorbits which request release)
326:D        1       [
327:D        1         (48)  operator_request,
328:D        1         (49)  diagnostic_request,
329:D        1         (50)  internal_maintenance_request
330:D        1       ];
```

```
   331:D        1 $page$
   332:D        1
   333:S             {
   334:S               NOTE: the following functions each perform a COMPLETE transaction. They:
   335:S                     .  issue a (device or transparent) command          (Command message)
   336:S                     .  transfer data if applicable                      (Execution message)
   337:S                     .  return the resulting QSTAT                       (Reporting message)
   338:D        1   }
   339:D        1   function chan_indep_clr  (uep: uep_type): unsgn8;
   340:D        1   function set_unit        (uep: uep_type; unit: unsgn4): unsgn8;
   341:D        1   function set_unitvol     (uep: uep_type): unsgn8;
   342:D        1   function status          (uep: uep_type; var status_bytes: status_type): unsgn8;
   343:D        1   function release         (uep: uep_type; unit: unsgn4): unsgn8;
   344:D        1   function describe        (uep: uep_type; var describe_bytes: describe_type): unsgn8;
   345:D        1   function set_release     (uep: uep_type; t: t_type; z: z_type): unsgn8;
   346:D        1   function set_options     (uep: uep_type; options_byte: unsgn8): unsgn8;
   347:D        1   function set_status_mask (uep: uep_type; status_mask: status_mask_type): unsgn8;
   348:S
   349:S
   350:S             {
   351:S               NOTE: The following routines do not, in themselves, perform a complete
   352:S                     transaction. They provide some of the messages necessary for
   353:S                     transactions which are broken apart to allow overlapped transfers.
   354:D        1   }
   355:D        1   procedure ICuvalc (uep: uep_type; address, len: integer; cmd: CMD_type);
   356:D        1   function qstat    (uep: uep_type): unsgn8;
   357:S
   358:D        1 implement (CS80)
   359:S
   360:S
   361:D        1 var
   362:D   -20  1   most_recent_status: status_type;  {for post-mortem diagnostic purposes only!!!}
   363:S
   364:S
   365:D        1 function qstat(uep: uep_type): unsgn8;
   366:S             {
   367:S               receive a REPORTING message
   368:S               return the QSTAT byte
   369:D        2   }
   370:D        2   var
   371:D        2     qstat_byte: {the 1 byte in the reporting message}
   372:D        2       packed record
   373:D        2         b: unsgn8;
   374:D    -2  2       end;
   375:C        2   begin {qstat}
   376:C        2     HPIBshort_msge_in(uep, reporting_sec, addr(qstat_byte), sizeof(qstat_byte));
   377:C        2     qstat := qstat_byte.b;
   378:C        2   end; {qstat}
```



```
   379:D   -20  1 $page$
   380:S
   381:D        1 function chan_indep_clr(uep: uep_type): unsgn8;
   382:S             {
   383:S               issue the CHANNEL_INDEPENDENT_CLEAR command
   384:S               return the QSTAT byte
   385:D        2   }
   386:D        2   var
   387:D        2     cic: {the 2 bytes in the channel independent clear command message}
   388:D        2       packed record
   389:D        2         setunit: CMD_type;
   390:D        2         ci_clr: unsgn8;
   391:D    -2  2       end;
   392:C        2   begin {chan_indep_clr}
   393:C        2     cic.setunit := CMD_type(signed16(CMDset_unit_0)+uep^.du);
   394:C        2     cic.ci_clr := 8;
   395:C        2     HPIBshort_msge_out(uep, transparent_sec, addr(cic), sizeof(cic));
   396:C        2     HPIBwait_for_ppol(uep);
   397:C        2     chan_indep_clr := qstat(uep);
   398:C        2   end; {chan_indep_clr}
   399:S
   400:S
   401:D        1 procedure ICc(uep: uep_type; cmd: CMD_type);
   402:S             {
   403:S               issue the specified command
   404:D        2   }
   405:D        2   var
   406:D        2     c: {the 1-byte command message}
   407:D        2       packed record
   408:D        2         cmd: CMD_type;
   409:D    -2  2       end;
   410:C        2   begin {ICc}
   411:C        2     c.cmd := cmd;
   412:C        2     HPIBshort_msge_out(uep, command_sec, addr(c), sizeof(c));
   413:C        2   end; {ICc}
   414:D   -20  1
   415:D   -20  1
   416:D        1 procedure ICuc(uep: uep_type; unit: unsgn4; cmd: CMD_type);
   417:S             {
   418:S               issue the specified SET_UNIT & command
   419:D        2   }
   420:D        2   var
   421:D        2     uc: {the 2-byte command message}
   422:D        2       packed record
   423:D        2         setunit: CMD_type;
   424:D        2         cmd: CMD_type;
   425:D    -2  2       end;
   426:C        2   begin {ICuc}
   427:C        2     uc.setunit := CMD_type(signed16(CMDset_unit_0)+unit);
   428:C        2     uc.cmd := cmd;
   429:C        2     HPIBshort_msge_out(uep, command_sec, addr(uc), sizeof(uc));
   430:C        2   end; {ICuc}
```

```
   431:D   -20  1 $page$
   432:D   -20  1
   433:D        1 function set_unit(uep: uep_type; unit: unsgn4): unsgn8;
   434:S          {
   435:S            issue the SET_UNIT command
   436:S            return the QSTAT byte
   437:D        2   }
   438:C        2   begin {set_unit}
   439:C        2     ICc(uep, CMD_type(signed16(CMDset_unit_0)+unit));
   440:C        2     HPIBwait_for_ppol(uep);
   441:C        2     set_unit := qstat(uep);
   442:C        2   end; {set_unit}
   443:S
   444:S
   445:D        1 function set_unitvol(uep: uep_type): unsgn8;
   446:S          {
   447:S            issue the SET_UNIT & SET_VOLUME commands
   448:S            return the QSTAT byte
   449:D        2   }
   450:C        2   begin {set_unitvol}
   451:C        2     ICuc(uep, uep^.du, CMD_type(signed16(CMDset_vol_0)+uep^.dv));
   452:C        2     HPIBwait_for_ppol(uep);
   453:C        2     set_unitvol := qstat(uep);
   454:C        2   end; {set_unitvol}
   455:S
   456:S
   457:D        1 function status(uep: uep_type; var status_bytes: status_type): unsgn8;
   458:S          {
   459:S            issue the REQUEST_STATUS command
   460:S            place the 20 bytes of status in the passed variable 'status_bytes'
   461:S            return the QSTAT byte
   462:D        2   }
   463:C        2   begin {status}
   464:C        2     ICc(uep, CMDrequest_status);
   465:C        2     HPIBwait_for_ppol(uep);
   466:C        2     HPIBshort_msge_in(uep, execution_sec, addr(status_bytes), sizeof(status_bytes));
   467:C        2     most_recent_status := status_bytes;  {for post-mortem diagnostic purposes only!!!}
   468:C        2     HPIBwait_for_ppol(uep);
   469:C        2     status := qstat(uep);
   470:C        2   end; {status}
   471:S
   472:D   -20  1
   473:D        1 function release(uep: uep_type; unit: unsgn4): unsgn8;
   474:S          {
   475:S            SET_UNIT & issue the RELEASE command
   476:S            return the QSTAT byte
   477:D        2   }
   478:C        2   begin {release}
   479:C        2     ICuc(uep, unit, CMDrelease);
   480:C        2     HPIBwait_for_ppol(uep);
   481:C        2     release := qstat(uep);
   482:C        2   end; {release}
```



```
   483:D   -20  1 $page$
   484:S
   485:D        1 function describe(uep: uep_type; var describe_bytes: describe_type): unsgn8;
   486:S          {
   487:S            issue the DESCRIBE command
   488:S            place the 37 bytes of description in the passed variable 'describe_bytes'
   489:S            return the QSTAT byte
   490:D        2   }
   491:C        2   begin {describe}
   492:C        2     ICc(uep, CMDdescribe);
   493:C        2     HPIBwait_for_ppol(uep);
   494:C        2     HPIBshort_msge_in(uep, execution_sec, addr(describe_bytes), sizeof(describe_bytes));
   495:C        2     HPIBwait_for_ppol(uep);
   496:C        2     describe := qstat(uep);
   497:C        2   end; {describe}
   498:S
   499:S
   500:D        1 function set_release(uep: uep_type; t: t_type; z: z_type): unsgn8;
   501:D        2   var
   502:D        2     sr: {the 3 bytes in the SET_UNIT & SET_RELEASE command message}
   503:D        2       packed record
   504:D        2         setunit: CMD_type;
   505:D        2         setrel: CMD_type;
   506:D        2         Tbit: t_type;
   507:D        2         Zbit: z_type;
   508:D        2         pad: 0..63;
   509:D    -4  2       end;
   510:C        2   begin {set_release}
   511:C        2     sr.setunit := CMDset_unit_15;  {always addressed to the controller}
   512:C        2     sr.setrel  := CMDset_release;
   513:C        2     sr.Tbit    := t;
   514:C        2     sr.Zbit    := z;
   515:C        2     sr.pad     := 0;
   516:C        2     HPIBshort_msge_out(uep, command_sec, addr(sr), sizeof(sr));
   517:C        2     HPIBwait_for_ppol(uep);
   518:C        2     set_release := qstat(uep);
   519:C        2   end; {set_release}
   520:S
   521:S
   522:D        1 function set_options(uep: uep_type; options_byte: unsgn8): unsgn8;
   523:D        2   var
   524:D        2     so: {the 2 bytes in the SET_OPTIONS command message}
   525:D        2       packed record
   526:D        2         setoptn: CMD_type;
   527:D        2         ob: unsgn8;
   528:D    -2  2       end;
   529:C        2   begin {set_options}
   530:C        2     so.setoptn := CMDset_options;
   531:C        2     so.ob      := options_byte;
   532:C        2     HPIBshort_msge_out(uep, command_sec, addr(so), sizeof(so));
   533:C        2     HPIBwait_for_ppol(uep);
   534:C        2     set_options := qstat(uep);
   535:C        2   end; {set_options}
```

```
   536:D   -20  1 $page$
   537:S
   538:D    -8  1 function set_status_mask(uep: uep_type; status_mask: status_mask_type): unsgn8;
   539:D    -8  2   var
   540:D    -8  2     ssm: {the 10 bytes in the SET_STATUS_MASK command message}
   541:D    -8  2       packed record
   542:D    -8  2         nop: CMD_type;
   543:D    -8  2         setstsmsk: CMD_type;
   544:D    -8  2         stsmsk: status_mask_type;
   545:D   -18  2       end;
   546:C        2   begin {set_status_mask}
   547:C        2     ssm.nop       := CMDno_op;
   548:C        2     ssm.setstsmsk := CMDset_status_mask;
   549:C        2     ssm.stsmsk    := status_mask;
   550:C        2     HPIBshort_msge_out(uep, command_sec, addr(ssm), sizeof(ssm));
   551:C        2     HPIBwait_for_ppol(uep);
   552:C        2     set_status_mask := qstat(uep);
   553:C        2   end; {set_status_mask}
   554:S
   555:S
   556:D        1 procedure ICuvalc(uep: uep_type; address, len: integer; cmd: CMD_type);
   557:S          {
   558:S            issue the following command sequence:
   559:S              .  SET_UNIT                (u)
   560:S              .  SET_VOLUME              (v)
   561:S              .  SET_ADDRESS             (a)
   562:S              .  SET_LENGTH              (l)
   563:S              .  specified COMMAND       (c)
   564:D        2   }
   565:D        2   var
   566:D        2     uvalc: {the 17 bytes in the command message}
   567:D        2       packed record
   568:D        2         setunit: CMD_type;       {set unit}
   569:D        2         setvol: CMD_type;        {set volume}
   570:D        2         nop1: CMD_type;          {nop}
   571:D        2         setadd: CMD_type;        {set address}
   572:D        2         sva: sva_type;           {single vector address}
   573:D        2         nop2: CMD_type;          {nop}
   574:D        2         setlen: CMD_type;        {set length}
   575:D        2         length: integer;         {length}
   576:D        2         cmd: CMD_type;           {specified command}
   577:D   -18  2       end;
   578:C        2   begin {ICuvalc}
   579:C        2     uvalc.setunit := CMD_type(signed16(CMDset_unit_0)+uep^.du);
   580:C        2     uvalc.setvol  := CMD_type(signed16(CMDset_vol_0)+uep^.dv);
   581:C        2     uvalc.nop1    := CMDno_op;
   582:C        2     uvalc.setadd  := CMDset_address_1V;
   583:C        2     uvalc.sva.utb := 0;
   584:C        2     uvalc.sva.lfb := address;
   585:C        2     uvalc.nop2    := CMDno_op;
   586:C        2     uvalc.setlen  := CMDset_length;
   587:C        2     uvalc.length  := len;
   588:C        2     uvalc.cmd     := cmd;
   589:C        2     HPIBshort_msge_out(uep, command_sec, addr(uvalc), sizeof(uvalc));
   590:C        2   end; {ICuvalc}
   591:S
   592:C        1 end; {CS80}
```



```
   593:D        1 $page$
   594:S
   595:D        1 module CS80dsr; {Command Set '80 Driver Support Routines}
   596:S
   597:D        1 import
   598:D        1   sysglobals, bkgnd, tapebuf, CS80;
   599:S
   600:D        1 export
   601:S
   602:D        1   procedure invalidate_stateinfo(uep: uep_type);
   603:S
   604:D        1   procedure hardle_bad_status(uep: uep_type; ok_to_config: boolean; var retry_required: boolean);
   605:D        1
   606:D        1   procedure configure(uep: uep_type);
   607:S
   608:D        1 implement {CS80dsr}
   609:S
   610:S
   611:D        1 procedure invalidate_stateinfo(uep: uep_type);
   612:D        2   var
   613:D    -2  2     lun: unitnum;
   614:D    -6  2     scanner_uep: uep_type;
   615:C        2   begin {invalidate_stateinfo}
   616:C        2     for lun := 1 to maxunit do
   617:C        3       begin
   618:C        3         scanner_uep := addr(unitable^[lun]);
   619:C        3         if (scanner_uep^.letter='D') and
   620:C        4            (scanner_uep^.sc = uep^.sc) and
   621:C        4            (scanner_uep^.ba = uep^.ba) and
   622:C        4            (scanner_uep^.du = uep^.du)
   623:C        4            {don't qualify dv because all volumes ARE affected!}
   624:C        4         then  {invalidate all CS80 state info!}
   625:C        4           begin
   626:C        4             scanner_uep^.umediavalid := false;  {media possibly changed}
   627:C        4             scanner_uep^.dvrtemp2 := -1;        {block size possibly changed!}
   628:C        4             if scanner_uep=tapebuf_uep then
   629:C        5               tapebuf_state := undefined;
   630:C        4           end; {if}
   631:C        3       end; {for}
   632:C        2   end; {invalidate_stateinfo}
```

```
633:D        1 $page$
634:S
635:D        1 procedure handle_bad_status(uep: uep_type; ok_to_config: boolean; var retry_required: boolean);
636:D        2
637:D        2   var
638:D     -2 2     iorval_to_report: iorsltwd;  {to hold the first reportable error}
639:D     -4 2     working_iorval: iorsltwd;  {cleared each time status is read}
640:D    -24 2     status_bytes: status_type;
641:D    -28 2     eb_scan, parameter_field_owner: errorbit_type;
642:D    -29 2     reconfiguration_needed: boolean;
643:D    -29 2
644:C        2   begin {handle_bad_status}
645:C        2
646:C        2     iorval_to_report := inoerror;
647:C        2
648:C        2     repeat
649:C        3
650:C        3       if status(uep, status_bytes)<>0 then
651:C        4         ioresc_bkgnd(uep, zbadhardware);
652:C        3
653:C        3       working_iorval := inoerror;
654:C        3       parameter_field_owner := channel_parity_error;  {doesn't REALLY own it!}
655:C        3       reconfiguration_needed := false;
656:C        3
657:C        3       for eb_scan := eb63 downto eb0 do
658:C        4         if status_bytes.errorbits[eb_scan] then
659:C        5           begin
660:S
661:C        5             if eb_scan in errorbits_owning_parmfield then
662:C        6               parameter_field_owner := eb_scan;
663:S
664:C        5             case eb_scan of
665:S
666:C        6               {specific fatal errors}
667:C        6                 channel_parity_error,
668:C        6                 controller_fault,
669:C        6                 unit_fault,
670:C        6                 diagnostic_result:
671:C        6                     working_iorval := zbadhardware;
672:C        6                 illegal_opcode,
673:C        6                 parameter_bounds,
674:C        6                 illegal_parameter:
675:C        6                     working_iorval := zbadmode;  {some cmds optional in SS/80}
676:C        6                 module_addressing:
677:C        6                     working_iorval := znodevice;
678:C        6                 address_bounds,
679:C        6                 end_of_volume:
680:C        6                     working_iorval := znosuchblk;
681:C        6                 uninitialized_media:
682:C        6                     if status_bytes.errorbits[power_fail]
683:C        7                       then {probably an uncertified tape; allow access anyway}
684:C        7                       else working_iorval := zuninitialized;
685:C        6                 no_spares_available:
686:C        6                     working_iorval := zinitfail;
687:C        6                 not_ready:
688:C        6                     working_iorval := znotready;
689:C        6                 write_protect:
690:C        6                     working_iorval := zprotected;
691:C        6                 no_data_found,
692:C        6                 end_of_file:
```



```
693:C        6                     working_iorval := znoblock;
694:C        6                 unrecoverable_data_overflow,
695:C        6                 unrecoverable_data:
696:C        6                     working_iorval := zbadblock;
697:S
698:C        6               {power fail}
699:C        6                 power_fail:
700:C        6                     begin
701:C        6                       invalidate_stateinfo(uep);
702:C        6                       if uep^.ureportchange then
703:C        7                         working_iorval := zmediumchanged;
704:C        6                       reconfiguration_needed := true;
705:C        6                       retry_required := true;
706:C        6                     end;
707:S
708:C        6               {retryable errors}
709:C        6                 operator_release_required,
710:C        6                 diagnostic_release_required,
711:C        6                 internal_maintenance_required,
712:C        6                 retransmit:
713:C        6                     retry_required := true;
714:S
715:C        6               {errors indicating release requested}
716:C        6                 operator_request,
717:C        6                 diagnostic_request,
718:C        6                 internal_maintenance_request:
719:C        6                     {do nothing here; release below if parmeter field owned};
720:S
721:C        6               {errors indicating reconfiguration needed}
722:C        6                 media_wear,                    {supposed to be masked out}
723:C        6                 latency_induced,               {supposed to be masked out}
724:C        6                 eb53,                          {supposed to be masked out}
725:C        6                 eb54,                          {supposed to be masked out}
726:C        6                 auto_sparing_invoked,          {supposed to be masked out}
727:C        6                 eb56,                          {supposed to be masked out}
728:C        6                 recoverable_data_overflow,     {supposed to be masked out}
729:C        6                 marginal_data,                 {supposed to be masked out}
730:C        6                 recoverable_data,              {supposed to be masked out}
731:C        6                 eb60,                          {supposed to be masked out}
732:C        6                 maintenance_track_overflow,    {supposed to be masked out}
733:C        6                 eb62,                          {supposed to be masked out}
734:C        6                 eb63:                          {supposed to be masked out}
735:C        6                     reconfiguration_needed := true;
736:S
737:C        6               {errors not covered by the above cases}
738:C        6                 otherwise
739:S                               { specifically including:
740:S                                 message_sequence,
741:S                                 message_length,
742:S                                 cross_unit,
743:C        6                        illegal_parallel_operation }
744:C        6                   working_iorval := zcatchall;
745:S
746:C        6             end; {case}
747:S
748:C        5           end; {if}
```

```
 749:C        3 $page$
 750:S
 751:C        3       if iorval_to_report=inoerror then  {none previously found; report this one}
 752:C        4         iorval_to_report := working_iorval;  {it can be inoerror also!}
 753:C        3
 754:C        3       if parameter_field_owner in errorbits_requesting_release then
 755:C        4         if not (status_bytes.urr[1] in [0..15]) then
 756:C        5           ioresc_bkgnd(uep, zcatchall)
 757:C        5         else if release(uep, status_bytes.urr[1])<>0 then
 758:C        6           {handle the bad qstat elsewhere; worry not, the device won't forget!};
 759:S
 760:C        3       if reconfiguration_needed and ok_to_config then
 761:C        4         configure(uep);
 762:C        3
 763:C        3     until set_unit(uep, status_bytes.current_vu.uuuu)=0;  {restore original command unit}
 764:C        2
 765:C        2     if iorval_to_report<>inoerror then
 766:C        3       ioresc_bkgnd(uep, iorval_to_report);
 767:C        2
 768:C        2   end; {handle_bad_status}
 769:S
 770:S
 771:D        1 procedure configure(uep: uep_type);
 772:D        2
 773:D        2   var
 774:D    -1  2     escape_caught: boolean;
 775:D    -2  2     saved_ureportchange: boolean;
 776:D    -3  2     retry_required: boolean;
 777:D   -40  2     describe_bytes: describe_type;
 778:D   -40  2     bcd_prod_num: {within the describe bytes}
 779:D   -40  2       packed record case integer of
 780:D   -40  2         0: (dn:  unsgn24);
 781:D   -40  2         1: (bcd: packed array[1..6] of unsgn4);
 782:D   -44  2       end;
 783:D   -44  2     fixed_volume_byte:
 784:D   -44  2       packed record case integer of
 785:D   -44  2         0: (b: unsgn8);
 786:D   -44  2         1: (bit: packed array[0..7] of boolean);
 787:D   -46  2       end;
 788:D   -48  2     prod_num: signed16;
 789:D   -50  2     index: signed16;
 790:D   -50  2
 791:D   -50  2   const
 792:D   -50  2     masked = true;
 793:D   -50  2     unmasked = false;
 794:D   -50  2
 795:D   -50  2     my_status_mask = status_mask_type
 796:D   -50  2       [
 797:D   -50  2                        {REJECT ERRORS FIELD}
 798:D   -50  2       { 0  eb0:                              }  unmasked,
 799:D   -50  2       { 1  eb1:                              }  unmasked,
 800:D   -50  2       { 2  channel_parity_error:             }  unmasked,
 801:D   -50  2       { 3  eb3:                              }  unmasked,
 802:D   -50  2       { 4  eb4:                              }  unmasked,
 803:D   -50  2       { 5  illegal_opcode:                   }  unmasked,
 804:D   -50  2       { 6  module_addressing:                }  unmasked,
 805:D   -50  2       { 7  address_bounds:                   }  unmasked,
 806:D   -50  2       { 8  parameter_bounds:                 }  unmasked,
 807:D   -50  2       { 9  illegal_parameter:                }  unmasked,
 808:D   -50  2       {10  message_sequence:                 }  unmasked,
```



```
 809:D   -50  2       {11  eb11:                             }  unmasked,
 810:D   -50  2       {12  message_length:                   }  unmasked,
 811:D   -50  2       {13  eb13:                             }  unmasked,
 812:D   -50  2       {14  eb14:                             }  unmasked,
 813:D   -50  2       {15  eb15:                             }  unmasked,
 814:D   -50  2                        {FAULT ERRORS FIELD}
 815:D   -50  2       {16  eb16:                             }  unmasked,   {unmaskable error}
 816:D   -50  2       {17  cross_unit:                       }  unmasked,   {unmaskable error}
 817:D   -50  2       {18  eb18:                             }  unmasked,   {unmaskable error}
 818:D   -50  2       {19  controller_fault:                 }  unmasked,   {unmaskable error}
 819:D   -50  2       {20  eb20:                             }  unmasked,   {unmaskable error}
 820:D   -50  2       {21  eb21:                             }  unmasked,   {unmaskable error}
 821:D   -50  2       {22  unit_fault:                       }  unmasked,   {unmaskable error}
 822:D   -50  2       {23  eb23:                             }  unmasked,   {unmaskable error}
 823:D   -50  2       {24  diagnostic_result:                }  unmasked,   {unmaskable error}
 824:D   -50  2       {25  eb25:                             }  unmasked,   {unmaskable error}
 825:D   -50  2       {26  operator_release_required:        }  unmasked,   {unmaskable error}
 826:D   -50  2       {27  diagnostic_release_required:      }  unmasked,   {unmaskable error}
 827:D   -50  2       {28  internal_maintenance_required:    }  unmasked,   {unmaskable error}
 828:D   -50  2       {29  eb29:                             }  unmasked,   {unmaskable error}
 829:D   -50  2       {30  power_fail:                       }  unmasked,   {unmaskable error}
 830:D   -50  2       {31  retransmit:                       }  unmasked,   {unmaskable error}
 831:D   -50  2                        {ACCESS ERRORS FIELD}
 832:D   -50  2       {32  illegal_parallel_operatior:       }  unmasked,
 833:D   -50  2       {33  uninitialized_media:              }  unmasked,
 834:D   -50  2       {34  no_spares_available:              }  unmasked,
 835:D   -50  2       {35  not_ready:                        }  unmasked,
 836:D   -50  2       {36  write_protect:                    }  unmasked,
 837:D   -50  2       {37  no_data_found:                    }  unmasked,
 838:D   -50  2       {38  eb38:                             }  unmasked,
 839:D   -50  2       {39  eb39:                             }  unmasked,
 840:D   -50  2       {40  unrecoverable_data_overflow:      }  unmasked,
 841:D   -50  2       {41  unrecoverable_data:               }  unmasked,
 842:D   -50  2       {42  eb42:                             }  unmasked,
 843:D   -50  2       {43  end_of_file:                      }  unmasked,
 844:D   -50  2       {44  end_of_volume:                    }  unmasked,
 845:D   -50  2       {45  eb45:                             }  unmasked,
 846:D   -50  2       {46  eb46:                             }  unmasked,
 847:D   -50  2       {47  eb47:                             }  unmasked,
 848:D   -50  2                        {INFORMATION ERRORS FIELD}
 849:D   -50  2       {48  operator_request:                 }  unmasked,
 850:D   -50  2       {49  diagnostic_request:               }  unmasked,
 851:D   -50  2       {50  internal_maintenance_request:     }  unmasked,
 852:D   -50  2       {51  media_wear:                       }  masked,
 853:D   -50  2       {52  latency_induced:                  }  masked,
 854:D   -50  2       {53  eb53:                             }  masked,
 855:D   -50  2       {54  eb54:                             }  masked,
 856:D   -50  2       {55  auto_sparing_invoked:             }  masked,
 857:D   -50  2       {56  eb56:                             }  masked,
 858:D   -50  2       {57  recoverable_data_overflow:        }  masked,
 859:D   -50  2       {58  marginal_data:                    }  masked,
 860:D   -50  2       {59  recoverable_data:                 }  masked,
 861:D   -50  2       {60  eb60:                             }  masked,
 862:D   -50  2       {61  maintenance_track_overflow:       }  masked,
 863:D   -50  2       {62  eb62:                             }  masked,
 864:D   -50  2       {63  eb63:                             }  masked
 865:D   -50  2       ];
```

```
 866 D   -50  2 $page$
 867 D   -50  2
 868 C        2   begin {configure}
 869 S
 870 C        2     with uep^ do
 871 C        3       begin
 872 S
 873 C        3         escape_caught := false;
 874 C        3         saved_ureportchange := ureportchange;
 875 C        3         try
 876 C        4           ureportchange := false;  {NEVER report media change while in configure}
 877 S
 878 C        4           {configure the control unit}
 879 S
 880 C        4           repeat
 881 C        5             retry_required := false;
 882 C        5             if set_release(uep, allow_release_timeout, disable_auto_release)<>0 then
 883 C        6               handle_bad_status(uep, false, retry_required);
 884 C        5           until not retry_required;
 885 S
 886 C        4           repeat
 887 C        5             retry_required := false;
 888 C        5             if set_status_mask(uep, my_status_mask)<>0 then
 889 C        6               handle_bad_status(uep, false, retry_required);
 890 C        5           until not retry_required;
 891 S
 892 S
 893 C        4           {configure the required unit}
 894 S
 895 C        4           repeat
 896 C        5             retry_required := false;
 897 C        5             if chan_indep_clr(uep)<>0 then
 898 C        6               handle_bad_status(uep, false, retry_required);
 899 C        5           until not retry_required;
 900 S
 901 C        4           repeat
 902 C        5             retry_required := false;
 903 C        5             if set_unitvol(uep)<>0 then
 904 C        6               handle_bad_status(uep, false, retry_required);
 905 C        5           until not retry_required;
 906 S
 907 C        4           repeat
 908 C        5             retry_required := false;
 909 C        5             if describe(uep, describe_bytes)<>0 then
 910 C        6               handle_bad_status(uep, false, retry_required);
 911 C        5           until not retry_required;
 912 S
 913 C        4           with describe_bytes do
 914 C        5             begin
 915 S
 916 C        5               bcd_prod_num.dn := dn;
 917 C        5               prod_num := 0;
 918 C        5               for index := 1 to 5 do
 919 C        6                 prod_num := prod_num*10+bcd_prod_num.bcd[index];
 920 S
 921 C        5               if ( (devid<>prod_num) and (devid<>-1) )  {wrong product number}
 922 C        6                  or ( dt<0 )                            {can't detect media change}
 923 C        6                 then ioresc_bkgnd(uep, znodevice);
 924 S
 925 C        5               dvrtemp2 := 0;
```



```
 926 C        5               index := nbpb;
 927 C        5               while (index>0) and not odd(index) do
 928 C        6                 begin
 929 C        6                   dvrtemp2 := dvrtemp2+1;
 930 C        6                   index := index div 2;
 931 C        6                 end; {while}
 932 C        5               if index<>1 then  {blocksize isn't a power of 2!}
 933 C        6                 dvrtemp2 := -1; {don't panic; might be just no medium present}
 934 S
 935 C        5               fixed_volume_byte.b := fvb;  {fixed volume byte}
 936 C        5               uisfixed := fixed_volume_byte.bit[7-dv];
 937 S
 938 C        5               if devid=-1 then  {variable-sized removeable volume; set its size}
 939 C        6                 umaxbytes := (maxsvadd.lfb+1)*nbpb;
 940 S
 941 C        5               if dt=2 then  {it's a tape}
 942 C        6                 repeat  {enable auto-jump sparing}
 943 C        7                   retry_required := false;
 944 C        7                   if set_options(uep, 4)<>0 then
 945 C        8                     handle_bad_status(uep, false, retry_required);
 946 C        7                 until not retry_required;
 947 S
 948 C        5               repeat
 949 C        6                 retry_required := false;
 950 C        6                 if set_status_mask(uep, my_status_mask)<>0 then
 951 C        7                   handle_bad_status(uep, false, retry_required);
 952 C        6               until not retry_required;
 953 S
 954 C        5             end; {with}
 955 S
 956 C        4         recover
 957 C        4           escape_caught := true;
 958 C        3         ureportchange := saved_ureportchange;
 959 C        3         if escape_caught then
 960 C        4           escape(escapecode);
 961 S
 962 C        3       end; {with}
 963 S
 964 C        2   end; {configure}
 965 S
 966 C        1 end; {CS80dsr}
```

```
   967:D        1 $page$
   968:S
   969:D        1 module CS80dvr; {Command Set '80 Driver}
   970:S
   971:D        1 import
   972:D        1   sysglobals, asm, mini, drvasm, bkgnd, discHPIB, tapebuf, CS80, CS80dsr;
   973:S
   974:D        1 export
   975:D        1   type
   976:D        1     mp_type =  {media parameters}
   977:D        1       record
   978:D        1         tpm: integer;  {tracks per medium}
   979:D        1         bpt: integer;  {bytes per track}
   980:D        1       end;
   981:S
   982:D        1   procedure get_letter(uep: uep_type; ident: shortint; var letter: char);
   983:D        1
   984:D        1   procedure get_parms(var devtype: byte; var devid: integer;
   985:D        2                         var hardvols: shortint; var mp: mp_type);
   986:S
   987:D        1   procedure CS80io(fp: fibp; request: amrequesttype;
   988:D        2                      anyvar buffer: window; length, position: integer);
   989:S
   990:D        1 implement {CS80dvr}
   991:S
   992:D        1 var
   993:D    -2  1   CS80_devtype: byte;
   994:D    -6  1   CS80_devid: integer;
   995:D    -8  1   CS80_hardvols: shortint;
   996:D   -16  1   CS80_mp: mp_type;
   997:S
   998:S
   999:D        1 procedure clear_unit(uep: uep_type);
  1000:C        2   begin {clear_unit}
  1001:C        2     try
  1002:C        3       allocate_bkgnd_info(uep);
  1003:C        3       HPIBcheck_sc(uep);
  1004:C        3       if HPIBamigo_identify(uep) div 256<>2 then
  1005:C        4         ioresc_bkgnd(uep, znodevice);
  1006:C        3       configure(uep);
  1007:C        3       deallocate_bkgnd_info(uep);
  1008:C        3     recover
  1009:C        3       abort_bkgnd_process(uep);
  1010:C        2   end; {clear_unit}
  1011:S
  1012:S
  1013:S          {
  1014:S            procedures for CTABLE self-configuration
  1015:D   -16  1 }
  1016:S
  1017:D        1 procedure get_parms(var devtype: byte; var devid: integer;
  1018:D        2                       var hardvols: shortint; var mp: mp_type);
  1019:C        2   begin {get_parms}
  1020:C        2     devtype := CS80_devtype;
  1021:C        2     devid := CS80_devid;
  1022:C        2     hardvols := CS80_hardvols;
  1023:C        2     mp := CS80_mp;
  1024:C        2   end; {get_parms}
```



```
  1025:D   -16  1 $page$
  1026:S
  1027:D        1 procedure get_letter(uep: uep_type; ident: shortint; var letter: char);
  1028:D        2   var
  1029:D    -1  2     retry_required: boolean;
  1030:D   -38  2     describe_bytes: describe_type;
  1031:D   -38  2     bcd_prod_num: {within the describe bytes}
  1032:D   -38  2       packed record case integer of
  1033:D   -38  2         0: (dn:  unsgn24);
  1034:D   -38  2         1: (bcd: packed array[1..6] of unsgn4);
  1035:D   -42  2       end;
  1036:D   -44  2     index: signed16;
  1037:D   -44  2     volumes_byte: {with the describe bytes}
  1038:D   -44  2       packed record case integer of
  1039:D   -44  2         0: (vb:  unsgn8);
  1040:D   -44  2         1: (bools: packed array[0..7] of boolean);
  1041:D   -46  2       end;
  1042:C        2   begin {get_letter}
  1043:C        2     uep^.ureportchange := false;  {don't report media changes/power-on now!!!}
  1044:S
  1045:C        2     repeat  {cmd w/o execution msge avoids escape if in power-on holdoff!}
  1046:C        3       retry_required := false;
  1047:C        3       if set_unitvol(uep)<>0 then
  1048:C        4         handle_bad_status(uep, false, retry_required);  {don't configure!!!}
  1049:C        3     until not retry_required;
  1050:S
  1051:C        2     repeat
  1052:C        3       retry_required := false;
  1053:C        3       if describe(uep, describe_bytes)<>0 then
  1054:C        4         handle_bad_status(uep, false, retry_required);  {don't configure!!!}
  1055:C        3     until not retry_required;
  1056:S
  1057:C        2     with describe_bytes do
  1058:C        3       begin
  1059:C        3         CS80_devtype := dt;
  1060:S
  1061:C        3         bcd_prod_num.dn := dn;
  1062:C        3         CS80_devid := 0;
  1063:C        3         for index := 1 to 5 do
  1064:C        4           CS80_devid := CS80_devid*10+bcd_prod_num.bcd[index];
  1065:S
  1066:C        3         volumes_byte.vb := fvb+rvb;
  1067:C        3         CS80_hardvols := 0;
  1068:C        3         for index := 0 to 7 do
  1069:C        4           CS80_hardvols := CS80_hardvols+ord(volumes_byte.bools[index]);
  1070:S
  1071:C        3         with CS80_mp do
  1072:C        4           begin
  1073:C        4             tpm := (maxcadd+1)*(maxhadd+1);   {tracks per medium}
  1074:C        4             if tpm=1  {only single-vector addressing info given}
  1075:C        5               then bpt := (maxsvadd.lfb+1)*nbpb    {bytes per track}
  1076:C        5               else bpt := (maxsadd+1)*nbpb;        {bytes per track}
  1077:C        4           end; {with}
  1078:S
  1079:C        3       end; {with}
  1080:C        2     letter := 'Q';
  1081:C        2   end; {get_letter}
```

```
1082:D   -16  1 $page$
1083:S
1084:S          {
1085:S            low-level read/write routines
1086:D   -16  1 }
1087:S
1088:D        1 procedure flagit(uep: anyptr);
1089:C        2   begin {flagit}
1090:C        2     bip_type(uep_type(uep)^.dvrtemp)^.xfr_chain_semaphore := false;
1091:C        2   end; {flagit}
1092:S
1093:S
1094:D        1 procedure xfr(uep: uep_type; request: amrequesttype;
1095:D        2               bufptr: anyptr; block_address, length: integer);
1096:D        2   var
1097:D    -2  2     command: CMD_type;
1098:D    -3  2     retry_required: boolean;
1099:C        2   begin {xfr}
1100:C        2     allocate_bkgnd_info(uep);
1101:C        2     with bip_type(uep^.dvrtemp)^ do
1102:C        3       try
1103:C        4         if HPIBamigo_identify(uep) div 256<>2 then
1104:C        5           ioresc_bkgnd(uep, znodevice);
1105:C        4
1106:C        4         read_operation := (request=readbytes) or (request=startread);
1107:C        4         if read_operation
1108:C        5           then command := CMDlocate_and_read
1109:C        5           else command := CMDlocate_and_wrt;
1110:C        4         ICuvalc(uep, block_address, length, command);
1111:S
1112:C        4         if length>0 then
1113:C        5           begin
1114:C        5             HPIBwait_for_ppol(uep);
1115:C        5             xfr_chain_semaphore := true;  {merely a flag for xfr busy here}
1116:C        5             HPIBupon_dxfr_comp(uep, execution_sec, bufptr, length, flagit);
1117:C        5             while xfr_chain_semaphore do
1118:C        6               {nothing};
1119:C        5             if iores<>inoerror then escape(-10);
1120:C        5           end {if}
1121:C        5         else
1122:C        5           bdx_pre_eoi := false;
1123:S
1124:C        4         HPIBwait_for_ppol(uep);
1125:C        4         retry_required := false;
1126:C        4         if qstat(uep)<>0 then
1127:C        5           handle_bad_status(uep, true, retry_required)
1128:C        5         else if bdx_pre_eoi then
1129:C        6           ioresc_bkgnd(uep, zcatchall);  {unresolved premature eoi!}
1130:S
1131:C        4         deallocate_bkgnd_info(uep);
1132:C        4       recover
1133:C        4         abort_bkgnd_process(uep);
1134:C        2     ioresult := uep^.dvrtemp;
1135:C        2     if (ioresult<>ord(inoerror)) or retry_required then
1136:C        3       escape(-10);
1137:C        2   end; {xfr}
```



```
1138:D   -16  1 $page$
1139:S
1140:S          {
1141:S            tapebuf manipulation routines
1142:D   -16  1 }
1143:S
1144:D        1 procedure flush_tapebuf;
1145:D        2   var
1146:D    -1  2     escape_caught: boolean;
1147:D    -2  2     saved_ureportchange: boolean;
1148:C        2   begin {flush_tapebuf}
1149:C        2     if tapebuf_state=modified then
1150:C        3       with tapebuf_uep^ do
1151:C        4         begin
1152:C        4           escape_caught := false;
1153:C        4           saved_ureportchange := ureportchange;
1154:C        4           try
1155:C        5             ureportchange := true;  {don't flush out to different media!}
1156:C        5             tapebuf_state := undefined;  {while attempting the write}
1157:C        5             xfr(tapebuf_uep, writebytes, tapebuf_ptr, tapebuf_block, tapebuf_size);
1158:C        5             tapebuf_state := unmodified;  {write was successful!}
1159:C        5           recover
1160:C        5             escape_caught := true;
1161:C        4           ureportchange := saved_ureportchange;
1162:C        4           if escape_caught then
1163:C        5             escape(escapecode);
1164:C        4         end; {with}
1165:C        2   end; {flush_tapebuf}
1166:S
1167:S
1168:D        1 procedure load_tapebuf(uep: uep_type; request: amrequesttype; block: integer);
1169:D        2   var
1170:D    -1  2     xfr_required: boolean;
1171:C        2   begin {load_tapebuf}
1172:C        2     xfr_required := (tapebuf_uep<>uep) or (tapebuf_block<>block) or (tapebuf_state=undefined);
1173:C        2     if xfr_required then
1174:C        3       begin
1175:C        3         flush_tapebuf;
1176:C        3         tapebuf_uep := uep;
1177:C        3         tapebuf_block := block;
1178:C        3         tapebuf_state := undefined;
1179:C        3       end; {if}
1180:S
1181:C        2     if not xfr_required        {then confirm media present & unchanged}
1182:C        3       or (request=writebytes) {then confirm media not write protected}
1183:C        3       then
1184:C        3       begin
1185:C        3         xfr(tapebuf_uep, request, nil, tapebuf_block, 0);
1186:C        3         if tapebuf_state=undefined then
1187:C        4           xfr_required := true;
1188:C        3       end; {if}
1189:S
1190:C        2     if xfr_required then
1191:C        3       begin
1192:C        3         tapebuf_size := shifted_left(1, uep^.dvrtemp2);
1193:C        3         xfr(tapebuf_uep, readbytes, tapebuf_ptr, tapebuf_block, tapebuf_size);
1194:C        3         tapebuf_state := unmodified;  {read was successful!}
1195:C        3       end; {if}
1196:C        2   end; {load_tapebuf}
```

```
1197:D   -16  1 $page$
1198:S
1199:S          {
1200:S            read/write routine
1201:S
1202:S            The new Subset/80 devices coming out which support multiple block
1203:S            sizes have forced us to abandon the 2.X driver's essentially never-
1204:S            used asynchronous capabilities.  It's simply too difficult to handle
1205:S            the media change situation when the media's block size also changes.
1206:S            For instance, an asynchronous transfer started on the 256-byte block
1207:S            assumption might discover new media formatted to 1024-byte blocks, in
1208:S            which case entire transfer would need re-starting, this time using
1209:S            tapebuf for buffering.  This situation would not be detected until
1210:S            well into the asynchronous transfer chain, at which point we could
1211:S            be deadlocked.
1212:D   -16  1 }
1213:S
1214:D         1 procedure transfer(uep: uep_type; fp: fibp; request: amrequesttype;
1215:D         2                    bufptr: charptr; abs_position, length: integer);
1216:D         2   var
1217:D    -1   2     retry_required: boolean;
1218:D    -4   2     blockpower: shortint;
1219:D    -8   2     blocksize: integer;
1220:D   -20   2     block, intra_block_offset, partial_length: integer;
1221:S
1222:C         2   begin {transfer}
1223:S
1224:C         2     repeat
1225:C         3       retry_required := false;
1226:S
1227:C         3       try
1228:C         4         if uep^.dvrtemp2<0 then  {block size unknown; try to determine}
1229:C         5           begin
1230:C         5             clear_unit(uep);
1231:C         5             ioresult := uep^.dvrtemp;
1232:C         5             if ioresult<>ord(inoerror) then
1233:C         6               escape(-10);
1234:C         5           end; {if}
1235:S
1236:C         4         blockpower := uep^.dvrtemp2;
1237:C         4         if blockpower<0 then
1238:C         5           ioresc(znomedium);  {this or block size isn't a power of 2!!!}
1239:S
1240:C         4         blocksize := shifted_left(1, blockpower);
1241:C         4         if blocksize>tapebuf_maxsize then
1242:C         5           ioresc(zuninitialized);  {our buffer is too small to handle}
1243:S
1244:C         4         block := shifted_right(abs_position, blockpower);
1245:C         4         intra_block_offset := mod_power_of_2(abs_position, blockpower);
1246:S
1247:C         4         if blockpower<=8 then
1248:C         5           begin {handle a 256-byte or smaller block media}
1249:C         5             if intra_block_offset<>0 then ioresc(zbadmode);
1250:C         5             xfr(uep, request, bufptr, block, length);
1251:C         5           end {handle a 256-byte or smaller block device}
```



```
1252:C         5 $page$
1253:S
1254:C         5         else
1255:C         5           begin {handle buffering for up to tapebuf_maxsize block media}
1256:S
1257:C         5             partial_length := blocksize-intra_block_offset;
1258:C         5             if partial_length>length then partial_length := length;
1259:S
1260:C         5             case request of
1261:S
1262:C         6               readbytes, startread:
1263:C         6                 begin {read operations}
1264:S
1265:C         6                   if intra_block_offset>0 then  {partial block at front}
1266:C         7                     begin
1267:C         7                       load_tapebuf(uep, readbytes, block);
1268:C         7                       moveleft(tapebuf_ptr^[intra_block_offset], bufptr^, partial_length);
1269:C         7                       bufptr := addr(bufptr^, partial_length);
1270:C         7                       abs_position := abs_position+partial_length;
1271:C         7                       block := shifted_right(abs_position, blockpower);
1272:C         7                       length := length-partial_length;
1273:C         7                     end; {if}
1274:S
1275:C         6                   if length>=blocksize then  {one or more blocks remain}
1276:C         7                     begin
1277:C         7                       flush_tapebuf;  {because we may travel far, far away}
1278:C         7                       xfr(uep, readbytes, bufptr, block, length);
1279:C         7                     end
1280:C         7                   else if length>0 then  {partial block at back}
1281:C         8                     begin
1282:C         8                       load_tapebuf(uep, readbytes, block);
1283:C         8                       moveleft(tapebuf_ptr^, bufptr^, length);
1284:C         8                     end;
1285:S
1286:C         6                 end; {read operations}
1287:S
1288:C         6               writebytes, startwrite:
1289:C         6                 begin {write operations}
1290:S
1291:C         6                   if intra_block_offset>0 then  {partial block at front}
1292:C         7                     begin
1293:C         7                       load_tapebuf(uep, writebytes, block);
1294:C         7                       tapebuf_state := modified;
1295:C         7                       moveleft(bufptr^, tapebuf_ptr^[intra_block_offset], partial_length);
1296:C         7                       bufptr := addr(bufptr^, partial_length);
1297:C         7                       abs_position := abs_position+partial_length;
1298:C         7                       block := shifted_right(abs_position, blockpower);
1299:C         7                       length := length-partial_length;
1300:C         7                     end; {if}
1301:S
1302:C         6                   partial_length := length-mod_power_of_2(length, blockpower);
1303:C         6                   if partial_length>0 then  {one or more whole blocks remain}
1304:C         7                     begin
1305:C         7                       flush_tapebuf;  {because we may travel far, far away}
1306:C         7                       tapebuf_state := undefined;  {in case this overwrites!}
1307:C         7                       xfr(uep, writebytes, bufptr, block, partial_length);
1308:C         7                       bufptr := addr(bufptr^, partial_length);
1309:C         7                       abs_position := abs_position+partial_length;
1310:C         7                       block := shifted_right(abs_position, blockpower);
1311:C         7                       length := length-partial_length;
```

```
1312:C        7                    end; {if}
1313:S
1314:C        6                  if length>0 then  {a partial block remains}
1315:C        7                    begin
1316:C        7                      load_tapebuf(uep, writebytes, block);
1317:C        7                      tapebuf_state := modified;
1318:C        7                      moveleft(bufptr^, tapebuf_ptr^, length);
1319:C        7                    end; {if}
1320:S
1321:C        6                  flush_tapebuf;  {so errors get reported in the right place!}
1322:S
1323:C        6                end; {write operations}
1324:S
1325:C        6            end; {case}
1326:S
1327:C        5          end; {handle buffering for up to tapebuf_maxsize block media}
1328:S
1329:C        4       recover
1330:C        4         begin
1331:C        4           if escapecode<>-10 then
1332:C        5             escape(escapecode);
1333:C        4           if ioresult=ord(inoerror) then  {media changed; restart}
1334:C        5             retry_required := true;
1335:C        4         end; {recover}
1336:S
1337:C        3     until not retry_required;
1338:S
1339:C        2     if (request=startread) or (request=startwrite) then
1340:C        3       call(fp^.feot, fp)  {call the end of transfer procedure}
1341:C        3     else
1342:C        3       uep^.dvrtemp := ord(inoerror);  {report synchronous errors only once}
1343:S
1344:C        2   end; {transfer}
```



```
1345:D   -16  1 $page$
1346:S
1347:S          {
1348:S            CS80 transfer method request handler
1349:D   -16  1 }
1350:S
1351:D        1 procedure CS80io;
1352:D        2   var
1353:D     -4 2     uep: uep_type;
1354:C        2   begin {CS80io}
1355:S
1356:C        2     ioresult := ord(inoerror);
1357:C        2     uep := addr(unitable^[fp^.funit]);
1358:S
1359:C        2     if uep^.offline then
1360:C        3       ioresult := ord(znodevice)
1361:C        3     else
1362:C        3       case request of
1363:S
1364:C        4         clearunit:
1365:C        4           begin
1366:C        4             clear_unit(uep);
1367:C        4             ioresult := uep^.dvrtemp;
1368:C        4             uep^.dvrtemp := ord(inoerror);  {report synchronous errors only once}
1369:C        4             if uep=tapebuf_uep then tapebuf_state := undefined;
1370:C        4           end;
1371:S
1372:C        4         unitstatus:
1373:C        4           fp^.fbusy := unit_busy(uep);
1374:S
1375:C        4         flush:
1376:C        4           begin
1377:C        4             if uep=tapebuf_uep then
1378:C        5               try
1379:C        6                 flush_tapebuf;
1380:C        6               recover
1381:C        6                 if escapecode<>-10 then escape(escapecode);
1382:C        4             uep^.dvrtemp := ord(inoerror);  {report synchronous errors only once}
1383:C        4           end;
1384:S
1385:C        4         readbytes, writebytes, startread, startwrite:
1386:C        4           begin
1387:C        4             uep^.dvrtemp := ord(inoerror);  {report synchronous errors only once}
1388:C        4             if Simon_no_DMA(uep) then
1389:C        5               ioresult := ord(zbaddma)
1390:C        5             else if uep^.ureportchange and not uep^.umediavalid then
1391:C        6               ioresult := ord(zmediumchanged)
1392:C        6             else if (position<0) or (length<0) or (position+length>fp^.fpeof) then
1393:C        7               ioresult := ord(ieof)
1394:C        7             else
1395:C        7               transfer(uep, fp, request, addr(buffer), position+fp^.fileid+uep^.byteoffset, length);
1396:C        4           end;
1397:S
1398:C        4         otherwise  {unrecognized request}
1399:C        4           ioresult := ord(ibadrequest);
1400:S
1401:C        4       end; {cases}
1402:C        2   end; {CS80io}
1403:S
1404:C        1 end; {CS80dvr}
```

```
1405:D        1 $page$
1406:S
1407:S
1408:D        1 { program CS80init }
1409:S
1410:D        1 import
1411:D        1   tapebuf, loader;
1412:S
1413:C        1 begin {CS80init}
1414:C        1   init_tapebuf;
1415:C        1   markuser;
1416:C        1 end. {CS80init}
1417:S

No errors. No warnings.
             ***** Nonstandard language features enabled *****
```

CTABLE

Description

CTABLE is the automatic-configuration program, whose code file (named TABLE) is executed at boot-up.

Notes

Refer to the Pascal 3.0 Workstation System manual for details on how to modify this program. TABLE can be re-executed at any time to re-initialize the device table.

```
                                                       (*

     (c) Copyright Hewlett-Packard Company, 1983.
    All rights are reserved.  Copying or other
    reproduction of this program except for archival
    purposes is prohibited without the prior
    written consent of Hewlett-Packard Company.


                RESTRICTED RIGHTS LEGEND

    Use, duplication, or disclosure by the Government
    is subject to restrictions as set forth in
    paragraph (b) (3) (B) of the Rights in Technical
    Data and Computer Software clause in
    DAR 7-104.9(a).

    HEWLETT-PACKARD COMPANY
  0 Fort Collins, Colorado                             *)
```



```
0 $page, sysprog$

0 (*******************************************)
0 (*                                         *)
0 (*  Note: You will need to use the following  *)
0 (*  compiler directive if the 'INTERFACE'   *)
0 (*  file is not in your current LIBRARY.    *)
0 (*  Modify the volume name as necessary for  *)
0 (*  your configuration.                      *)
0 (*                                         *)
0 (*  $search  'CONFIG:INTERFACE.'$            *)
0 (*                                         *)
0 (*******************************************)

0 program (self-configuring) ctable;

1 module options;

1    (*******************************************)
1    (* Choose the desired configuration options *)
1    (* by editing the CONSTant declarations in  *)
1    (* this module.                             *)
1    (*******************************************)

1 import
1    sysglobals;

1 export

1 {power-up system unit}
1    const
1       specified_system_unit =
1         0;  {<>0 overrides auto-assignment}


1 {local mass storage directory access method}
1    type
1       lms_dam_type = {local mass storage dam}
1         ( LIF, UCSD );
1    const
1       primary_lms_dam =
1         LIF;


1 {floppy/harddisc unit number slot tradeoff's}
1    const
1       floppy_unit_pairs =  {[1..10]}
1         3;
1       first_harddisc_lun = {do not edit!}
1         7+(floppy_unit_pairs-1)*2;
1       last_harddisc_lun =
1         40;
```

```
    75:D        1 $page$
    76:S
    77:D        1 (local printer type option)
    78:D        1   type
    79:D        1     local_printer_type = (HPIB, RS232);
    80:D        1   const
    81:D        1     local_printer_option = HPIB;
    82:S
    83:S
    84:D        1 (local printer timeout)
    85:S            {
    86:S              maximum allowed delay between any two bytes:
    87:S                >0  specifies milliseconds (up to one hour)
    88:S                =0  specifies infinite timeout
    89:S
    90:S              recommended values:
    91:S                -  HP2630 series  (HP-IB)       3000
    92:S                -  HP2670 series  (HP-IB)       3000
    93:S                -  HP9876         (HP-IB)       7000
    94:S                -  HP82905        (HP-IB)      12000
    95:S              Note: the HP82905 is currently NOT supported
    96:S                due to its improper response to interface
    97:S                clear (IFC), and its incompatible graphics
    98:S                dump sequence.
    99:D        1   }
   100:D        1   const
   101:D        1     local_printer_timeout =
   102:D        1       $IF local_printer_option=HPIB$
   103:D        1         12000;  {milliseconds}
   104:D        1       $END$
   105:S                $IF local_printer_option=RS232$
   106:S                  0;      {infinite}
   107:D        1       $END$
   108:S
   109:S
   110:D        1 {default dav's for devices not found by scanning}
   111:D        1   type
   112:D        1     dav_type = {device address vector}
   113:D        1       packed record
   114:D        1         sc, ba, du, dv: -128..127;
   115:D        1       end;
   116:D        1   const
   117:D        1     HP9885_default_dav =
   118:D        1       dav_type[sc: 12, ba: -1, du:  0, dv: -1];
   119:D        1     SRM_default_dav =
   120:D        1       dav_type[sc: 21, ba: {node} 0,
   121:D        1                du: {unit} 8, dv: -1];
   122:D        1     BUBBLE_default_dav =
   123:D        1       dav_type[sc: 30, ba:  0, du:  0, dv:  0];
   124:D        1     local_HPIB_printer_default_dav =
   125:D        1       dav_type[sc:  7, ba:  1, du: -1, dv: -1];
   126:D        1     local_RS232_printer_default_dav =
   127:D        1       dav_type[sc:  9, ba:  0, du: -1, dv: -1];
   128:S
```



```
   129:D        1 $page$
   130:S
   131:D        1 (local hard disc partitioning parameters)
   132:D        1   type
   133:D        1     pp_type =  {partitioning parameters}
   134:D        1       record
   135:D        1         mvs: integer;   {min vol size in bytes}
   136:D        1         mnv: shortint;  {max number of volumes}
   137:D        1       end;
   138:S            {   In general, MNV puts an upper bound on the
   139:S              number of logical volumes that a physical
   140:S              volume can be partitioned into.  Depending
   141:S              upon MNV's range, however, several types of
   142:S              behavior can occur.
   143:S                If MNV=0, then no logical volumes will ever
   144:S              be assigned for the device.
   145:S                If abs(MNV)=1, then exactly one logical
   146:S              volume will be assigned per physical volume
   147:S              of the device.  This corresponds to the
   148:S              2.X CTABLE's "single_volume" mode.
   149:S                If MNV>1, then partitioning will always be
   150:S              performed, subject to meeting the minimum
   151:S              volume size restrictions.  This corresponds
   152:S              to the 2.X CTABLE's "multi_volume" mode.
   153:S                If MNV<-1, then partitioning will be
   154:S              performed, but afterwards any logical volume
   155:S              that does not contain a valid directory will
   156:S              be coalesced with a previous adjacent logical
   157:S              volume if that one DOES contain a valid
   158:S              directory. As an extreme case, if only a
   159:S              single directory exists, and it is at the
   160:S              beginning of the physical volume, then all
   161:S              following logical volumes will be coalesced
   162:S              with the first, providing the same behavior
   163:S              as the 2.X CTABLE's "auto_volume" mode. With
   164:S              the less extreme cases, a wide variety of
   165:S              partitioning options are now possible without
   166:D        1     modification to CTABLE.                    }
   167:D        1   const
   168:D        1     min_size = {in bytes [1..maxint]}
   169:D        1       1000000;
   170:D        1     max_vols = {[-30..30]; <0 means autocoalesce}
   171:D        1       -30;
   172:D        1     HP913X_A_pp =
   173:D        1       pp_type[mvs: min_size, mnv: max_vols];
   174:D        1     HP913X_B_pp =
   175:D        1       pp_type[mvs: min_size, mnv: max_vols];
   176:D        1     HP913X_C_pp =
   177:D        1       pp_type[mvs: min_size, mnv: max_vols];
   178:D        1     CS80disc_pp =
   179:D        1       pp_type[mvs: min_size, mnv: max_vols];
```

```
180:D        1 $page$
181:S
182:D        1 {system unit auto-search declarations}
183:D        1   const
184:D        1     sysunit_list_length =
185:D        1       7;
186:D        1   type
187:D        1     sysunit_list_type =
188:D        1       array[1..sysunit_list_length] of unitnum;
189:D        1   const
190:D        1     sysunit_list =
191:D        1       sysunit_list_type[
192:D        1           first_harddisc_lun,
193:D        1           45,   {srm, prefixed to user's sysvol}
194:D        1           4,    {floppy unit 1, primary DAM}
195:D        1           44,   {floppy unit 1, secondary DAM}
196:D        1           3,    {floppy unit 0, primary DAM}
197:D        1           43,   {floppy unit 0, secondary DAM}
198:D        1           42];  {bubble}
199:S
200:S
201:D        1 {HP-IB select code scanning declarations}
202:D        1   const
203:D        1     sc_list_length =
204:D        1       3;
205:D        1   type
206:D        1     sc_list_type =
207:D        1       array[1..sc_list_length] of shortint;
208:D        1   const
209:D        1     sc_list =
210:D        1       sc_list_type[
211:D        1           7,    {internal HP-IB}
212:D        1           8,    {default sc for HP98624 HP-IB}
213:D        1           14];  {default sc for HP98625 HP-IB}
214:S
215:S
216:D        1 implement {options}
217:S
218:C        1 end; {options}
```



```
219:D        1 $page, range off, ovflcheck off, partial_eval on$
220:S
221:D        1 module ctr; {ctable routines}
222:S
223:D        1   (*********************************************)
224:D        1   (*                                           *)
225:D        1   (*                 Warning:                  *)
226:D        1   (*   This module should not be modified!     *)
227:D        1   (*                                           *)
228:D        1   (*********************************************)
229:S
230:D        1 import
231:D        1   sysglobals, loader, options, ldr;
232:S
233:D        1 export
234:S
235:D        1   const {mass storage letter specifiers}
236:D        1     INTERNAL  = 'M';
237:D        1     HP8290X   = 'N';
238:D        1     HP9885    = 'F';
239:D        1     HP9895    = 'H';
240:D        1     HP913X_A  = 'U';
241:D        1     HP913X_B  = 'V';
242:D        1     HP913X_C  = 'W';
243:D        1     CS80      = 'Q';
244:D        1     SRM       = 'G';
245:D        1     PRINTER   = 'J';
246:D        1     RAM       = 'R';
247:D        1     BUBBLE    = 'B';
248:D        1     EPROM     = 'E';
249:D        1     NODEVICE  = #255;
250:S
251:S
252:D        1   type
253:D        1     flpy_flags_type = {flags governing floppy unit pair assignments}
254:D        1       packed record
255:D        1         assign_even_unit, assign_odd_unit: boolean;
256:D        1       end;
257:S
258:D        1   const
259:D        1     assign_neither_flpy_unit =
260:D        1       flpy_flags_type[assign_even_unit: false, assign_odd_unit: false];
261:D        1     assign_both_flpy_units =
262:D        1       flpy_flags_type[assign_even_unit: true, assign_odd_unit: true];
263:S
264:D        1   type
265:D        1     MSUS_type = {Mass Storage Unit Specifier}
266:D        1       record
267:D        1         flpy_flags: flpy_flags_type;
268:D        1         letter: char;  {from the above mass storage letter specifiers}
269:D        1         dav: dav_type;
270:D        1       end;
```

```
   271:D        1 $page$
   272:S
   273:D        1     mp_type =  (medium parameters)
   274:D        1       record
   275:D        1         tpm: integer;  {tracks per medium}
   276:D        1         bpt: integer;  {bytes per track}
   277:D        1       end;
   278:S
   279:D        1     ds_type =  {Directory access method Specifier for local mass storage}
   280:D        1       ( primary_dam,     {either LIF or UCSD, as specified in options}
   281:D        1         secondary_dam,   {the one not selected as primary}
   282:D        1         LIF_dam,         {LIF, regardless of primary/secondary choice}
   283:D        1         UCSD_dam  );     {UCSD, regardless of primary/secondary choice}
   284:S
   285:D        1   var
   286:D    -6  1     bootdev_MSUS: MSUS_type;
   287:D    -8  1     bootdev_lun: unitnum;
   288:S
   289:S
   290:D    -8  1   procedure create_temp_unitable;
   291:D    -8  1   procedure assign_and_clear_unit(lunit: unitnum);
   292:D    -8  1   procedure assign_temp_unitable;
   293:D    -8  1   function sysunit_ok(system_unit: unitnum): boolean;
   294:D    -8  1   procedure zap_assigned_unit(lunit: unitnum);
   295:D    -8  1   function on_same_medium(lun1, lun2: unitnum): boolean;
   296:D    -8  1   procedure remove_extraneous_volumes(first_lun, last_lun: unitnum);
   297:D    -8  1   function medium_parameters(letter: char): mp_type;
   298:D    -8  1   function partitioning_parameters(letter: char): pp_type;
   299:D    -8  1   function number_vols(mp: mp_type; pp: pp_type): shortint;
   300:D    -8  1   function svol_bytes(letter: char): integer;
   301:D    -8  1   function vol_bytes(current_vol, number_vols: shortint; mp: mp_type): integer;
   302:D    -8  1   function vol_offset(current_vol, number_vols: shortint; mp: mp_type): integer;
   303:D    -8  1   function block_boundaries(mp: mp_type): mp_type;
   304:D    -8  1   function value(symbol: string255): integer;
   305:D    -8  1   function MSUSs_match(MSUS1, MSUS2: MSUS_type): boolean;
   306:S
   307:S
   308:S
   309:D    -8  1   { table entry assignment procedures }
   310:S
   311:D    -8  1   procedure tea_memory_volume_dam(ds:ds_type);
   312:D    -8  1   procedure tea_boot(un:unitnum);
   313:D    -8  1   procedure tea_srm(un:unitnum;sc,ba,du:shortint);
   314:D    -8  1   procedure tea_crt(un:unitnum);
   315:D    -8  1   procedure tea_kbd(un:unitnum);
   316:D    -8  1   procedure tea_local_printer(un:unitnum;sc,ba:shortint;uvid:vid;bto:integer);
   317:D    -8  1   procedure tea_mini(un:unitnum;ds:ds_type;du:shortint);
   318:D    -8  1   procedure tea_HP9885(un:unitnum;ds:ds_type;sc,du,block_os:shortint);
   319:D    -8  1   procedure tea_HP9895(un:unitnum;ds:ds_type;sc,ba,du,block_os:shortint);
   320:D    -8  1   procedure tea_HP8290X(un:unitnum;ds:ds_type;sc,ba,du:shortint);
   321:D    -8  1   procedure tea_flpy(un:unitnum;lr:char;ds:ds_type;sc,ba,du:shortint);
   322:D    -8  1   procedure tea_amigo_sv(un:unitnum;ds:ds_type;sc,ba,du:shortint;os:integer;lr:char;mb:integer);
   323:D    -8  1   procedure tea_CS80_mv(un:unitnum;ds:ds_type;sc,ba,du,dv:shortint;os,id,mb:integer);
   324:D    -8  1   procedure tea_CS80_sv(un:unitnum;ds:ds_type;sc,ba,du,dv:shortint);
   325:D    -8  1   procedure tea_BUBBLE(un:unitnum;ds:ds_type;sc:shortint);
   326:D    -8  1   procedure tea_EPROM(un:unitnum;ds:ds_type;sn:shortint);
```



```
   327:D    -8  1 $page$
   328:S
   329:D    -8  1 implement (ctr)
   330:S
   331:D    -8  1 const  (abbreviation for tea procedure calls)
   332:D    -8  1   T = true;
   333:D    -8  1   F = false;
   334:S
   335:D    -8  1 const  (actual driver entry point names)
   336:D    -8  1   NO_DAM_name          = 'INITUNITS_NODAM';
   337:D    -8  1   BOOT_DAM_name        = 'BOOTDAMMODULE_BOOTDAM';
   338:D    -8  1   LIF_DAM_name         = 'LIFMODULE_LIFDAM';
   339:D    -8  1   UCSD_DAM_name        = 'UCSDMODULE_UCSD_DAM';
   340:D    -8  1   UNBLOCKED_DAM_name   = 'MISC_UNBLOCKEDDAM';
   341:D    -8  1   SRM_DAM_name         = 'SRMDAMMODULE_SRMDAM';
   342:S
   343:D    -8  1   NULL_TM_name         = 'INITUNITS_NOUNIT';
   344:D    -8  1   BOOT_TM_name         = 'BOOTDAMMODULE_BOOTTM';
   345:D    -8  1   CRT_TM_name          = 'SYSDEVS_CRTIO';
   346:D    -8  1   KBD_TM_name          = 'SYSDEVS_KBDIO';
   347:D    -8  1   MINI_TM_name         = 'MINI_MINIIO';
   348:D    -8  1   SRM_TM_name          = 'SRMAMMODULE_SRMAM';
   349:D    -8  1   PRINTER_TM_name      = 'PRTDVR_PRTIO';
   350:D    -8  1   F9885_TM_name        = 'F9885DVR_F9885IO';
   351:D    -8  1   AMIGO_TM_name        = 'AMIGODVR_AMIGOIO';
   352:D    -8  1   CS80_TM_name         = 'CS80DVR_CS80IO';
   353:D    -8  1   BUBBLE_TM_name       = 'BUBBLE_BUB_TM';
   354:D    -8  1   EPROM_TM_name        = 'EPROMS_EPROM_TM';
   355:S
   356:S
   357:D    -8  1 var
   358:D   -12  1   temp_unitable: unitableptr;
   359:S
   360:S
   361:D        1 procedure check(parameter, lower_bound, upper_bound: integer);
   362:C        2   begin
   363:C        2     if (parameter<lower_bound) or (parameter>upper_bound) then
   364:C        3       halt(-8) {value range error}
   365:C        3   end;
   366:S
```

```
367:D   -12  1 $page$
368:S
369:D  -256  1 function value(symbol: string255): integer;
370:D  -256  2   var
371:D  -260  2     modp: moddescptr;
372:D  -268  2     ptr, valueptr: addrec;
373:D  -269  2     found: boolean;
374:C        2   begin {value}
375:C        2     value := 0;
376:C        2     found := false;
377:C        2     modp := sysdefs;
378:C        2     while (modp<>nil) and not found do
379:C        3       with modp^ do
380:C        4         begin
381:C        4           ptr := defaddr;
382:C        4           while (ptr.a<defaddr.a+defsize) and not found do
383:C        5             begin
384:C        5               found := ptr.syp^=symbol;
385:C        5               ptr.a := ptr.a+strlen(ptr.syp^)+1;
386:C        5               ptr.a := ptr.a+ord(odd(ptr.a));
387:C        5               valueptr.a := ptr.a+2;
388:C        5               if found then
389:C        6                 value := valueptr.vep^.value;
390:C        5               ptr.a := ptr.a+ptr.gvp^.short;
391:C        5             end; {while}
392:C        4           modp := link;
393:C        4         end; {with modp^}
394:C        2   end; {value}
395:S
396:S
397:D   -12  1 function MSUSs_match(MSUS1, MSUS2: MSUS_type): boolean;
398:C        2   begin {MSUSs_match}
399:C        2     MSUSs_match := (MSUS1.letter = MSUS2.letter) and
400:C        2                    (MSUS1.dav.sc = MSUS2.dav.sc) and
401:C        2                    (MSUS1.dav.ba = MSUS2.dav.ba) and
402:C        2                    (MSUS1.dav.du = MSUS2.dav.du) and
403:C        2                    (MSUS1.dav.dv = MSUS2.dav.dv);
404:C        2   end; {MSUSs_match}
```



```
405:D   -12  1 $page$
406:S
407:D        1 procedure tea {lowest-level Table Entry Assignment procedure}
408:D        2     ( un:unitnum;                    {unit number}
409:D        2       dam_name: string255;           {directory access method}
410:D        2       tm_name: string255;            {transfer method (driver)}
411:D        2       p_sc: shortint;                {select code}
412:D        2       p_ba: shortint;                {bus address}
413:D        2       p_du: shortint;                {disc unit}
414:D        2       p_dv: shortint;                {disc volume}
415:D        2       p_byteoffset: integer;         {physical starting byte of volume}
416:D        2       p_devid: integer;              {device identifier (driver dependent)}
417:D        2       p_uvid: vid;                   {volume id}
418:D        2     { p_drvtemp: integer             {driver temp}
419:D        2     { p_drvtemp2: shortint;          {second driver temp}
420:D        2       p_letter: char;                {device specifier letter}
421:D        2     { p_offline: boolean             {unit offline flag}
422:D        2       p_uisinteractive: boolean;     {device echos input}
423:D        2     { p_umediavalid: boolean;        {open files are valid}
424:D        2     { p_uuppercase: boolean;         {volume name should be uppercased}
425:D        2       p_uisfixed: boolean;           {medium not removable flag}
426:D        2     { p_..eportchange: boolean;      {driver directive to report/ignore medium changes}
427:D        2     { p_pad: 0..1                    {(not used)}
428:D        2       p_uisblkd: boolean;            {blocked volume flag}
429:D  -530  2       p_umaxbytes: integer  );       {volume size in bytes}
430:S
431:D  -530  2   var
432:D  -530  2     dam_proc:
433:D  -530  2       packed record case integer of
434:D  -530  2         0: (dam: damtype);
435:D  -530  2         1: (value, slink: integer);
436:D  -538  2       end;
437:S
438:D  -538  2     tm_proc:
439:D  -538  2       packed record case integer of
440:D  -538  2         0: (tm: amtype);
441:D  -538  2         1: (value, slink: integer);
442:D  -546  2       end;
443:S
444:C        2   begin {tea}
445:C        2     if temp_unitable=nil then halt(-3); {unassigned pointer}
446:S
447:C        2     dam_proc.value := value(dam_name);
448:C        2     dam_proc.slink := 0;
449:S
450:C        2     tm_proc.value := value(tm_name);
451:C        2     tm_proc.slink := 0;
452:S
453:C        2     if (dam_proc.value<>0) and (tm_proc.value<>0) then  {assign the entry}
454:C        3       begin
455:S
456:C        3         with temp_unitable^[un] do
457:C        4           begin
458:C        4             dam             := dam_proc.dam;
459:C        4             tm              := tm_proc.tm;
460:C        4             sc              := p_sc;
461:C        4             ba              := p_ba;
462:C        4             du              := p_du;
463:C        4             dv              := p_dv;
464:C        4             byteoffset      := p_byteoffset;
```

```
465:C        4             devid           := p_devid;
466:C        4             uvid            := p_uvid;
467:C        4             dvrtemp         := 0;                 {always initially zero!}
468:C        4             dvrtemp2        := -1;                {always initially -1!}
469:C        4             letter          := p_letter;
470:C        4             offline         := false;             {always initially online!}
471:C        4             uisinteractive  := p_uisinteractive;
472:C        4             umediavalid     := false;             {never valid to start with}
473:C        4             uuppercase      := not p_uisblkd;     {assume case is significant}
474:C        4             uisfixed        := p_uisfixed;
475:C        4             ureportchange   := true;              {do report media changes}
476:C        4             pad             := 0;                 {not used}
477:C        4             uisblkd         := p_uisblkd;
478:C        4             if uisblkd then
479:C        5               umaxbytes     := p_umaxbytes;
480:C        4           end; {with}
481:S
482:C        3         with bootdev_MSUS, dav do {see if this entry points to it}
483:C        4           if (p_letter=letter) and  {wish we could use MSUSs_match function}
484:C        5              (p_sc=sc) and (p_ba=ba) and (p_du=du) and (p_dv=dv) and
485:C        5              ((dam_name=LIF_DAM_name) or (p_letter=SRM)) and
486:C        5              (p_byteoffset=0) then  {remember this unit number!}
487:C        5             bootdev_lun := un;
488:S
489:C        3       end; {if}
490:C        2   end; {tea}
491:S
492:S
493:D        1 function dam(ds: ds_type): string255;
494:C        2   begin
495:C        2     case ds of
496:C        3       primary_dam:
497:C        3         if primary_lms_dam=LIF
498:C        4           then dam := LIF_DAM_name
499:C        4           else dam := UCSD_DAM_name;
500:C        3       secondary_dam:
501:C        3         if primary_lms_dam=LIF
502:C        4           then dam := UCSD_DAM_name
503:C        4           else dam := LIF_DAM_name;
504:C        3       LIF_dam:
505:C        3         dam := LIF_DAM_name;
506:C        3       UCSD_dam:
507:C        3         dam := UCSD_DAM_name;
508:C        3     end; {case}
509:C        2   end;
```



```
510:D   -12  1 $page$
511:S
512:D        1 function medium_parameters(letter: char): mp_type;
513:D        2   const {LOGICAL sizes unless otherwise noted}
514:D        2     INTERNAL_mp = mp_type[tpm: 2* 33, bpt: 16*256];
515:D        2     HP8290X_mp  = mp_type[tpm: 2* 33, bpt: 16*256];
516:D        2     HP9885_mp   = mp_type[tpm: 1* 77, bpt: 30*256];  {physical size}
517:D        2     HP9895_mp   = mp_type[tpm: 2* 77, bpt: 30*256];  {physical size}
518:D        2     HP913X_A_mp = mp_type[tpm: 4*152, bpt: 31*256];
519:D        2     HP913X_B_mp = mp_type[tpm: 4*305, bpt: 31*256];
520:D        2     HP913X_C_mp = mp_type[tpm: 6*305, bpt: 31*256];
521:D        2     BUBBLE_mp   = mp_type[tpm: 1*512, bpt:  1*256];  {1 megabit unit}
522:D        2    {BUBBLE_mp   = mp_type[tpm: 4*512, bpt:  1*256];} {4 megabit unit}
523:D        2     null_mp     = mp_type[tpm:     0, bpt:      0];
524:C        2   begin
525:C        2     case letter of
526:C        3       INTERNAL:  medium_parameters := INTERNAL_mp;
527:C        3       HP8290X:   medium_parameters := HP8290X_mp;
528:C        3       HP9885:    medium_parameters := HP9885_mp;
529:C        3       HP9895:    medium_parameters := HP9895_mp;
530:C        3       HP913X_A:  medium_parameters := HP913X_A_mp;
531:C        3       HP913X_B:  medium_parameters := HP913X_B_mp;
532:C        3       HP913X_C:  medium_parameters := HP913X_C_mp;
533:C        3       BUBBLE:    medium_parameters := BUBBLE_mp;
534:C        3       otherwise  medium_parameters := null_mp;
535:C        3     end; {case}
536:C        2   end;
537:S
538:S
539:D        1 function partitioning_parameters(letter: char): pp_type;
540:D        2   const
541:D        2     null_pp = pp_type[mvs: 0, mnv: 0];
542:C        2   begin
543:C        2     case letter of
544:C        3       HP913X_A:  partitioning_parameters := HP913X_A_pp;
545:C        3       HP913X_B:  partitioning_parameters := HP913X_B_pp;
546:C        3       HP913X_C:  partitioning_parameters := HP913X_C_pp;
547:C        3       CS80:      partitioning_parameters := CS80disc_pp;
548:C        3       otherwise  partitioning_parameters := null_pp;
549:C        3     end; {case}
550:C        2   end;
```

```
551:D   -12  1 $page$
552:S
553:D   -14  1 function number_vols(mp: mp_type; pp: pp_type): shortint;
554:D   -14  2   var
555:D   -16  2     nvols: shortint;
556:C        2   begin
557:C        2     if pp.mnv<0 then  {negative implies autovolume feature; use absolute value}
558:C        3       pp.mnv := -pp.mnv;
559:C        2     if pp.mvs<=0 then pp.mvs := 1;  {guard against div's by 0}
560:C        2     if mp.bpt<=0 then mp.bpt := 1;  {guard against div's by 0}
561:C        2     nvols := mp.tpm div ((pp.mvs*mp.bpt-1) div mp.bpt);
562:C        2     if (nvols=0) and (mp.tpm>0) then
563:C        3       nvols := 1;  {physical vol smaller than the specified minimum vol size}
564:C        2     if nvols>pp.mnv then  {cut back, even to zero if specified}
565:C        3       nvols := pp.mnv;
566:C        2     number_vols := nvols;
567:C        2   end;
568:S
569:S
570:D        1 function svol_bytes(letter: char): integer;
571:D        2   var
572:D    -8  2     mp: mp_type;
573:C        2   begin
574:C        2     mp := medium_parameters(letter);
575:C        2     svol_bytes := mp.bpt*mp.tpm;  {single volume bytes}
576:C        2   end;
577:S
578:S
579:D    -8  1 function vol_bytes(current_vol, number_vols: shortint; mp: mp_type): integer;
580:D    -8  2   var
581:D   -12  2     tracks: integer;
582:C        2   begin
583:C        2     tracks := mp.tpm div number_vols;           {each vol gets this much}
584:C        2     if current_vol=number_vols-1 then
585:C        3       tracks := tracks+mp.tpm mod number_vols;  {last vol gets any extra}
586:C        2     vol_bytes := tracks*mp.bpt;
587:C        2   end;
588:S
589:S
590:D    -8  1 function vol_offset(current_vol, number_vols: shortint; mp: mp_type): integer;
591:C        2   begin
592:C        2     vol_offset := (mp.tpm div number_vols)*current_vol*mp.bpt;
593:C        2   end;
594:S
595:S
596:D    -8  1 function block_boundaries(mp: mp_type): mp_type;
597:C        2   begin
598:C        2     block_boundaries.tpm := mp.tpm*mp.bpt div 512;
599:C        2     block_boundaries.bpt := 512;
600:C        2   end;
```



```
601:D   -12  1 $page$
602:S
603:D   -12  1 { standard driver-oriented table entry assignment procedures }
604:S
605:S
606:D        1 procedure tea_nounit(un:unitnum);
607:C        2   begin
608:C        2     tea(un,NO_DAM_name,NULL_TM_name,0,0,0,0,0,0,'',#0,F,F,F,0);
609:C        2   end;
610:S
611:S
612:D        1 procedure tea_memory_volume_dam(ds:ds_type);
613:C        2   begin
614:C        2     tea(0,dam(ds),NULL_TM_name,0,0,0,0,0,0,'',RAM,F,T,T,0);
615:C        2   end;
616:S
617:S
618:D        1 procedure tea_crt(un:unitnum);
619:C        2   begin
620:C        2     tea(un,UNBLOCKED_DAM_name,CRT_TM_name,0,0,0,0,0,0,'CONSOLE',#0,T,T,F,0);
621:C        2   end;
622:S
623:S
624:D        1 procedure tea_kbd(un:unitnum);
625:C        2   begin
626:C        2     tea(un,UNBLOCKED_DAM_name,KBD_TM_name,0,0,0,0,0,0,'SYSTERM',#0,F,T,F,0);
627:C        2   end;
628:S
629:S
630:D        1 procedure tea_mini(un:unitnum;ds:ds_type;du:shortint);
631:C        2   begin
632:C        2     check(du, 0, 1);
633:C        2     tea(un,dam(ds),MINI_TM_name,0,0,du,0,0,0,'',INTERNAL,F,F,T,svol_bytes(INTERNAL));
634:C        2   end;
635:S
636:S
637:D        1 procedure tea_boot(un: unitnum);
638:C        2   begin
639:C        2     tea(un,BOOT_DAM_name,BOOT_TM_name,0,0,0,0,0,0,'',#0,F,F,T,maxint);
640:C        2   end;
641:S
642:S
643:D        1 procedure tea_srm(un:unitnum;sc,{node}ba,{unit}du:shortint);
644:C        2   begin
645:C        2     check(sc, 7, 31);
646:C        2     check(ba, 0, 63);
647:C        2     if du<>0 then check(du, 7, 26);
648:C        2     tea(un,SRM_DAM_name,SRM_TM_name,sc,ba,du,0,0,0,'',SRM,F,T,T,maxint);
649:C        2   end;
650:S
651:S
652:D   -18  1 procedure tea_local_printer(un:unitnum;sc,ba:shortint;uvid:vid;bto:integer);
653:C        2   begin
654:C        2     check(sc, 7, 31);
655:C        2     check(ba, 0, 30);
656:C        2     check(bto, 0, 60*60*1000);  {one hour should be enough!}
657:C        2     tea(un,UNBLOCKED_DAM_name,PRINTER_TM_name,sc,ba,0,0,0,bto,uvid,#0,F,T,F,0);
658:C        2   end;
```

```
   659:D   -12  1 $page$
   660:S
   661:D        1 procedure tea_HP9885(un:unitnum;ds:ds_type;sc,du,block_os:shortint);
   662:D        2   var
   663:D    -4  2     os: integer;
   664:C        2   begin
   665:C        2     check(sc, 8, 31);
   666:C        2     check(du, 0, 3);
   667:C        2     os := block_os*512;
   668:C        2     check(os, 0, svol_bytes(HP9885)-1);
   669:C        2     tea(un,dam(ds),F9885_TM_name,sc,0,du,0,os,0,'',HP9885,F,F,T,svol_bytes(HP9885)-os);
   670:C        2   end;
   671:S
   672:S
   673:D        1 procedure tea_HP9895(un:unitnum;ds:ds_type;sc,ba,du,block_os:shortint);
   674:D        2   var
   675:D    -4  2     os: integer;
   676:C        2   begin
   677:C        2     check(sc, 7, 31);
   678:C        2     check(ba, 0, 7);
   679:C        2     check(du, 0, 3);
   680:C        2     os := block_os*512;
   681:C        2     check(os, 0, svol_bytes(HP9895)-1);
   682:C        2     tea(un,dam(ds),AMIGO_TM_name,sc,ba,du,0,os,0,'',HP9895,F,F,T,svol_bytes(HP9895)-os);
   683:C        2   end;
   684:S
   685:S
   686:D        1 procedure tea_HP8290X(un:unitnum;ds:os_type;sc,ba,du:shortint);
   687:C        2   begin
   688:C        2     check(sc, 7, 31);
   689:C        2     check(ba, 0, 7);
   690:C        2     check(du, 0, 3);
   691:C        2     tea(un,dam(ds),AMIGO_TM_name,sc,ba,du,0,0,0,'',HP8290X,F,F,T,svol_bytes(HP8290X));
   692:C        2   end;
   693:S
   694:S
   695:D        1 procedure tea_amigo_sv(un:unitnum;ds:ds_type;sc,ba,du:shortint;os:integer;lr:char;mb:integer);
   696:D        2   var
   697:D    -4  2     medium_size: integer;
   698:C        2   begin
   699:C        2     check(sc, 7, 31);
   700:C        2     check(ba, 0, 7);
   701:C        2     check(du, 0, 7);
   702:C        2     if not (lr in [HP913X_A, HP913X_B, HP913X_C]) then
   703:C        3       halt(-8); {value range error}
   704:C        2     medium_size := svol_bytes(lr);
   705:C        2     check(os, 0, medium_size-1);
   706:C        2     if os mod 256<>0 then halt(-8) {value range error};
   707:C        2     check(mb, 1, medium_size-os);
   708:C        2     tea(un,dam(ds),AMIGO_TM_name,sc,ba,du,0,os,0,'',lr,F,T,T,mb);
   709:C        2   end;
```



```
   710:D   -12  1 $page$
   711:S
   712:D        1 procedure tea_CS80_mv(un:unitnum;ds:ds_type;sc,ba,du,dv:shortint;os,id,mb:integer);
   713:S          {
   714:S            CS80 multiple (logical) volume assignment procedure:
   715:S            1) devid must match the actual HP product number as found in describe
   716:S            2) offset and umaxbytes are fixed
   717:S            3) the uisfixed field is assigned by the driver in the clearunit procedure
   718:D        2   }
   719:C        2   begin
   720:C        2     check(sc, 7, 31);
   721:C        2     check(ba, 0, 7);
   722:C        2     check(du, 0, 14);
   723:C        2     check(dv, 0, 7);
   724:C        2     tea(un,dam(ds),CS80_TM_name,sc,ba,du,dv,os,id,'',CS80,F,F,T,mb);
   725:C        2   end;
   726:S
   727:S
   728:D        1 procedure tea_CS80_sv(un:unitnum;ds:ds_type;sc,ba,du,dv:shortint);
   729:S          {
   730:S            CS80 single (logical) volume assignment procedure:
   731:S            1) byteoffset always assumed to be zero
   732:S            2) umaxbytes is dependent upon the media loaded, thus it is set by the
   733:S               driver at clearunit time and whenever it detects a media change.
   734:S               BOTTOM LINE: the medium CANNOT be partitioned into multiple volumes!
   735:S            3) the uisfixed field is assigned by the driver in the clearunit procedure
   736:S            4) device can either be a disc (presumably a floppy) or a tape
   737:D        2   }
   738:C        2   begin
   739:C        2     check(sc, 7, 31);
   740:C        2     check(ba, 0, 7);
   741:C        2     check(du, 0, 14);
   742:C        2     check(dv, 0, 7);
   743:C        2     tea(un,dam(ds),CS80_TM_name,sc,ba,du,dv,0,-1,'',CS80,F,F,T,0);
   744:C        2   end;
   745:S
   746:S
   747:D        1 procedure tea_flpy(un:unitnum;lr:char;ds:ds_type;sc,ba,du:shortint);
   748:C        2   begin
   749:C        2     case lr of
   750:C        3       INTERNAL: tea_mini(un,ds,du);
   751:C        3       HP8290X:  tea_HP8290X(un,ds,sc,ba,du);
   752:C        3       CS80:     tea_CS80_sv(un,ds,sc,ba,du,0);
   753:C        3       HP9885:   tea_HP9885(un,ds,sc,du,0);
   754:C        3       HP9895:   tea_HP9895(un,ds,sc,ba,du,0);
   755:C        3       otherwise halt(-8) {value range error}
   756:C        3     end;  {case}
   757:C        2   end;
   758:S
```

```
 759:D   -12  1 $page$
 760:S
 761:D        1 procedure tea_BUBBLE(un:unitnum;ds:ds_type;sc:shortint);
 762:C        2   begin
 763:C        2     { NOTE THAT UMAXBYTES IS ASSIGNED AT CLEARUNIT TIME }
 764:C        2     check(sc,7,31);
 765:C        2     tea(un,dam(ds),BUBBLE_TM_name,sc,0,0,0,0,0,'',BUBBLE,F,T,T,0);
 766:C        2   end;
 767:S
 768:D        1 procedure tea_EPROM(un:unitnum;ds:ds_type;sn:shortint);
 769:C        2   begin
 770:C        2     { NOTE THAT UMAXBYTES IS ASSIGNED AT CLEARUNIT TIME }
 771:C        2     check(sn,0,MAXUNIT);
 772:C        2     tea(un,dam(ds),EPROM_TM_name,0,0,0,sn,0,0,'',EPROM,F,T,T,0);
 773:C        2   end;
 774:S
```



```
 775:D   -12  1 $page$
 776:S
 777:D        1 procedure create_temp_unitable;
 778:D        2   var
 779:D    -2  2     lunit: unitnum;
 780:C        2   begin
 781:C        2     new(temp_unitable);
 782:C        2     tea_nounit(0);  {assign one dummy entry}
 783:C        2     for lunit := 1 to maxunit do  {copy others; avoid symbol table search each time!}
 784:C        3       temp_unitable^[lunit] := temp_unitable^[0];
 785:C        2   end;
 786:S
 787:S
 788:D        1 procedure assign_and_clear_unit(lunit: unitnum);
 789:D        2   var
 790:D  -662  2     f: fib;
 791:C        2   begin
 792:C        2     if temp_unitable=nil then halt(-3); {unassigned pointer}
 793:C        2     with unitable^[lunit], f do
 794:C        3       if (letter<>RAM) or (lunit=0) then
 795:C        4         begin
 796:C        4           unitable^[lunit] := temp_unitable^[lunit];
 797:C        4           funit := lunit;
 798:C        4           call(tm, addr(f), clearunit, lunit, 0, 0);
 799:C        4           offline := uisblkd and (ioresult<>0);
 800:C        4         end; {with}
 801:C        2   end;
 802:S
 803:S
 804:D        1 procedure assign_temp_unitable;
 805:D        2   var
 806:D    -2  2     lunit: unitnum;
 807:C        2   begin
 808:C        2     if temp_unitable=nil then halt(-3);  {unassigned pointer}
 809:C        2     lockfiles;  {close all standard system files}
 810:C        2     for lunit := 0 to maxunit do
 811:C        3       assign_and_clear_unit(lunit);
 812:C        2   end;
 813:S
 814:S
 815:D        1 function sysunit_ok(system_unit: unitnum): boolean;
 816:C        2   begin
 817:C        2     sysunit := system_unit;
 818:C        2     initsysunit;
 819:C        2     with unitable^[system_unit] do
 820:C        3       sysunit_ok := uisblkd and not offline and (uvid<>'');
 821:C        2   end;
 822:S
 823:S
 824:D        1 procedure zap_assigned_unit(lunit: unitnum);
 825:C        2   begin
 826:C        2     tea_nounit(lunit);  {zap the temp_unitable entry}
 827:C        2     unitable^[lunit] := temp_unitable^[lunit];  {now zap the real one!}
 828:C        2   end;
```

```
 829:D   -12  1 $page$
 830:S
 831:D        1 function on_same_medium(lun1, lun2: unitnum): boolean;
 832:D        2   var
 833:D    -4  2     uep: ^unitentry;
 834:C        2   begin {on_same_medium}
 835:C        2     uep := addr(unitable^[lun2]);
 836:C        2     with unitable^[lun1] do
 837:C        3       on_same_medium := (sc=uep^.sc) and (ba=uep^.ba) and
 838:C        3                         (du=uep^.du) and (dv=uep^.dv) and
 839:C        3                         (letter=uep^.letter);
 840:C        2   end; {on_same_medium}
 841:S
 842:S
 843:D        1 procedure remove_extraneous_volumes(first_lun, last_lun: unitnum);
 844:D        2   var
 845:D    -1  2     first_lun_ok: boolean;
 846:D    -4  2     lun: unitnum;
 847:C        2   begin {remove_extraneous_volumes}
 848:C        2     first_lun_ok := false;
 849:C        2     while first_lun<last_lun do
 850:C        3       if first_lun_ok then
 851:C        4         begin
 852:C        4           lun := first_lun+1;
 853:C        4           with unitable^[first_lun] do
 854:C        5             while (lun<=last_lun) and not sysunit_ok(lun) do
 855:C        6               begin
 856:C        6                 if unitable^[lun].byteoffset = byteoffset+umaxbytes then
 857:C        7                   begin
 858:C        7                     umaxbytes := umaxbytes+unitable^[lun].umaxbytes;
 859:C        7                     zap_assigned_unit(lun);
 860:C        7                   end;
 861:C        6                 lun := lun+1;
 862:C        6               end; {while}
 863:C        4           first_lun := lun;
 864:C        4         end {then}
 865:C        4       else if sysunit_ok(first_lun) then
 866:C        5         first_lun_ok := true
 867:C        5       else
 868:C        5         first_lun := first_lun+1;
 869:C        2   end; {remove_extraneous_volumes}
 870:S
 871:C        1 end; {ctr}
```



```
 872:D        1 $page$
 873:S
 874:D        1 module BRstuff;  {BOOTROM stuff}
 875:S
 876:D        1   (*********************************************)
 877:D        1   (*                                           *)
 878:D        1   (*                Warning:                   *)
 879:D        1   (*   This module should not be modified!     *)
 880:D        1   (*                                           *)
 881:D        1   (*********************************************)
 882:S
 883:D        1 import
 884:D        1   sysglobals, options, ctr;
 885:S
 886:D        1 export
 887:D        1   const
 888:D        1     INTERNAL_MSUS = MSUS_type
 889:D        1       [ flpy_flags: assign_neither_flpy_unit, letter: INTERNAL,
 890:D        1         dav: dav_type[sc: -1, ba: -1, du:  0, dv: -1] ];
 891:S
 892:D        1   function internal_mini_present: boolean;
 893:D        1   procedure get_bootdevice_MSUS(var MSUS: MSUS_type);
 894:S
 895:D        1 implement {BRstuff}
 896:S
 897:D        1 type
 898:D        1   signed4     = -8..7;
 899:D        1   signed8     = -128..127;
 900:S
 901:D        1   fmt_type =  {format field in the msus byte}
 902:D        1     (f0,f1,f2,f3,f4,f5,f6,f7);
 903:S
 904:D        1   dev_type =  {device field in the msus byte}
 905:D        1     ( d0, d1, d2, d3, d4, d5, d6, d7, d8, d9,d10,d11,d12,d13,d14,d15,
 906:D        1       d16,d17,d18,d19,d20,d21,d22,d23,d24,d25,d26,d27,d28,d29,d30,d31);
 907:S
 908:D        1   BR_msus_type = {BOOTROM's mass storage unit specifier}
 909:D        1     packed record case boolean of
 910:D        1       false:  {8-bit unit number}
 911:D        1         ( fmt: fmt_type;  {directory format}
 912:D        1           dev: dev_type;  {device}
 913:D        1           un: signed8;    {8-bit unit number}
 914:D        1           sc: signed8;    {select code}
 915:D        1           ba: signed8     {bus address}  );
 916:D        1       true:  {4-bit volume / 4-bit unit number for CS80 discs}
 917:D        1         ( pad: signed8;   {format/device byte}
 918:D        1           vn4: signed4;   {4-bit volume number}
 919:D        1           un4: signed4;   {4-bit unit number}  );
 920:D        1     end; {BR_msus_type}
 921:S
 922:D        1 var
 923:D        1   ROM_ID[16382]:  {BOOTROM identification word}
 924:D        1     shortint;
 925:S
 926:D        1   ncrives[-296]:  {Maximum Unit for Internal Mini-Floppy}
 927:D        1     packed record  b: signed8;  end;
 928:S
 929:D        1   default_msus[-292]:  {boot device's msus}
 930:D        1     BR_msus_type;
```

```
931:D        1 $page$
932:S
933:D        1 function internal_mini_present: boolean;
934:C        2   begin
935:C        2     if ROM_ID<0
936:C        3       then internal_mini_present := true           {1.0 BOOTROM on 9826}
937:C        3       else internal_mini_present := ndrives.b<>-1;  {2.0 or greater BOOTROM}
938:C        2   end;
939:S
940:S
941:D        1 procedure get_bootdevice_MSUS(var MSUS: MSUS_type);
942:D        2   type
943:D        2     letter_table_type = array[dev_type] of char;
944:D        2   const
945:D        2     letter_table =  {BOOTROM dev to Pascal letter conversion table}
946:D        2       letter_table_type
947:D        2         [ INTERNAL, NODEVICE, NODEVICE, NODEVICE, HP9895,   HP8290X,  HP9885,   HP913X_A,
948:D        2           HP913X_B, HP913X_C, NODEVICE, NODEVICE, NODEVICE, NODEVICE, NODEVICE, NODEVICE,
949:D        2           CS80,     CS80,     NODEVICE, NODEVICE, NODEVICE, NODEVICE, BUBBLE  , NODEVICE,
950:D        2           NODEVICE, NODEVICE, NODEVICE, NODEVICE, NODEVICE, NODEVICE, NODEVICE, NODEVICE  ];
951:C        2   begin
952:C        2     if ROM_ID<0 then  {1.0 Boot ROM on 9826; internal minifloppy only}
953:C        3       MSUS := INTERNAL_MSUS
954:C        3     else  {2.0 or greater Boot ROM}
955:C        3       with default_msus do
956:C        4         begin
957:C        4           if fmt=f7 then  {non sector-oriented device}
958:C        5             if dev=d1
959:C        6               then MSUS.letter := SRM
960:C        6               else MSUS.letter := NODEVICE
961:C        6           else  {sector-oriented device}
962:C        5             MSUS.letter := letter_table[dev];
963:C        4           MSUS.dav.sc := sc;
964:C        4           MSUS.dav.ba := ba;
965:C        4           if MSUS.letter=CS80 then
966:C        5             begin
967:C        5               MSUS.dav.du := un4;
968:C        5               MSUS.dav.dv := vn4;
969:C        5             end  (then)
970:C        5           else
971:C        5             begin
972:C        5               MSUS.dav.du := un;
973:C        6               MSUS.dav.dv := 0;
974:C        5             end;  {else}
975:C        4         end; {with}
976:C        2   end;
977:S
978:C        1 end; {BRstuff}
```



```
979:D        1 $page$
980:S
981:D        1 module scanstuff;
982:S
983:D        1   (*******************************************)
984:D        1   (*                                         *)
985:D        1   (*                 Warning:                *)
986:D        1   (*   This module should not be modified!   *)
987:D        1   (*                                         *)
988:D        1   (*******************************************)
989:S
990:D        1 import
991:D        1   sysglobals, options, ctr;
992:S
993:D        1 export
994:S
995:D        1   procedure init_scanstuff;
996:D        1   function scanneddevice_letter(scan_dav: dav_type): char;
997:D        1   procedure get_CS80_parms(CS80dav: dav_type;
998:D        2                            var CS80dt: byte; var CS80id: integer;
999:D        2                            var CS80hardvols: shortint; var CS80mp: mp_type);
1000:S
1001:D       1 implement {scanstuff}
1002:S
1003:S
1004:D       1 type
1005:D       1   uep_type = ^unitentry;
1006:S
1007:D       1   uep_proc_type = procedure(uep: uep_type);
1008:D       1   HPIBget_amigo_ident_type = procedure(uep: uep_type; var ident: shortint);
1009:D       1   get_letter_type = procedure(uep: uep_type; ident: shortint; var letter: char);
1010:D       1   get_CS80_parms_type = procedure(var CS80dt: byte;
1011:D       1                                   var CS80id: integer;
1012:D       1                                   var CS80hardvols: shortint;
1013:D       1                                   var CS80mp: mp_type);
1014:S
1015:D       1   proc_type =
1016:D       1     packed record case integer of
1017:D       1       0: (value, slink: integer);
1018:D       1       1: (up: uep_proc_type);
1019:D       1       2: (gai: HPIBget_amigo_ident_type);
1020:D       1       3: (gl: get_letter_type);
1021:D       1       4: (gcp: get_CS80_parms_type);
1022:D       1     end;
1023:S
1024:S
1025:D       1 var
1026:D      -8   1   allocate_bkgnd_info_proc: proc_type;
1027:D     -16   1   deallocate_bkgnd_info_proc: proc_type;
1028:D     -24   1   abort_bkgnd_process_proc: proc_type;
1029:D     -32   1   HPIBcheck_sc_proc: proc_type;
1030:D     -40   1   HPIBget_amigo_ident_proc: proc_type;
1031:S
1032:D     -48   1   get_amigo_letter_proc: proc_type;
1033:D     -56   1   get_CS80_letter_proc: proc_type;
1034:D     -64   1   get_CS80_parms_proc: proc_type;
1035:S
1036:D     -65   1   bkgnd_and_dischpib_present: boolean;
```

```
1037:D   -65  1 $page$
1038:S
1039:D        1 function scanneddevice_letter(scan_dav: dav_type): char;
1040:S
1041:D        2   type
1042:D        2     amigo_class_type = (upper three bits of the first ident byte)
1043:D        2       (storage, display, data_communication, processor,
1044:D        2        stimulus, mesasurement, unassigned6, unassigned7);
1045:S
1046:D        2   var
1047:D   -58  2     ue: unitentry;
1048:D   -58  2     ident:
1049:D   -58  2       packed record case integer of
1050:D   -58  2         0: (word: shortint);
1051:D   -58  2         1: (upper_byte, lower_byte: byte);
1052:D   -58  2         2: (amigo_class: amigo_class_type);
1053:D   -60  2       end;
1054:S
1055:D    -8  2   procedure set_scanneddevice_letter(get_letter_proc: proc_type);
1056:D    -8  3     var
1057:D    -9  3       device_letter: char;
1058:C        3     begin {set_scanneddevice_letter}
1059:C        3       if get_letter_proc.value<>0 then
1060:C        4         begin
1061:C        4           call(get_letter_proc.gl, addr(ue), ident.word, device_letter);
1062:C        4           scanneddevice_letter := device_letter;
1063:C        4         end; {if}
1064:C        3     end; {set_scanneddevice_letter}
1065:S
1066:C        2   begin {scanneddevice_letter}
1067:C        2     scanneddevice_letter := NODEVICE;  {until proven otherwise}
1068:C        2     if bkgnd_and_dischpib_present then
1069:C        3       try
1070:C        4         ue.sc := scan_dav.sc;
1071:C        4         ue.ba := scan_dav.ba;
1072:C        4         ue.du := scan_dav.du;
1073:C        4         ue.dv := scan_dav.dv;
1074:C        4         call(allocate_bkgnd_info_proc.up, addr(ue));
1075:C        4         call(HPIBcheck_sc_proc.up, addr(ue));
1076:C        4         call(HPIBget_amigo_ident_proc.gai, addr(ue), ident.word);
1077:C        4         if ident.amigo_class=storage then
1078:C        5             if ident.upper_byte=2
1079:C        6               then set_scanneddevice_letter(get_CS80_letter_proc)
1080:C        6               else set_scanneddevice_letter(get_amigo_letter_proc)
1081:C        6         else if ident.amigo_class=display then
1082:C        6             scanneddevice_letter := PRINTER;
1083:C        4         call(deallocate_bkgnd_info_proc.up, addr(ue));
1084:C        4       recover
1085:C        4         call(abort_bkgnd_process_proc.up, addr(ue));
1086:C        2   end; {scanneddevice_letter}
```



```
1087:D   -65  1 $page$
1088:S
1089:D        1 procedure get_CS80_parms(CS80dav: dav_type;
1090:D        2                          var CS80dt: byte; var CS80id: integer;
1091:D        2                          var CS80hardvols: shortint; var CS80mp: mp_type);
1092:C        2   begin {get_CS80_parms}
1093:C        2     if (scanneddevice_letter(CS80dav)=CS80) and (get_CS80_parms_proc.value<>0) then
1094:C        3       call(get_CS80_parms_proc.gcp, CS80dt, CS80id, CS80hardvols, CS80mp)
1095:C        3     else
1096:C        3       begin
1097:C        3         CS80dt := 255;
1098:C        3         CS80id := 0;
1099:C        3         CS80hardvols := 0;
1100:C        3         CS80mp := medium_parameters(NODEVICE);
1101:C        3       end; {else}
1102:C        2   end; {get_CS80_parms}
1103:S
1104:S
1105:D        1 procedure init_scanstuff;
1106:S          {
1107:S            NOTE: all procedure variables are GLOBAL, so their static links are
1108:S                  guaranteed to have been cleared @ load time
1109:D        2   }
1110:C        2   begin {init_scanstuff}
1111:C        2     allocate_bkgnd_info_proc.value   := value('BKGND_ALLOCATE_BKGND_INFO');
1112:C        2     deallocate_bkgnd_info_proc.value := value('BKGND_DEALLOCATE_BKGND_INFO');
1113:C        2     abort_bkgnd_process_proc.value   := value('BKGND_ABORT_BKGND_PROCESS');
1114:C        2     HPIBcheck_sc_proc.value          := value('DISCHPIB_HPIBCHECK_SC');
1115:C        2     HPIBget_amigo_ident_proc.value   := value('DISCHPIB_HPIBGET_AMIGO_IDENT');
1116:S
1117:C        2     get_amigo_letter_proc.value      := value('AMIGODVR_GET_LETTER');
1118:C        2     get_CS80_letter_proc.value       := value('CS80DVR_GET_LETTER');
1119:C        2     get_CS80_parms_proc.value        := value('CS80DVR_GET_PARMS');
1120:S
1121:C        2     bkgnd_and_dischpib_present := (allocate_bkgnd_info_proc.value<>0) and
1122:C        2                                   (deallocate_bkgnd_info_proc.value<>0) and
1123:C        2                                   (abort_bkgnd_process_proc.value<>0) and
1124:C        2                                   (HPIBcheck_sc_proc.value<>0) and
1125:C        2                                   (HPIBget_amigo_ident_proc.value<>0);
1126:C        2   end; {init_scanstuff}
1127:S
1128:S
1129:C        1 end; {scanstuff}
```

```
1130:D        1 $page$
1131:S
1132:D        1 (program ctable)
1133:S
1134:D        1   (*********************************************)
1135:D        1   (*                Caution:                   *)
1136:D        1   (* Modify this section only if the desired   *)
1137:D        1   (* configuration cannot be achieved by       *)
1138:D        1   (* modifying the OPTIONS module.             *)
1139:D        1   (*********************************************)
1140:S
1141:D        1 import
1142:D        1   sysglobals, fs, ldr, options, ctr, BRstuff, scanstuff, bootDAMmodule;
1143:S
1144:D        1 const
1145:D        1   srmsysprefix = '/WORKSTATIONS/SYSTEM';
1146:D        1   null_dav =
1147:D        1     dav_type[sc: -1, ba: -1, du: -1, dv: -1];
1148:D        1   null_MSUS =
1149:D        1     MSUS_type[flpy_flags: assign_neither_flpy_unit, letter: NODEVICE, dav: null_dav];
1150:D        1   HP9885_default_MSUS =
1151:D        1     MSUS_type[flpy_flags: assign_neither_flpy_unit, letter: HP9885, dav: HP9885_default_dav];
1152:D        1   MSUS_array_size = 10;
1153:S
1154:D        1 type
1155:D        1   MSUS_array_type = array [1..MSUS_array_size] of MSUS_type;
1156:D        1   log_MSUS_options = (search_for_other_units, do_not_search_for_other_units);
1157:S
1158:D        1 var
1159:D   -60  1   flpy_MSUS: MSUS_array_type;
1160:D  -120  1   harddisc_MSUS: MSUS_array_type;
1161:D  -180  1   CS80tape_MSUS: MSUS_array_type;
1162:D  -186  1   scanner_MSUS: MSUS_type;
1163:S
1164:D  -190  1   local_printer_dav: dav_type;
1165:D  -194  1   SRM_dav: dav_type;
1166:D  -198  1   BUBBLE_dav: dav_type;
1167:S
1168:D  -206  1   index, bus_address, i, nvols: shortint;
1169:D  -212  1   lun, lun1, lun2: unitnum;
1170:D  -214  1   CS80dt: byte;
1171:D  -218  1   CS80id: integer;
1172:D  -220  1   CS80hardvols: shortint;
1173:D  -228  1   mp: mp_type;
1174:D  -234  1   pp: pp_type;
1175:D  -235  1   ok: boolean;
1176:S
1177:S
1178:D        1 function increment_and_test_lun: boolean;
1179:C        2   begin {increment_and_test_lun}
1180:C        2     lun := lun+1;
1181:C        2     increment_and_test_lun := lun<=last_harddisc_lun;
1182:C        2   end; {increment_and_test_lun}
```



```
1183:D  -235  1 $page$
1184:S
1185:D  -122  1 function unit_prefix_successful(dirname: fid): boolean;
1186:D  -122  2   var
1187:D  -126  2     unitnum: integer;
1188:D  -144  2     kvid: vid;
1189:C        2   begin {unit_prefix_successful}
1190:C        2     doprefix(dirname, kvid, unitnum, true);
1191:C        2     unit_prefix_successful := ioresult=ord(inoerror);
1192:C        2   end; {unit_prefix_successful}
1193:S
1194:S
1195:D        1 procedure zero_out_NA_fields(var device_MSUS: MSUS_type);
1196:D        2   const
1197:D        2     clear = true;
1198:D        2     retain = false;
1199:D        2   procedure zero_fields(sc, ba, du, dv: boolean);
1200:C        3     begin  {zero_fields}
1201:C        3       if sc then device_MSUS.dav.sc := 0;
1202:C        3       if ba then device_MSUS.dav.ba := 0;
1203:C        3       if du then device_MSUS.dav.du := 0;
1204:C        3       if dv then device_MSUS.dav.dv := 0;
1205:C        3     end;  {zero_fields}
1206:C        2   begin {zero_out_NA_fields}
1207:C        2     case device_MSUS.letter of
1208:C        3       INTERNAL:
1209:C        3         zero_fields({sc} clear , {ba} clear , {du} retain, {dv} clear );
1210:C        3       HP9885:
1211:C        3         zero_fields({sc} retain, {ba} clear , {du} retain, {dv} clear );
1212:C        3       HP9895, HP8290X, HP913X_A, HP913X_B, HP913X_C, SRM:
1213:C        3         zero_fields({sc} retain, {ba} retain, {du} retain, {dv} clear );
1214:C        3       BUBBLE:
1215:C        3         zero_fields({sc} retain, {ba} clear , {du} clear , {dv} clear );
1216:C        3       EPROM:
1217:C        3         zero_fields({sc} clear , {ba} clear , {du} clear , {dv} retain);
1218:C        3       otherwise  {includes CS80}
1219:C        3         {do nothing};
1220:C        3     end; {case}
1221:C        2   end; {zero_out_NA_fields}
1222:S
1223:S
1224:D        1 procedure assign_flpy_unit_pair(lun: unitnum; dam: ds_type; index: shortint);
1225:C        2   begin {assign_flpy_unit_pair}
1226:C        2     with flpy_MSUS[index], flpy_flags, dav do
1227:C        3       begin
1228:C        3         if assign_even_unit then
1229:C        4           begin
1230:C        4             tea_flpy(lun, letter, dam, sc, ba, du);
1231:C        4             lun := lun+1;
1232:C        4           end; {if}
1233:C        3         if assign_odd_unit then
1234:C        4           tea_flpy(lun, letter, dam, sc, ba, du+1);
1235:C        3       end; {with}
1236:C        2   end; {assign_flpy_unit_pair}
```

```
1237:D  -235  1 $page$
1238:S
1239:D    -6  1 procedure log_MSUS(MSUS: MSUS_type; log_MSUS_option: log_MSUS_options);
1240:S
1241:S
1242:D    -6  2   type
1243:D    -6  2     log_flpy_MSUS_options = (assign_both_units, assign_only_this_unit);
1244:S
1245:S
1246:D        2   procedure log_specific_MSUS(var specific_MSUS: MSUS_array_type);
1247:D        3     var
1248:D    -2  3       index: shortint;
1249:D    -3  3       found: boolean;
1250:C        3     begin {log_specific_MSUS}
1251:C        3       index := 0;
1252:C        3       repeat
1253:C        4         index := index+1;
1254:C        4         found := MSUSs_match(specific_MSUS[index], MSUS);
1255:C        4       until found or (index=MSUS_array_size);
1256:C        3       if found
1257:C        4         then MSUS.flpy_flags := specific_MSUS[index].flpy_flags {preserve}
1258:C        4         else MSUS.flpy_flags := assign_neither_flpy_unit;       {initialize}
1259:C        3       while index>1 do
1260:C        4         begin
1261:C        4           specific_MSUS[index] := specific_MSUS[index-1];
1262:C        4           index := index-1;
1263:C        4         end;  {while}
1264:C        3       specific_MSUS[1] := MSUS;
1265:C        3     end; {log_specific_MSUS}
1266:S
1267:S
1268:D        2   procedure log_flpy_MSUS(log_flpy_MSUS_option: log_flpy_MSUS_options);
1269:D        3     var
1270:D    -1  3       odd_unit: boolean;
1271:C        3     begin {log_flpy_MSUS}
1272:C        3       with MSUS.dav do
1273:C        4         begin  {since floppy units are assigned in pairs...}
1274:C        4           odd_unit := odd(du);     {remember which unit this actually is...}
1275:C        4           du := du-ord(odd_unit);  {but log only the even-numbered unit!}
1276:C        4         end; {with}
1277:C        3       log_specific_MSUS(flpy_MSUS);
1278:C        3       with flpy_MSUS[1] do  {update the flpy_flags}
1279:C        4         if log_flpy_MSUS_option=assign_both_units then
1280:C        5           flpy_flags := assign_both_flpy_units
1281:C        5         else  {set only this unit's assignment flag}
1282:C        5           if odd_unit
1283:C        6             then flpy_flags.assign_odd_unit  := true
1284:C        6             else flpy_flags.assign_even_unit := true;
1285:C        3     end; {log_flpy_MSUS}
1286:S
1287:S
1288:D        2   procedure log_harddisc_MSUS;
1289:C        3     begin {log_harddisc_MSUS}
1290:C        3       MSUS.dav.dv := 0;  {all vols will be assigned, so log only volume zero}
1291:C        3       log_specific_MSUS(harddisc_MSUS);
1292:C        3     end; {log_harddisc_MSUS}
```



```
1293:D    -6  2 $page$
1294:S
1295:D        2   function any_9895_unit_missing: boolean;
1296:D        3     var
1297:D    -4  3       temp_dav: dav_type;
1298:D    -5  3       unit_missing: boolean;
1299:C        3     begin {any_9895_unit_missing}
1300:C        3       temp_dav := MSUS.dav;
1301:C        3       temp_dav.du := 3;  {start with unit 3 and work down}
1302:C        3       repeat  {see if all four units are present}
1303:C        4         unit_missing := scanneddevice_letter(temp_dav)<>HP9895;
1304:C        4         temp_dav.du := temp_dav.du-1;
1305:C        4       until (temp_dav.du<0) or unit_missing;
1306:C        3       any_9895_unit_missing := unit_missing;
1307:C        3     end; {any_9895_unit_missing}
1308:S
1309:S
1310:D        2   procedure search_higher_numbered_CS80_units;
1311:D        3     var
1312:D    -6  3       temp_MSUS: MSUS_type;
1313:C        3     begin {search_higher_numbered_CS80_units}
1314:C        3       temp_MSUS := MSUS;
1315:C        3       with temp_MSUS, dav do
1316:C        4         if du<14 then  {potentially there are higher-numbered units}
1317:C        5           begin
1318:C        5             du := du+1;
1319:C        5             dv := 0;  {always look for volume 0}
1320:C        5             letter := scanneddevice_letter(dav);
1321:C        5             log_MSUS(temp_MSUS, search_for_other_units);  {recurse!}
1322:C        5           end; {if}
1323:C        3     end; {search_higher_numbered_CS80_units}
1324:S
1325:S
1326:D        2   procedure log_CS80_MSUS;
1327:D        3     var
1328:D    -2  3       CS80dt: byte;              {device type}
1329:D    -6  3       CS80id: integer;           {HP product number}
1330:D    -8  3       CS80hardvols: shortint;    {number of volumes}
1331:D   -16  3       CS80mp: mp_type;           {media parameters}
1332:D   -16  3     const
1333:D   -16  3       tape_dt = 2;
1334:D   -16  3       min_hd_size = 10000000;  {bytes}
1335:C        3     begin {log_CS80_MSUS}
1336:C        3       get_CS80_parms(MSUS.dav, CS80dt, CS80id, CS80hardvols, CS80mp);
1337:C        3       if CS80dt=tape_dt then
1338:C        4         log_specific_MSUS(CS80tape_MSUS)
1339:C        4       else if (CS80hardvols=1) and (CS80mp.bpt*CS80mp.tpm<min_hd_size) then
1340:C        5         log_flpy_MSUS(assign_only_this_unit)
1341:C        5       else
1342:C        5         log_harddisc_MSUS;
1343:C        3     end; {log_CS80_MSUS}
```

```
1344:D     -6  2 $page$
1345:S
1346:C         2   begin {log_MSUS}
1347:S
1348:C         2     zero_out_NA_fields(MSUS);
1349:S
1350:C         2     case MSUS.letter of
1351:S
1352:C         3       INTERNAL, HP8290X, HP9885:
1353:C         3         log_flpy_MSUS(assign_both_units);
1354:S
1355:C         3       HP913X_A, HP913X_B, HP913X_C:
1356:C         3         log_harddisc_MSUS;
1357:S
1358:C         3       HP9895:
1359:C         3         if any_9895_unit_missing then  {ultimately assign only two units }
1360:C         4           log_flpy_MSUS(assign_both_units)
1361:C         4         else  {ultimately assign all four units (probably a 913X)}
1362:C         4           log_harddisc_MSUS;
1363:S
1364:C         3       CS80:
1365:C         3         begin
1366:C         3           if log_MSUS_option=search_for_other_units then
1367:C         4             search_higher_numbered_CS80_units;
1368:C         3           log_CS80_MSUS;  {distinguishes tapes, floppies, & hard discs!}
1369:C         3         end;
1370:S
1371:C         3       SRM:
1372:C         3         SRM_dav := MSUS.dav;
1373:S
1374:C         3       PRINTER:
1375:C         3         local_printer_dav := MSUS.dav;
1376:S
1377:C         3       BUBBLE:
1378:C         3         BUBBLE_dav := MSUS.dav;
1379:S
1380:C         3       otherwise
1381:C         3         {do nothing};
1382:S
1383:C         3     end; {case}
1384:S
1385:C         2   end; {log_MSUS}
```



```
1386:D   -235  1 $page$
1387:S
1388:C         1 begin {ctable}
1389:S
1390:C         1   { various initializations }
1391:S
1392:C         1   call(cleariohook); {init IO cards in case the BOOTROM drivers touched them}
1393:S
1394:C         1   init_scanstuff;
1395:S
1396:C         1   for index := 1 to MSUS_array_size do
1397:C         2     begin
1398:C         2       flpy_MSUS[index]     := null_MSUS;
1399:C         2       harddisc_MSUS[index] := null_MSUS;
1400:C         2       CS80tape_MSUS[index] := null_MSUS;
1401:C         2     end;  {for}
1402:S
1403:C         1   SRM_dav                  := SRM_default_dav;     {overridden if bootdevice}
1404:S
1405:C         1   BUBBLE_dav               := BUBBLE_default_dav;  {overridden if bootdevice}
1406:S
1407:C         1   if local_printer_option=HPIB
1408:C         2     then local_printer_dav := local_HPIB_printer_default_dav   {scan may override}
1409:C         2     else local_printer_dav := local_RS232_printer_default_dav; {scan may override}
1410:S
1411:S
1412:C         1   { log the default 9885 floppy pair, since HP-IB scanning won't include it }
1413:S
1414:C         1   log_MSUS(HP9885_default_MSUS, search_for_other_units);
1415:S
1416:S
1417:C         1   { scan the HP-IB's for mass storage devices and possibly a local printer }
1418:S
1419:C         1   with scanner_MSUS, dav do
1420:C         2     for index := 1 to sc_list_length do
1421:C         3       for bus_address := 0 to 7 do
1422:C         4         begin
1423:C         4           sc := sc_list[index];
1424:C         4           ba := bus_address;
1425:C         4           du := 0;
1426:C         4           dv := 0;
1427:C         4           letter := scanneddevice_letter(dav);
1428:C         4           log_MSUS(scanner_MSUS, search_for_other_units);
1429:C         4         end; {for}
1430:S
1431:S
1432:C         1   { log internal mini if present, since HP-IB scanning didn't include it }
1433:S
1434:C         1   if internal_mini_present then
1435:C         2     log_MSUS(INTERNAL_MSUS, search_for_other_units);
1436:S
1437:S
1438:C         1   { get the bootdevice MSUS & log it }
1439:S
1440:C         1   get_bootdevice_MSUS(bootdev_MSUS);
1441:C         1   zero_out_NA_fields(bootdev_MSUS);  {for tea routine comparisons}
1442:C         1   log_MSUS(bootdev_MSUS, do_not_search_for_other_units);
1443:C         1   bootdev_lun := 0;  {set otherwise if/when bootdevice is assigned in tea}
```

```
1444:C        1 $page$
1445:S
1446:C        1   { Create a temporary table & fill it with dummy entries }
1447:S
1448:C        1   create_temp_unitable;
1449:S
1450:S
1451:C        1   { standard assignemnts: avoid changing }
1452:S
1453:C        1   tea_memory_volume_dam(primary_dam);
1454:S
1455:C        1   tea_crt( 1);
1456:C        1   tea_kbd( 2);
1457:S
1458:C        1   assign_flpy_unit_pair( 3, primary_dam, {flpy_MSUS[]} 1);
1459:S
1460:C        1   with SRM_dav do
1461:C        2     tea_srm( 5, sc, ba, du);
1462:S
1463:C        1   with local_printer_dav do
1464:C        2     tea_local_printer( 6, sc, ba, {uvid} 'PRINTER', local_printer_timeout);
1465:S
1466:S
1467:C        1   { optional floppy unit pairs }
1468:S
1469:C        1   for index := 2 to floppy_unit_pairs do
1470:C        2     assign_flpy_unit_pair(7+(index-2)*2, primary_dam, {flpy_MSUS[]} index);
1471:S
```



```
1472:C        1 $page$
1473:S
1474:C        1   { local hard discs }
1475:S
1476:C        1   $if true$
1477:S
1478:C        1     lun := first_harddisc_lun-1;
1479:S
1480:C        1     for index := 1 to MSUS_array_size do
1481:C        2       with harddisc_MSUS[index], dav do
1482:C        3         case letter of
1483:S
1484:C        4           HP9895: {9895 ident with all four units present; probably a HP913X}
1485:C        4             for i := 0 to 3 do
1486:C        5               if increment_and_test_lun then
1487:C        6                 tea_HP9895(lun, primary_dam, sc, ba, {du} i, {block_offset} 0);
1488:S
1489:C        4           HP913X_A, HP913X_B, HP913X_C:
1490:C        4             begin
1491:C        4               mp := medium_parameters(letter);
1492:C        4               pp := partitioning_parameters(letter);
1493:C        4               nvols := number_vols(mp, pp);
1494:C        4               for i := 0 to nvols-1 do
1495:C        5                 if increment_and_test_lun then
1496:C        6                   tea_amigo_sv(lun, primary_dam, sc, ba, du,
1497:C        6                                vol_offset(i, nvols, mp),
1498:C        6                                letter,
1499:C        6                                vol_bytes(i, nvols, mp));
1500:C        4             end;
1501:S
1502:C        4           CS80:
1503:C        4             begin
1504:C        4               pp := partitioning_parameters(letter);
1505:C        4               repeat
1506:C        5                 get_CS80_parms(dav, CS80dt, CS80id, CS80hardvols, mp);
1507:C        5                 if mp.tpm=1 then  {track partitioning info unavailable...}
1508:C        6                   mp := block_boundaries(mp);  {will have to fake it!}
1509:C        5                 nvols := number_vols(mp, pp);
1510:C        5                 for i := 0 to nvols-1 do
1511:C        6                   if increment_and_test_lun then
1512:C        7                     tea_CS80_mv(lun, primary_dam, sc, ba, du, dv,
1513:C        7                                 vol_offset(i, nvols, mp),
1514:C        7                                 {devid} CS80id,
1515:C        7                                 vol_bytes(i, nvols, mp));
1516:C        5                 dv := dv+1;
1517:C        5               until dv>=CS80hardvols;
1518:C        4             end;
1519:S
1520:C        4           otherwise
1521:C        4             {no local hard disc logged};
1522:S
1523:C        4         end; {case}
1524:S
1525:C        1   $end$  { local hard discs }
1526:S
1527:S
1528:C        1   { CS80 tapes }
1529:S
1530:C        1   $if true$
1531:S
```

```
1532:C        1     with CS80tape_MSUS[1], dav do
1533:C        2       if letter=CS80 then
1534:C        3         tea_CS80_sv(41, LIF_dam, sc, ba, du, dv);
1535:S
1536:C        1     with CS80tape_MSUS[2], dav do
1537:C        2       if letter=CS80 then
1538:C        3         tea_CS80_sv(42, LIF_dam, sc, ba, du, dv);
1539:S
1540:C        1   $end$  { CS80 tapes }
1541:S
1542:S
1543:C        1   { secondary directory access method entries for highest priority floppies }
1544:S
1545:C        1   assign_flpy_unit_pair(43, secondary_dam, {flpy_MSUS[]} 1);
1546:S
1547:S
1548:C        1   { duplicate entries for prefixing down the SRM }
1549:S
1550:C        1   (*************************************************************************)
1551:C        1   (*  NOTE:  Additional duplicate SRM entries may be assigned here, then *)
1552:C        1   (*    prefixed down below after assigning the temp_unitable.  However  *)
1553:C        1   (*    for correct behavior in assigning the system unit, specifically  *)
1554:C        1   (*    if booting off the SRM, unit #45 must be the assigned AFTER all  *)
1555:C        1   (*    the other SRM units have been assigned!                          *)
1556:C        1   (*************************************************************************)
1557:S
1558:C        1   with SRM_dav do
1559:C        2     begin
1560:C        2       {  tea_srm(46, sc, ba, du);  {free}
1561:C        2       tea_srm(45, sc, ba, du);  {for possible use as the system unit}
1562:C        2     end; {with}
1563:S
1564:S
1565:C        1   { secondary directory access method entries for additional floppies }
1566:S
1567:C        1   assign_flpy_unit_pair(47, secondary_dam, {flpy_MSUS[]} 2);
1568:S
1569:C        1   assign_flpy_unit_pair(49, secondary_dam, {flpy_MSUS[]} 3);
```



```
1570:C        1 $page$
1571:S
1572:C        1   { templates for "manually" specifying mass storage table entry assignments }
1573:S
1574:S
1575:S            $if false$ { internal minifloppy in a 9826/9836 }
1576:S              tea_mini( 3, primary_dam, {du} 0);
1577:S              tea_mini( 4, primary_dam, {du} 1);
1578:C        1   $end$
1579:S
1580:S            $if false$ { HP8290X, HP9121, or the floppy in an HP913X }
1581:S              tea_HP8290X( 3, primary_dam, {sc}  7, {ba} 0, {du} 0);
1582:S              tea_HP8290X( 4, primary_dam, {sc}  7, {ba} 0, {du} 1);
1583:C        1   $end$
1584:S
1585:S            $if false$ { HP9895 }
1586:S              tea_HP9895( 7, primary_dam, {sc}  7, {ba} 0, {du} 0, {block_offset} 0);
1587:S              tea_HP9895( 8, primary_dam, {sc}  7, {ba} 0, {du} 1, {block_offset} 0);
1588:C        1   $end$
1589:S
1590:S            $if false$ { HP913X (four volume 9895 look-a-like version) }
1591:S              for i := 0 to 3 do
1592:S                tea_HP9895(11+i, primary_dam, {sc}  7, {ba} 0, {du} i, {block_offset} 0);
1593:C        1   $end$
1594:S
1595:S            $if false$ { HP913X_A (5 Mbyte single volume version) }
1596:S              mp := medium_parameters(HP913X_A);
1597:S              nvols := 4;
1598:S              for i := 0 to nvols-1 do
1599:S                tea_amigo_sv(11+i, primary_dam, {sc}  7, {ba} 0, {du} 0,
1600:S                                   vol_offset(i, nvols, mp),
1601:S                                   HP913X_A,
1602:S                                   vol_bytes(i, nvols, mp));
1603:C        1   $end$
1604:S
1605:S            $if false$ { HP913X_B (10 Mbyte single volume version) }
1606:S              mp := medium_parameters(HP913X_B);
1607:S              nvols := 9;
1608:S              for i := 0 to nvols-1 do
1609:S                tea_amigo_sv(11+i, primary_dam, {sc}  7, {ba} 0, {du} 0,
1610:S                                   vol_offset(i, nvols, mp),
1611:S                                   HP913X_B,
1612:S                                   vol_bytes(i, nvols, mp));
1613:C        1   $end$
1614:S
1615:S            $if false$ { HP913X_C (15 Mbyte single volume version) }
1616:S              mp := medium_parameters(HP913X_C);
1617:S              nvols := 14;
1618:S              for i := 0 to nvols-1 do
1619:S                tea_amigo_sv(11+i, primary_dam, {sc}  7, {ba} 0, {du} 0,
1620:S                                   vol_offset(i, nvols, mp).
1621:S                                   HP913X_C,
1622:S                                   vol_bytes(i, nvols, mp));
1623:C        1   $end$
```

```
1624:C        1 $page$
1625:S
1626:S            $if false$ { current CS/80 discs "soft" partitioned by the host }
1627:S                CS80id := 7908; nvols := 16; mp.tpm :=  5* 370; mp.bpt := 35*256;  {7908}
1628:S              { CS80id := 7911; nvols := 27; mp.tpm :=  3* 572; mp.bpt := 64*256;  {7911}
1629:S              { CS80id := 7912; nvols := 30; mp.tpm :=  7* 572; mp.bpt := 64*256;  {7912}
1630:S              { CS80id := 7914; nvols := 30; mp.tpm :=  7*1152; mp.bpt := 64*256;  {7914}
1631:S              { CS80id := 7933; nvols := 30; mp.tpm := 13*1321; mp.bpt := 92*256;  {7933}
1632:S              { CS80id := 7935; nvols := 30; mp.tpm := 13*1321; mp.bpt := 92*256;  {7935}
1633:S              { mp := block_boundaries(mp);  {override track boundary partitioning}
1634:S              for i := 0 to nvols-1 do
1635:S                tea_CS80_mv(11+i, primary_dam, {sc}  7, {ba} 0, {du} 0, {dv} 0,
1636:S                                vol_offset(i, nvols, mp),
1637:S                                {devid} CS80id,
1638:S                                vol_bytes(i, nvols, mp));
1639:C        1   $end$
1640:S
1641:S
1642:S            $if false$ { current CS/80 discs "hard" partitioned by the device }
1643:S                CS80hardvols :=  3;
1644:S                for i := 0 to CS80hardvols-1 do
1645:S                  tea_CS80_sv(11+i, primary_dam, {sc}  7, {ba} 0, {du} 0, {dv} i);
1646:C        1   $end$
1647:S
1648:S
1649:S            $if false$ { Command Set/80 floppy }
1650:S              tea_CS80_sv( 3, primary_dam, {sc}  7, {ba} 0, {du} 0, {dv} 0);
1651:C        1   $end$
1652:S
1653:S
1654:S            $if false$ { Command Set/80 tape }
1655:S              tea_CS80_sv(41, LIF_dam, {sc}  7, {ba} 0, {du} 1, {dv} 0);
1656:C        1   $end$
1657:S
1658:S
1659:S            $if false$ { BUBBLE memory }
1660:S              {watch for conflicting uses of unit 42}
1661:S              {BUBBLE_DAV.SC default is 30 but may have been changed to boot SC}
1662:S              tea_BUBBLE(42,primary_dam,BUBBLE_dav.SC);
1663:C        1   $end$
1664:S
1665:S
1666:S            $if false$ { EPROM DISC }
1667:S              {watch for conflicting uses of unit 42}
1668:S              tea_EPROM(42,primary_dam,{ sequence number } 0);
1669:C        1   $end$
1670:S
1671:S
1672:C        1   { end of templates }
```



```
1673:C        1 $page$
1674:S
1675:C        1   { assign the new unitable and unitclear all units }
1676:S
1677:C        1   assign_temp_unitable;
1678:S
1679:S
1680:C        1   { prefix the primary and secondary SRM unit entries }
1681:S
1682:C        1   if not unit_prefix_successful('#S:/') then
1683:C        2     {do nothing};  {tries to set up uvid for possible default unit assignment below}
1684:S
1685:C        1   { if not unit_prefix_successful('#46:/?') then zap_assigned_unit(46);  {free}
1686:S
1687:C        1   if not unit_prefix_successful('#45:'+srmsysprefix+srmnode(unitable^[45].sc)) then
1688:C        2     if not unit_prefix_successful('#45:'+srmsysprefix) then
1689:C        3       zap_assigned_unit(45);
1690:S
1691:S
1692:C        1   { remove extraneous local hard disc entries if necessary }
1693:S
1694:C        1   lun2 := first_harddisc_lun;
1695:C        1   while lun2<last_harddisc_lun do
1696:C        2     begin
1697:C        2       lun1 := lun2;
1698:C        2       repeat
1699:C        3         lun2 := lun2+1;
1700:C        3       until (lun2>last_harddisc_lun) or not on_same_medium(lun1, lun2);
1701:C        2       pp := partitioning_parameters(unitable^[lun1].letter);
1702:C        2       if pp.mnv<-1 then
1703:C        3         remove_extraneous_volumes(lun1, lun2-1);
1704:C        2     end; {while}
1705:S
1706:S
1707:C        1   { assign the system unit }
1708:S
1709:C        1   if specified_system_unit<>0 then
1710:C        2     ok := sysunit_ok(specified_system_unit)
1711:C        2   else if (bootdev_lun<>0) and (unitable^[bootdev_lun].umaxbytes>300000) then
1712:C        3     ok := sysunit_ok(bootdev_lun)
1713:C        3   else  {search for a more suitable system unit}
1714:C        3     begin
1715:C        3       index := 0;
1716:C        3       repeat
1717:C        4         index := index+1;
1718:C        4         ok := sysunit_ok(sysunit_list[index]);
1719:C        4       until ok or (index>=sysunit_list_length);
1720:C        3       if not ok then  {revert back to boot device, hoping it was identified}
1721:C        4         ok := sysunit_ok(bootdev_lun);
1722:C        3     end; {else}
1723:S
1724:S
1725:C        1   { special case for default unit assignment }
1726:S
1727:C        1   if sysunit=45 then  {set the default unit to the primary SRM unit entry}
1728:C        2     dkvid := unitable^[5].uvid;
```

```
172:C        1 $page$
173:S
1731:C       1   { re-open the standard system files }
1732:S
1733:C       1   openfiles;
1734:S
1735:C       1 end. {ctable}

No errors. No warnings.
             ***** Nonstandard language features enabled *****
```

Description

DC_DRV provides low-level driver support.

Usage

DC_DRV is used for the following interfaces:

- 98628 (Data Comm)
- 98629 (SRM)

Requirements

DC and the I/O library kernel (IODECLARATIONS, etc.)

```
Pascal [Rev 3.0M  6/ 4/84] DC_DRV.TEXT            26-Dec-84 21:28:45 Page 1   *** File name: DC_DRV ***

     1:S                                                    (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1983.
     4:S          All rights are reserved.  Copying or other
     5:S          reproduction of this program except for archival
     6:S          purposes is prohibited without the prior
     7:S          written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                       RESTRICTED RIGHTS LEGEND
    11:S
    12:S          Use, duplication, or disclosure by the Government
    13:S          is subject to restrictions as set forth in
    14:S          paragraph (b) (3) (B) of the Rights in Technical
    15:S          Data and Computer Software clause in
    16:S          DAR 7-104.9(a).
    17:S
    18:S          HEWLETT-PACKARD COMPANY
    19:D        0 Fort Collins, Colorado                        *)
    20:S
    21:S
    22:D        0 $MODCAL ON$
    23:D        0 $PARTIAL_EVAL ON$
    24:D        0 $STACKCHECK ON$
    25:D        0 $RANGE OFF$
    26:D        0 $DEBUG OFF$
    27:D        0 $OVFLCHECK OFF$
```

```
Pascal [Rev 3.0M  6/ 4/84] DC_DRV.TEXT            26-Dec-84 21:28:45 Page 2   *** File name: DC_DRV ***

    28:D        0 $PAGE$
    29:D        0 $SEARCH 'IOLIB:KERNEL'$
    30:S
    31:D        0 (*************************************************************************)
    32:D        0 (*                                                                       *)
    33:D        0 (*      RELEASED        VERSION         3.0                              *)
    34:D        0 (*                                                                       *)
    35:D        0 (*************************************************************************)
    36:D        0 (*                                                                       *)
    37:D        0 (*                                                                       *)
    38:D        0 (*      IOLIB           DC_DRIVERS                                       *)
    39:D        0 (*                                                                       *)
    40:D        0 (*                                                                       *)
    41:D        0 (*************************************************************************)
    42:D        0 (*                                                                       *)
    43:D        0 (*                                                                       *)
    44:D        0 (*      library     -   IOLIB                                            *)
    45:D        0 (*      name        -   DC_DRIVERS                                       *)
    46:D        0 (*      module(s)   -   init_dc                                          *)
    47:D        0 (*                  -   extd                                             *)
    48:D        0 (*                                                                       *)
    49:D        0 (*      author      -                                                    *)
    50:D        0 (*      phone       -                                                    *)
    51:D        0 (*                                                                       *)
    52:D        0 (*      date        -   Oct  20 , 1981                                   *)
    53:D        0 (*      update      -   Aug  11 , 1983                                   *)
    54:D        0 (*      release     -   ????????????                                     *)
    55:D        0 (*                                                                       *)
    56:D        0 (*      source      -   IOLIB:DC_DRV.TEXT                                *)
    57:D        0 (*      object      -   IOLIB:DC_DRV.CODE                                *)
    58:D        0 (*                                                                       *)
    59:D        0 (*                                                                       *)
    60:D        0 (*************************************************************************)
```

```
    61:D    0 $PAGE$
    62:D    0 (*************************************************************************)
    63:D    0 (*                                                                       *)
    64:D    0 (*                                                                       *)
    65:D    0 (*      BUG FIX HISTORY          - after release 1.0                     *)
    66:D    0 (*                                                                       *)
    67:D    0 (*                                                                       *)
    68:D    0 (*      BUG #   BY  / ON         LOC             DESCRIPTION            *)
    69:D    0 (*      -----   ----------       --------------  ---------------------  *)
    70:D    0 (*                                                                       *)
    71:D    0 (*      1283    -----------      idc_wtc         IOCONTROL(s,0,1) gives *)
    72:D    0 (*              01/08/82                         error, but should work.*)
    73:D    0 (*                                               IOCONTROL(s,513,x)     *)
    74:D    0 (*                                               would give an error.   *)
    75:D    0 (*                                                                       *)
    76:D    0 (*      zzzz    -----------      idc_init        An error during init   *)
    77:D    0 (*              06/16/82         init_dc         can blow away the work *)
    78:D    0 (*                                               station.               *)
    79:D    0 (*                                                                       *)
    80:D    0 (*      bbbb    -----------      init_dc         No bug sheet.          *)
    81:D    0 (*              07/12/82                         Setting up a machine   *)
    82:D    0 (*                                               number ID if card is a *)
    83:D    0 (*                                               Ganglia.               *)
    84:D    0 (*                                                                       *)
    85:D    0 (*      cccc    -----------      init_dc         No bug sheet.          *)
    86:D    0 (*              08/16/82         intdc           Hunting for 98629 and  *)
    87:D    0 (*                                               98628 dsndl.           *)
    88:D    0 (*                                               See IODECLARATIONS also.*)
    89:D    0 (*                                                                       *)
    90:D    0 (*      367     -----------      dc_initialize   Allow eXecute of driver*)
    91:D    0 (*              09/22/82                         and have it install    *)
    92:D    0 (*                                               itself in the system.  *)
    93:D    0 (*                                                                       *)
    94:D    0 (*      uuuu    -----------      init_dc         Differentiate between  *)
    95:D    0 (*              09/28/82                         628 and 629 card for   *)
    96:D    0 (*                                               the card type.         *)
    97:D    0 (*                                               See IODECLARATIONS also.*)
    98:D    0 (*                                                                       *)
    99:D    0 (*              ---------        init_dc         Remove machine # setup *)
   100:D    0 (*************************************************************************)
```



```
   101:D    0 $PAGE$
   102:D    0 (*************************************************************************)
   103:D    0 (*                                                                       *)
   104:D    0 (*                                                                       *)
   105:D    0 (*      This is the source code for an external procedures library       *)
   106:D    0 (*      to be used for general purpose interfacing on the HP 9826.       *)
   107:D    0 (*                                                                       *)
   108:D    0 (*      The library consists of 3 primary sets of modules -              *)
   109:D    0 (*                                                                       *)
   110:D    0 (*              1.      KERNEL modules                                   *)
   111:D    0 (*              2.      driver modules                                   *)
   112:D    0 (*              3.      IOLIB  modules                                   *)
   113:D    0 (*                                                                       *)
   114:D    0 (*      The KERNEL modules consist of the following modules -            *)
   115:D    0 (*                                                                       *)
   116:D    0 (*              1.      iodeclarations  ( contains static r/w space )    *)
   117:D    0 (*              2.      iocomasm                                         *)
   118:D    0 (*              3.      general_0       ( initialization & low level    *)
   119:D    0 (*                                        routines like ioread/iowrite) *)
   120:D    0 (*      The KERNEL modules also have an executable program segement      *)
   121:D    0 (*      that gets executed at the time it is loaded.  This program       *)
   122:D    0 (*      initializes the static read/write memory.  This program also     *)
   123:D    0 (*      allocates the temporary storage for any card that exists -       *)
   124:D    0 (*      independent of whether there is or is not a driver for it.       *)
   125:D    0 (*                                                                       *)
   126:D    0 (*      The driver modules consist of the actual assembly or PASCAL      *)
   127:D    0 (*      routines that deal with a specific interface card.  There is     *)
   128:D    0 (*      also an executable program segment for each driver module.       *)
   129:D    0 (*      This program searches the select code table in the static r/w    *)
   130:D    0 (*      initialized by the KERNEL general_0 module for all select codes *)
   131:D    0 (*      that have the right interface card ( HPIB drivers will search    *)
   132:D    0 (*      for the 98624 interface ).  This program will then set up the    *)
   133:D    0 (*      driver tables to point to the correct drivers.                   *)
   134:D    0 (*                                                                       *)
   135:D    0 (*      The rest of the IOLIB modules are high-level modules that are    *)
   136:D    0 (*      used by an end user in his/her application program.              *)
   137:D    0 (*                                                                       *)
   138:D    0 (*      The KERNEL and some set of driver modules will exist in the      *)
   139:D    0 (*      SYSTEM.INITLIB file as object code ( not EXPORT text ).  The     *)
   140:D    0 (*      export text will reside on the SYSTEM.LIBRARY file.  The rest    *)
   141:D    0 (*      of the library will reside on the SYSTEM.LIBRARY.                *)
   142:D    0 (*                                                                       *)
   143:D    0 (*************************************************************************)
```

```
144:D    0 $PAGE$
145:D    0 (*************************************************************************)
146:D    0 (*                                                                       *)
147:D    0 (*      REFERENCES :                                                     *)
148:D    0 (*                                                                       *)
149:D    0 (*                                                                       *)
150:D    0 (*      1.  9826 I/O Designers Guide          ( ---------- )           *)
151:D    0 (*                                                                       *)
152:D    0 (*      2.  68000 Manual                      ( Motorola )             *)
153:D    0 (*                                                                       *)
154:D    0 (*      3.  Pascal alpha site ERS             ( ---------- )           *)
155:D    0 (*                                                                       *)
156:D    0 (*      4.  Pascal I/O Library ERS            ( ---------- )           *)
157:D    0 (*                                                                       *)
158:D    0 (*      5.  9826 HPL EIO & IOD listings       ( ---------- )           *)
159:D    0 (*                                                                       *)
160:D    0 (*      6.  9826 HPL Misc. I/O Doc.           ( ---------- )           *)
161:D    0 (*                                                                       *)
162:D    0 (*      7.  9826 card documentation           ( Mfg. Specs. )          *)
163:D    0 (*                                                                       *)
164:D    0 (*      8.  Pascal I/O Library IRS            ( ---------- )           *)
165:D    0 (*                                                                       *)
166:D    0 (*      9.  98628 Data comm documentation     ( ---------- )           *)
167:D    0 (*                                                                       *)
168:D    0 (*                                                                       *)
169:D    0 (*                                                                       *)
170:D    0 (*************************************************************************)
```



```
171:D    0 $PAGE$
172:D    0 PROGRAM dc_initialize ( INPUT , OUTPUT );
```

```
173:D        1 $PAGE$
174:D        1 (*************************************************************************)
175:D        1 (*                                                                       *)
176:D        1 (*                                                                       *)
177:D        1 (*      DATA COMM DRIVERS                                                *)
178:D        1 (*                                                                       *)
179:D        1 (*                                                                       *)
180:D        1 (*************************************************************************)
181:D        1 EXTERNAL MODULE extdc;
182:S                  { by       -------------
183:S                    date     10/20/81
184:S                    update   11/03/81
185:S
186:S                    purpose This module is a declaration of the importation text for
187:S                            the external drivers.
188:S
189:S                    note    The assembly language code that is imported needs to be
190:S                            called 'extdc'.  The routines need to be called
191:S                            'extdc_@@@@@@'
192:D        1           }
193:S
194:D        1 IMPORT  sysglobals, iodeclarations;
195:S
196:D        1 EXPORT
197:S
198:D        1   PROCEDURE alvinit         ( temp : ANYPTR );
199:D        1   PROCEDURE alvinisr        ( temp : PISRIB );
200:D        1   PROCEDURE enter_data      ( temp : ANYPTR ;  x     : ANYPTR ;
201:D        2                                                 VAR c: INTEGER );
202:D        1   PROCEDURE output_data     ( temp : ANYPTR ;  x     : ANYPTR ;
203:D        2                                                 cnt   : INTEGER );
204:D        1   PROCEDURE output_end      ( temp : ANYPTR );
205:D        1   PROCEDURE direct_status   ( temp : ANYPTR ;  reg   : io_word;
206:D        2                                                 VAR x : io_word);
207:D        1   PROCEDURE direct_control  ( temp : ANYPTR ;  reg   : io_word;
208:D        2                                                 val   : io_word );
209:D        1   PROCEDURE control_bfd     ( temp : ANYPTR ;  reg   : io_word;
210:D        2                                                 val   : io_word );
211:D        1   PROCEDURE start_tfr_in    ( temp : ANYPTR );
212:D        1   PROCEDURE start_tfr_out   ( temp : ANYPTR );
213:S
214:D        1 END; { of extdc }
```



```
215:D        1 $PAGE$
216:D        1 (*************************************************************************)
217:D        1 (*                                                                       *)
218:D        1 (*                                                                       *)
219:D        1 (*      DATA COMM DRIVERS                                                *)
220:D        1 (*                                                                       *)
221:D        1 (*                                                                       *)
222:D        1 (*************************************************************************)
223:D        1 MODULE intdc;
224:S                  { by       -------------
225:S                    date     10/20/81
226:S                    update   08/16/82
227:S
228:S                    purpose This module contains the internal drivers.
229:S
230:D        1           }
231:S
232:D        1 IMPORT  sysglobals ,
233:D        1         iodeclarations ;
234:S
235:D        1 EXPORT
236:S
237:D        1   TYPE  dc_err_type     = PACKED ARRAY                          { zzzz TM 6/16/82 }
238:D        1                         [minrealisc..maxrealisc] OF io_word ; { zzzz TM 6/16/82 }
239:D        1         dc_err_ptr      = ^dc_err_type ;                        { zzzz TM 6/16/82 }
240:S
241:D    -1  1   VAR   dc_init_fault   : BOOLEAN ;                            { zzzz TM 6/16/82 }
242:D    -6  1         dc_error        : dc_err_ptr ;                          { zzzz TM 6/16/82 }
243:S
244:D    -6  1   PROCEDURE idc_init  ( temp : ANYPTR );
245:D    -6  1   PROCEDURE idc_isr   ( temp : PISRIB );
246:D    -6  1   PROCEDURE idc_rdb   ( temp : ANYPTR ;  VAR x : CHAR);
247:D    -6  1   PROCEDURE idc_wtb   ( temp : ANYPTR ;  val   : CHAR);
248:D    -6  1   PROCEDURE idc_rdw   ( temp : ANYPTR ;  VAR x : io_word);
249:D    -6  1   PROCEDURE idc_wtw   ( temp : ANYPTR ;  val   : io_word);
250:D        1   PROCEDURE idc_rds   ( temp : ANYPTR ;  reg   : io_word;
251:D    -6  2                                          VAR x : io_word);
252:D        1   PROCEDURE idc_wtc   ( temp : ANYPTR ;  reg   : io_word;
253:D    -6  2                                          val   : io_word );
254:D    -6  1   PROCEDURE idc_tfr   ( temp : ANYPTR ;  bcb   : ANYPTR );
255:S
256:D    -6  1 IMPLEMENT
257:S
258:S
259:D    -6  1   IMPORT  isr ,
260:D    -6  1           general_0 ,
261:D    -6  1           extdc ;
262:S
263:D        1   PROCEDURE idc_init  ( temp : ANYPTR );
264:D    -2  2   VAR dummyword     : io_word;
265:D    -6  2       dummy_isc     : INTEGER;                                  { bbbb TM 7/12/82 }
266:C        2   BEGIN
267:C        2     control_bfd ( temp , 0 , 1 );
268:S
269:C        2     dummy_isc := io_find_isc(temp);                             { bbbb TM 7/12/82 }
270:S
271:C        2     { make sure card is still async }
272:S
273:C        2     direct_status( temp , 3 , dummyword );
274:C        2     WITH isc_table[ dummy_isc ] DO BEGIN
```

```
275:C        3       card_id := hp_datacomm;                                  {      TM 7/8/82 }
276:C        3       IF dummyword = 1 THEN card_id := hp98628_async;          {      TM 7/8/82 }
277:C        3       IF dummyword = 2 THEN card_id := hp98628_dsndl;          { cccc TM 8/16/82 }
278:C        3     END; { of WITH DO BEGIN }                                  {      TM 7/8/82 }
279:S
280:C        2     IF dc_init_fault                                           { zzzz TM 6/16/82 }
281:C        3       THEN BEGIN                                               { zzzz TM 6/16/82 }
282:C        3         IF dc_error^[dummy_isc]<>0                             { zzzz TM 6/16/82 }
283:C        4           THEN BEGIN                                           { zzzz TM 6/16/82 }
284:C        4             io_escape(dc_error^[dummy_isc],                    { zzzz TM 6/16/82 }
285:C        4                       dummy_isc);                              { zzzz TM 6/16/82 }
286:C        4           END; { of IF dc_error <> 0 }                         { zzzz TM 6/16/82 }
287:C        3       END; { of IF dc_init_fault }                             { zzzz TM 6/16/82 }
288:S
289:C        2     { set up Ganglia card ID }                                 { bbbb TM 7/12/82 }
290:C        2     IF ioread_byte(dummy_isc,HEX('402F'))=3                    { bbbb TM 7/12/82 }
291:C        3       THEN BEGIN                                               { bbbb TM 7/12/82 }
292:C        3         { card is ganglia }                                    { bbbb TM 7/12/82 }
293:C        3         isc_table[ dummy_isc ].card_id := hp98629;             { cccc TM 8/16/82 }
294:C        3       {   iowrite_byte(dummy_isc ,                             { bbbb TM 7/12/82 }
295:C        3       {     HEX('4061')   , io_model_number DIV 256 );         { bbbb TM 7/12/82 }
296:C        3       { iowrite_byte(dummy_isc ,                               { bbbb TM 7/12/82 }
297:C        3       {     HEX('4061')  + 2 , io_model_number MOD 256 );      { bbbb TM 7/12/82 }
298:C        3       END; { of IF }                                           { bbbb TM 7/12/82 }
299:S
300:S
301:C        2   END; { of idc_init }
302:S
303:S
304:S
305:D        1   PROCEDURE idc_isr   ( temp : PISRIB );
306:C        2   BEGIN
307:C        2     alvinisr( temp );
308:C        2   END; { of idc_isr }
309:S
310:S
311:S
312:D        1   PROCEDURE idc_rdb   ( temp : ANYPTR ;  VAR x : CHAR);
313:D    -4  2   VAR count : INTEGER;
314:C        2   BEGIN
315:C        2     { this can escape with an eod escape }
316:C        2     count := 1;
317:C        2     enter_data( temp , ADDR(x) , count );
318:C        2   END; { of idc_rdb }
319:S
320:S
321:S
322:D        1   PROCEDURE idc_wtb   ( temp : ANYPTR ;  val   : CHAR);
323:C        2   BEGIN
324:C        2     { this can escape with an eod escape }
325:C        2     output_data( temp , ADDR(val) , 1 );
326:C        2   END; { of idc_wtb }
327:S
328:S
329:S
330:D        1   PROCEDURE idc_rdw   ( temp : ANYPTR ;  VAR x : io_word);
331:D    -4  2   VAR count : INTEGER;
332:C        2   BEGIN
333:C        2     { this can escape with an eod escape }
334:C        2     count := 2;
```



```
335:C        2     enter_data( temp , ADDR(x) , count );
336:C        2   END; { of idc_rdw }
337:S
338:S
339:S
340:D        1   PROCEDURE idc_wtw   ( temp : ANYPTR ;  val   : io_word);
341:C        2   BEGIN
342:C        2     { this can escape with an eod escape }
343:C        2     output_data( temp , ADDR(val) , 2 );
344:C        2   END; { of idc_wtw }
345:S
346:S
347:S
348:D        1   PROCEDURE idc_rds   ( temp : ANYPTR ;  reg   : io_word;
349:D        2                                          VAR x : io_word);
350:C        2   BEGIN
351:C        2     direct_status ( temp , reg , x );
352:C        2   END; { of idc_rds }
353:S
354:S
355:S
356:D        1   PROCEDURE idc_wtc   ( temp : ANYPTR ;  reg   : io_word;
357:D        2                                          val   : io_word );
358:D    -2  2   VAR dummyword     : io_word;
359:C        2   BEGIN
360:S              { range of valid registers -
361:S
362:S                 000-127   buffered control
363:S                 257-383   direct   control
364:S                 512       abrt tfr in
365:S                 513       abrt tfr out
366:S
367:C        2     }
368:S
369:C        2     IF ( ( reg >= 257 ) AND ( reg <= 383 ) ) OR
370:C        3          ( reg =  512 ) OR  ( reg =  513 )            { bug 1283 - TM:1/8/82}
371:S
372:C        3       THEN BEGIN
373:S
374:C        3         direct_control ( temp , reg-256 , val );
375:S
376:C        3       END
377:C        3       ELSE BEGIN
378:S
379:C        3         IF ( ( reg >= 000 ) AND ( reg <= 127 ) )     { bug 1283 - TM:1/8/82}
380:C        4           THEN BEGIN
381:S
382:C        4             control_bfd ( temp , reg , val );
383:S
384:C        4             IF ( reg = 0 )
385:C        5               THEN BEGIN
386:S
387:C        5                 { make sure card is still async }
388:S
389:C        5                 direct_status( temp , 3 , dummyword );
390:C        5                 WITH isc_table[ io_find_isc(temp) ] DO
391:C        6                 IF dummyword = 1 THEN card_id := hp98628_async
392:C        7                                  ELSE card_id := hp_datacomm;
393:C        5               END; { of IF reg=0 }
394:S
```

```
395:C        4           END
396:C        4           ELSE BEGIN
397:S
398:C        4             io_escape(ioe_misc,io_find_isc(temp));
399:S
400:C        4           END; { of IF reg<=127 }
401:S
402:C        3       END; { of IF >=257 }
403:S
404:C        2   END; { of idc_wtc }
405:S
406:S
407:S
408:D        1   PROCEDURE idc_tfr   ( temp : ANYPTR ;  bcb   : ANYPTR );
409:D    -4  2   VAR b_info : ^buf_info_type;
410:D    -6  2       io_isc : type_isc ;
411:D   -10  2       tmpptr : pio_tmp_ptr;
412:D   -14  2       tmpcnt : INTEGER ;
413:D   -18  2       mycount: INTEGER ;
414:D   -19  2       done   : BOOLEAN ;
415:C        2   BEGIN
416:C        2     b_info := ANYPTR( bcb );
417:C        2     tmpptr := ANYPTR( temp );
418:C        2     io_isc := tmpptr^.my_isc ;
419:C        2     WITH b_info^ DO BEGIN
420:C        3
421:C        3       IF ( usr_tfr  = serial_DMA )  OR
422:C        4          ( usr_tfr  = overlap_DMA ) OR
423:C        4          ( b_w_mode = TRUE )            { word }
424:C        4         THEN BEGIN
425:C        4           { error }
426:C        4           IF direction = from_memory
427:C        5             THEN tmpptr^.out_bufptr := NIL
428:C        5             ELSE tmpptr^.in_bufptr  := NIL;
429:C        4           io_escape( ioe_bad_tfr , io_isc );
430:C        4         END;
431:S
432:C        3       { mark buffer busy }
433:S
434:C        3       active_isc   := io_isc ;
435:S
436:S
437:C        3       IF usr_tfr < overlap_INTR
438:C        4         THEN BEGIN
439:C        4           { serial transfer - handled here }
440:C        4           IF direction = from_memory
441:C        5             THEN BEGIN
442:C        5
443:C        5               { serial output tfr }
444:S
445:C        5               output_data ( temp , buf_empty , term_count );
446:S
447:C        5               buf_empty := ANYPTR( INTEGER(buf_empty) + term_count);
448:C        5               term_count:= 0;
449:S
450:C        5               IF end_mode THEN output_end( temp );
451:S
452:C        5             END
453:C        5             ELSE BEGIN
454:C        5
```



```
455:C        5               { serial input tfr }
456:C        5
457:C        5               done := false;
458:C        5               mycount := term_count;   { tfr size }
459:C        5               REPEAT
460:C        6                 IF term_char = -1
461:C        7                   THEN BEGIN
462:C        7                     { serial input - no term }
463:C        7                     TRY
464:C        8                       enter_data( temp , buf_fill , mycount );
465:C        8                       term_count := term_count - mycount;
466:C        8                       buf_fill   := ANYPTR( INTEGER(buf_fill) + mycount);
467:C        8
468:C        8                       done := TRUE;
469:S
470:C        8                     RECOVER BEGIN
471:C        8                       IF ( escapecode = ioescapecode ) AND
472:C        9                          ( ioe_result = ioe_eod_seen ) AND
473:C        9                          ( ioe_isc    = io_isc )
474:C        9                         THEN BEGIN
475:C        9                           term_count := term_count - mycount;
476:C        9                           buf_fill   := ANYPTR( INTEGER(buf_fill) + mycount);
477:C        9                           mycount    := term_count;
478:C        9                           IF end_mode
479:C       10                             THEN BEGIN
480:C       10                               done := TRUE;
481:C       10                             END; { of IF end_mode }
482:C        9                         END
483:C        9                         ELSE BEGIN
484:C        9                           escape(escapecode);
485:C        9                         END; { of IF }
486:C        8                     END; { of TRY RECOVER BEGIN }
487:C        7                   END
488:C        7                   ELSE BEGIN
489:C        7                     { serial input - termination }
490:C        7                     TRY
491:C        8
492:C        8                       REPEAT
493:C        9                         tmpcnt :=1 ;
494:C        9                         enter_data( temp , buf_fill , tmpcnt );
495:C        9                         buf_fill   := ANYPTR( INTEGER(buf_fill) + 1 );
496:C        9                         term_count := term_count -1;
497:C        9                       UNTIL ( term_count = 0 ) OR
498:C        9                             ( CHARPTR(INTEGER(buf_fill)-1)^=CHR(term_char)) ;
499:C        8
500:C        8                       done := TRUE;
501:S
502:C        8                     RECOVER BEGIN
503:C        8                       IF ( escapecode = ioescapecode ) AND
504:C        9                          ( ioe_result = ioe_eod_seen ) AND
505:C        9                          ( ioe_isc    = io_isc )
506:C        9                         THEN BEGIN
507:C        9
508:S
509:C        9                           { I use tmpcnt because the eod may or may not have
                                           been caught with the character }
510:S
511:C        9                           term_count := term_count - tmpcnt;
512:C        9                           buf_fill   := ANYPTR( INTEGER(buf_fill) + tmpcnt );
513:C        9                           IF end_mode
514:C       10                             THEN BEGIN
```

```
515:C       10                              done := TRUE;
516:C       10                            END; { of IF end_mode }
517:C        9                         END
518:C        9                         ELSE BEGIN
519:C        9                           escape(escapecode);
520:C        9                         END; { of IF }
521:C        8                       END; { of TRY RECOVER BEGIN }
522:C        7                     END; { of IF term_char }
523:C        6                 UNTIL done;
524:S
525:C        5               END; { of IF direction }
526:S
527:S
528:C        4             { mark buffer not busy }
529:S
530:C        4             IF direction = from_memory
531:C        5               THEN tmpptr^.out_bufptr := NIL
532:C        5               ELSE tmpptr^.in_bufptr  := NIL;
533:C        4             active_isc  := no_isc;
534:C        4             drv_tmp_ptr := NIL;
535:S
536:C        4             IF eot_proc.dummy_pr <> NIL                    {JPC 02/22/82}
537:C        5               THEN BEGIN
538:C        5                  { call user eot procedure }
539:C        5                  CALL ( eot_proc.real_proc, eot_parm );    {JPC 02/22/82}
540:C        5               END; { of IF }
541:S
542:S
543:C        4           END
544:C        4           ELSE BEGIN
545:C        4
546:C        4             { overlap transfer - handled by drivers }
547:C        4
548:C        4             IF direction = to_memory
549:C        5               THEN BEGIN
550:S
551:C        5                  start_tfr_in ( temp );
552:S
553:C        5               END
554:C        5               ELSE BEGIN
555:S
556:C        5                  start_tfr_out ( temp );
557:S
558:C        5               END; { of IF direction }
559:S
560:C        4           END; { of IF overlap }
561:C        3       END; { of WITH b_info }
562:C        2     END; { of idc_tfr }
563:S
564:S
565:C        1 END; { of intdc }
```



```
566:D        1 $PAGE$
567:S
568:S
569:D        1 MODULE init_dc;
570:S                {
571:S                  date    10/20/81
572:S                  update  08/11/83  vers number only
573:S
574:S                  purpose This module initializes the data comm drivers.
575:S
576:D        1         }
577:S
578:S
579:D        1 IMPORT    iodeclarations ;
580:S
581:D        1 EXPORT
582:S
583:S
584:D        1   VAR
585:D  -120  1     dc_drivers    : drv_table_type;
586:S
587:D  -120  1   PROCEDURE io_init_dc;
588:S
589:S
590:S
591:D  -120  1 IMPLEMENT
592:S
593:S
594:D  -120  1   IMPORT  sysglobals ,
595:D  -120  1           isr ,
596:D  -120  1           general_0 ,
597:D  -120  1           extdc ,
598:D  -120  1           intdc ;
599:S
600:D        1   PROCEDURE io_init_dc;
601:D        2   VAR
602:D    -2  2     io_isc        : type_isc;
603:D    -4  2     dummyword     : io_word;
604:D    -6  2     io_lvl        : io_byte;
605:S
606:C        2   BEGIN
607:S
608:C        2     io_revid := io_revid + ' S3.0';     { SERIAL revision added 2/5/82 TM }
609:S
610:C        2     { set up the driver tables }
611:S
612:C        2     WITH dc_drivers DO BEGIN
613:C        3       dc_drivers := dummy_drivers ;
614:C        3       iod_init   := idc_init;
615:C        3       iod_isr    := idc_isr;
616:C        3       iod_rdb    := idc_rdb;
617:C        3       iod_wtb    := idc_wtb;
618:C        3       iod_rdw    := idc_rdw;
619:C        3       iod_wtw    := idc_wtw;
620:C        3       iod_rds    := idc_rds;
621:C        3       iod_wtc    := idc_wtc;
622:C        3       iod_tfr    := idc_tfr;
623:C        3     END; { of WITH }
624:S
625:S
```

```
626:C   2     { set up drivers for the interfaces }
627:S
628:C   2     FOR io_isc:=iominisc TO iomaxisc DO
629:C   3       WITH isc_table[io_isc] DO BEGIN
630:S
631:C   4         IF ( card_id = hp98628_async ) OR
632:C   5            ( card_id = hp98628_dsndl ) OR
633:C   5            ( card_id = hp98629 )
634:C   5           THEN BEGIN
635:C   5             card_id   := hp_datacomm;
636:C   5             card_type := serial_card;                    { uuuu TM 9/28/82 }
637:C   5           END;
638:S
639:C   4         IF card_id = hp_datacomm
640:C   5           THEN BEGIN
641:S
642:C   5             io_drv_ptr:=ADDR(dc_drivers);
643:S
644:C   5             { if the card exists then link in an ISR for it }
645:C   5             { ??? - what happens if an ISR fires during init }
646:S
647:C   5             io_lvl:=((ioread_byte(io_isc,3) DIV 16) MOD 4)+3;
648:C   5             IF io_tmp_ptr^.myisrib.INTREGADDR <> NIL
649:C   6               THEN BEGIN
650:C   6                 { if isr exists then unlink it }
651:C   6                 ISRUNLINK(io_lvl ,                       { level         }
652:C   6                     ADDR(io_tmp_ptr^.myisrib));          { isrib info    }
653:C   6               END; { of IF }
654:S
655:C   5             PERMISRLINK(io_drv_ptr^.iod_isr,             { isr           }
656:C   5                   ANYPTR(INTEGER(card_ptr)+3),           { card address  }
657:C   5                   192,                                   { intr. mask    }
658:C   5                   192,                                   { intr. value   }
659:C   5                   io_lvl,                                { level         }
660:C   5                   ADDR(io_tmp_ptr^.myisrib));            { isrib info    }
661:C   5           END; { of IF card_type = hp_datacomm }
662:S
663:C   4     END; { of FOR io_isc WITH isc_table[io_isc] BEGIN }
664:S
665:S
666:C   2     { call the actual driver initialization }
667:S
668:C   2     dc_init_fault := FALSE;          { set fault flag            zzzz TM 6/16/82 }
669:C   2     FOR io_isc:=iominisc TO iomaxisc DO
670:C   3       WITH isc_table[io_isc] DO
671:C   4       IF card_id = hp_datacomm
672:C   5         THEN BEGIN
673:S
674:C   5           TRY                                                { zzzz TM 6/16/82 }
675:S
676:C   6             alvinit( io_tmp_ptr );
677:S
678:C   6             direct_status( io_tmp_ptr , 3 , dummyword );
679:C   6             IF dummyword = 1 THEN card_id := hp98628_async;
680:C   6             IF dummyword = 2 THEN card_id := hp98628_dsndl;  { cccc TM 8/16/82 }
681:S
682:C   6             { set up Ganglia card ID }                       { bbbb TM 7/16/82 }
683:C   6             IF ioread_byte(io_isc,HEX('402F'))=3             { bbbb TM 7/16/82 }
684:C   7               THEN BEGIN                                     { bbbb TM 7/16/82 }
685:C   7                 { card is ganglia }                          { bbbb TM 7/16/82 }
```



```
686:C   7                 card_id   := hp98629;                      { cccc TM 8/16/82 }
687:C   7                 card_type := srm_card;                     { uuuu TM 9/28/82 }
688:C   7               {iowrite_byte(io_isc ,                       { bbbb TM 7/16/82 }
689:C   7                 {HEX('4061')   , io_model_number DIV 256 );  { bbbb TM 7/16/82 }
690:C   7               {iowrite_byte(io_isc ,                       { bbbb TM 7/16/82 }
691:C   7                 {HEX('4061')   + 2 , io_model_number MOD 256 ); { bbbb TM 7/16/82 }
692:C   7               END; { of IF }                               { bbbb TM 7/16/82 }
693:S
694:C   6           RECOVER BEGIN                                    { zzzz TM 6/16/82 }
695:S
696:C   6             IF ESCAPECODE<>ioescapecode THEN ESCAPE(ESCAPECODE);{ zzzz TM 6/16/82 }
697:C   6             IF ioe_isc <> io_isc        THEN ESCAPE(ESCAPECODE);{ zzzz TM 6/16/82 }
698:C   6
699:C   6             IF NOT dc_init_fault                         { zzzz TM 6/16/82 }
700:C   7               THEN BEGIN                                 { zzzz TM 6/16/82 }
701:C   7                 dc_init_fault := TRUE ;                  { zzzz TM 6/16/82 }
702:C   7                 NEW(dc_error);                 { get space   } { zzzz TM 6/16/82 }
703:C   7                 FOR dummyword:=minrealisc TO maxrealisc DO   { zzzz TM 6/16/82 }
704:C   8                   dc_error^[dummyword]:=0;     { clear space } { zzzz TM 6/16/82 }
705:C   7               END; { of IF NOT dc_init_fault }             { zzzz TM 6/16/82 }
706:S
707:C   6             dc_error^[io_isc] := ioe_result;    { set this err} { zzzz TM 6/16/82 }
708:S
709:C   6           END; { of RECOVER BEGIN }                        { zzzz TM 6/16/82 }
710:S
711:C   5         END; { of WITH IF }
712:S
713:C   2   END; { of io_init_dc }
714:S
715:C   1 END; { of MODULE init_dc }
```

```
   716:D        1 $PAGE$
   717:S
   718:D        ; IMPORT    init_dc ,
   719:D        1           LOADER ;                                    { 367  TM 9/22/82 }
   720:S
   721:S
   722:C        1 BEGIN
   723:C        1   io_init_dc;
   724:C        1   MARKUSER;                                           { 367  TM 9/22/82 }
   725:C        1 END. { of dc_initialize }

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

DGL_C_IN

Description

DGL_C_IN provides selection of different input device handlers.

Requirements

GLE_TYPES, GLE_HPGL_IN, GLE_UTLS, GLE_HPIB_IO, GENERAL_O, IODECLARATIONS, SYSGLOBALS, IOCOMASM, GLE_KNOB_IN, DGL_TOOLS, DGL_VARS, DGL_KNOB, DGL_TYPES, GLE_GEN, GLE_GENI, DGL_GEN, and DGL_CONFG_OUT.

Notes

See also LIB.

```
     1:D        0 {                                                                    }
     2:D        0 { Graphics Low End                                                   }
     3:D        0 {                                                                    }
     4:D        0 { Module    = GLE_CONFG_IN                                           }
     5:D        0 { Programer = BJS                                                    }
     6:D        0 { Date      = 10-10-82                                               }
     7:D        0 {                                                                    }
     8:D        0 { Purpose: To provide selection of differnt input device handlers.   }
     9:S
    10:D        0 { Rev history                                                        }
    11:D        0 {  Created  - 10-10-82                                               }
    12:D        0 {  Modified - 02-17-84 BDS - Changed from dynamic allocations to global }
    13:S
    14:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    15:S          All rights are reserved.  Copying or other
    16:S          reproduction of this program except for archival
    17:S          purposes is prohibited without the prior
    18:S          written consent of Hewlett-Packard Company.
    19:S
    20:S
    21:S                        RESTRICTED RIGHTS LEGEND
    22:S
    23:S          Use, duplication, or disclosure by the Government
    24:S          is subject to restrictions as set forth in
    25:S          paragraph (b) (3) (B) of the Rights in Technical
    26:S          Data and Computer Software clause in
    27:S          DAR 7-104.9(a).
    28:S
    29:S          HEWLETT-PACKARD COMPANY
    30:D        0 Fort Collins, Colorado                                 }
    31:S
    32:S
    33:D        0 $SEARCH 'GLE_LIB',
    34:D        0         'TYPES',
    35:D        0         'DGL_VARS',
    36:D        0         'DGL_TOOLS',
    37:D        0         'DGL_KNOB',
    38:D        0         'DGL_HPGLI'$
    39:D        0 $modcal$
    40:D        0 $include 'OPTIONS'$  { ***************** COMPILER OPTIONS ***************** }
    40:S
    41:S          { This include file specifies range checking, debug and other compiler
    42:D        0   options for the graphics library                                   }
    43:S
    44:D        0 $debug OFF$
    45:D        0 $range OFF$
    46:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    47:D        0 $FLOAT_HDW TEST$
    48:S
    49:S
    50:S
    51:S
    52:D        0 $include 'OPTICNS'$  { ***************** COMPILER OPTIONS ***************** }
 12000:D        0 $LINENUM 12000$
 12001:S
 12002:D        0 module DGL_CONFG_IN;
 12003:S
 12004:D        1 import gle_types;
 12005:S
 12006:D        1 export
```



```
 12007:D        1   procedure configure_input_gle (  gcbi : graphics_input_control_block_ptr );
 12008:S
 12009:D        1 implement
 12010:S
 12011:D        1 import gle_hpgl_in,  { hpgl plotter support }
 12012:D        1        gle_hpib_io,  { plotter HPIB support }
 12013:D        1        gle_knob_in,  { knob support }
 12014:D        1        gle_utils,    { general gle tools }
 12015:D        1        dgl_tools,    { general gle and dgl tools }
 12016:D        1        dgl_vars,     { global dgl variables }
 12017:D        1        sysglobals,   { global pascal information }
 12018:D        1        dgl_knob,     { knob device dependent DGL code }
 12019:D        1        dgl_hpgli;    { plotter device dependent DGL code }
 12020:S
 12021:D        1 var
 12022:D  -264  1        ascii_buffer_space    : ascii_buffer;
 12023:D  -284  1        knob_device_rec_space : knob_device_rec;
 12024:D  -302  1        hpib_iocb_space       : hpib_iocb;
 12025:S
 12026:D        1 procedure termknob ( anyvar iocb_ptr : anyptr );
 12027:S
 12028:D        2 var
 12029:D    -4  2   knob_rec : knob_device_rec_ptr;
 12030:S
 12031:C        2 begin
 12032:C        2   with gle_gcbi^ do
 12033:C        3     begin
 12034:C        3       knob_rec := dev_dep_stuff;
 12035:C        3       {dispose(knob_rec);}
 12036:C        3     end;
 12037:C        2 end;
 12038:S
 12039:D        1 procedure setupknob (  gcbi : graphics_input_control_block_ptr );
 12040:S
 12041:D        2 var
 12042:D    -4  2   knob_rec : knob_device_rec_ptr;
 12043:D    -6  2   cnt      : gle_shortint;
 12044:D   -10  2   address        : integer;
 12045:S
 12046:C        2 begin
 12047:C        2   with gcbi^ do
 12048:C        3     begin
 12049:C        3       address := gle_read_integer(device_info_char_count,device_info,cnt);
 12050:C        3       if (address = 2) and (not sysflag.nokeyboard) then
 12051:C        4         begin
 12052:C        4           io_term := termknob;
 12053:C        4           knob_rec := addr(knob_device_rec_space);
 12054:C        4           dev_dep_stuff := knob_rec;
 12055:C        4           knob_rec^.knob_type := return_machine_type;
 12056:C        4           gle_init_knob_input ( gcbi);
 12057:C        4           if error_return = 0 then
 12058:C        5             dgl_knob_init
 12059:C        5           else
 12060:C        5             release(knob_rec);
 12061:C        4         end
 12062:C        4       else
 12063:C        4         error_return := 1;
 12064:C        3     end;
 12065:C        2 end;
 12066:S
```

```
12067:D        1 procedure termhpgl ( anyvar iocb_ptr : anyptr );
12068:S
12069:D        2 var
12070:D    -4  2   iocb_ptr_hpib   : hpib_iocb_ptr;
12071:D    -8  2   buf             : ascii_buffer_ptr;
12072:S
12073:C        2 begin
12074:C        2   with gle_gcbi^ do
12075:C        3     begin
12076:C        3       hpib_term(iocb_ptr);          { perform io term then release mem }
12077:C        3       iocb_ptr_hpib := iocb;
12078:C        3       {dispose(iocb_ptr_hpib);}
12079:C        3       buf := device_buf;
12080:C        3       {dispose(buf);}
12081:C        3     end;
12082:C        2 end;
12083:S
12084:D        1 procedure setuphpgl (  gcbi : graphics_input_control_block_ptr );
12085:S
12086:D        2 var
12087:D    -4  2   iocb_ptr_hpib   : hpib_iocb_ptr;
12088:D    -8  2   buf             : ascii_buffer_ptr;
12089:D   -10  2   cnt             : gle_shortint;
12090:D   -14  2   address         : integer;
12091:S
12092:C        2 begin
12093:C        2   with gcbi^ do
12094:C        3     begin
12095:C        3       error_return := 1;
12096:C        3       try
12097:C        4         address := gle_read_integer(device_info_char_count,device_info,cnt);
12098:S
12099:C        4         buf := addr(ascii_buffer_space);
12100:C        4         device_buf := buf;
12101:C        4         iocb_ptr_hpib := addr(hpib_iocb_space);
12102:C        4         iocb := iocb_ptr_hpib;
12103:S
12104:C        4         io_write := hpib_write;
12105:C        4         io_read := hpib_read;
12106:C        4         io_term := termhpgl;
12107:C        4         io_inq_timeout := hpib_inq_timeout;
12108:C        4         io_set_timeout := hpib_set_timeout;
12109:C        4         with iocb_ptr_hpib^ do
12110:C        5           begin
12111:C        5             device_addr := device_info;
12112:C        5             name_size := device_info_char_count;
12113:C        5             hpib_init ( iocb_ptr_hpib );
12114:C        5             error_return := 0;
12115:C        5             if error = 0 then gle_init_hpgl_input (gcbi)
12116:C        6             else error_return := 1;
12117:C        5             if error_return = 0 then dgl_hpgli_init
12118:C        6               { otherwise clean up the hpib bus ( 2.1 bug fix ) }
12119:C        6             else                     hpib_init ( iocb_ptr_hpib );
12120:C        5           end;
12121:C        4       recover
12122:C        4         { ignor io, and value range errors }
12123:C        4         if (escapecode <> -26) and (escapecode <> -8) then escape(escapecode);
12124:S                 {if error_return <> 0 then
12125:C        3           begin}                                  { clean up }
12126:S                    {dispose(iocb_ptr_hpib); dispose(buf);
```



```
12127:C        3           end;}
12128:C        3     end;
12129:C        2 end;
12130:S
12131:D        1 procedure configure_input_gle (  gcbi : graphics_input_control_block_ptr );
12132:S
12133:C        2 begin
12134:C        2   setupknob ( gcbi );
12135:C        2   if gcbi^.error_return <> 0 then setuphpgl ( gcbi );
12136:C        2 end;
12137:S
12138:C        1 end. { of module }
12139:S

No errors. No warnings.
          ***** Nonstandard language features enabled *****
```

DGL_C_OUT

Description

DGL_C_OUT provides selection of different output device handlers.

Requirements

GLE_TYPES, SYSDEVS, SYSGLOBALS, GLE_HPGL_OUT, GLE_STEXT, GLE_ASTEXT, GLE_ASCLIP, GLE_SCLIP, GLE_SMARK, GLE_AUTL, GLE_UTLS, GLE_RAS_OUT, GLE_ARAS_OUT, GLE_FILE_IO, GLE_HPIB_IO, GENERAL_O, IODECLARATIONS, IOCOMASM, DGL_TOOLS, DGL_RASTER, DGL_TYPES, DGL_VARS, ASM, LOADER, GLE_GEN, DGL_GEN, and DGL_HPGL.

Notes

See also LIB.

```
Pascal [Rev 3.0M  6/ 4/84] DGL_C_OUT.TEXT          26-Dec-84 22:42:18 Page 1   *** File name: DGL_C_OUT ***

     1:D        0 {                                                                    }
     2:D        0 { Graphics Library                                                   }
     3:D        0 {                                                                    }
     4:D        0 { Module    = DGL_CONFG_OUT                                          }
     5:D        0 { Programer = BJS                                                    }
     6:D        0 { Date      = 10- 5-82                                               }
     7:D        0 {                                                                    }
     8:D        0 { Purpose: To link device dependent drivers with the graphics library.      }
     9:S
    10:D        0 { Rev history                                                        }
    11:D        0 {  Created  - 10- 5-82                                               }
    12:D        0 {  Modified -  1-12-84  BDS -Added Gator black-white support         }
    13:D        0 {  Modified -  2-17-84  BDS -Changed dynamic to global storage for PASC 3.0 }
    14:S
    15:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    16:S                All rights are reserved.  Copying or other
    17:S                reproduction of this program except for archival
    18:S                purposes is prohibited without the prior
    19:S                written consent of Hewlett-Packard Company.
    20:S
    21:S
    22:S                            RESTRICTED RIGHTS LEGEND
    23:S
    24:S                Use, duplication, or disclosure by the Government
    25:S                is subject to restrictions as set forth in
    26:S                paragraph (b) (3) (B) of the Rights in Technical
    27:S                Data and Computer Software clause in
    28:S                DAR 7-104.9(a).
    29:S
    30:S                HEWLETT-PACKARD COMPANY
    31:D        0       Fort Collins, Colorado                                 }
    32:S
    33:D        0 $SEARCH 'GLE_LIB',
    34:D        0         'TYPES',
    35:D        0         'DGL_VARS',
    36:D        0         'DGL_TOOLS',
    37:D        0         'DGL_RAS',
    38:D        0         'DGL_HPGL'$
    39:D        0 $modcal$
    39:D        0 $include 'OPTIONS'$              { compiler options }
    40:S
    41:S           { This include file specifies range checking, debug and other compiler
    42:D        0    options for the graphics library                                 }
    43:S
    44:D        0 $debug OFF$
    45:D        0 $range OFF$
    46:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    47:D        0 $FLOAT_HDW TEST$
    48:S
    49:S
    50:S
    51:S
    52:D        0 $include 'OPTIONS'$              { compiler options }
 11000:D        0 $LINENUM 11000$
 11001:S
 11002:D        0 module DGL_CONFG_OUT;
 11003:S
 11004:D        1 import gle_types, sysdevs;
 11005:S
 11006:D        1 export
```

```
Pascal [Rev 3.0M  6/ 4/84] DGL_C_OUT.TEXT          26-Dec-84 22:42:18 Page 2   *** File name: DGL_C_OUT ***

 11007:D        1   procedure configure_gle (  gcb : graphics_control_block_ptr );
 11008:S
 11009:D        1 implement
 11010:S
 11011:D        1 import gle_hpgl_out,   { hpgl plotter support }
 11012:D        1        gle_ras_out,    { raster support       }
 11013:D        1        gle_file_io,    { plotter spooling io  }
 11014:D        1        gle_hpib_io,    { plotter HPIB support }
 11015:D        1        gle_utils,      { general tools }
 11016:D        1        dgl_tools,      { used to get machine type }
 11017:D        1        sysglobals,     { address for GRAPHICSBASE }
 11018:D        1        iodeclarations, { used to get min, max selectcode ranges }
 11019:D        1        dgl_raster,     { DGL device dependent raster init code }
 11020:D        1        dgl_hpgl,       { DGL device dependent HPGL init code }
 11021:D        1        dgl_vars;       { DGL global data }
 11022:D        1 var
 11023:D    -8  1   save_crthook: crtlltype;
 11024:D   -12  1   hp98627A_address : anyptr;  { holds adr of first graphics plane }
 11025:D   -13  1   found_gator: boolean;
 11026:D   -16  1   select_code : shortint;
 11027:D   -17  1   has_color : boolean;
 11028:D   -22  1   frame_buffer : integer;
 11029:D   -26  1   stat : ^shortint;
 11030:D   -30  1   ptr: ^shortint;
 11031:D   -32  1   int_ext_gator : shortint;   { 0=neither, 1=internal, 2=external}
 11032:D  -550  1   raster_device_rec_space : raster_device_rec;
 11033:D  -552  1   hpgl_device_rec_space   : hpgl_device_rec;
 11034:D  -816  1   ascii_buffer_space      : ascii_buffer;
 11035:D -1488  1   file_iocb_space         : file_iocb;
 11036:D -1506  1   hpib_iocb_space         : hpib_iocb;
 11037:D -1507  1   took_type_ahead         : boolean;
 11038:D -1508  1   reduced_screen          : boolean;
 11039:D -1509  1   secondary               : boolean;
 11040:D -1510  1   moon                    : boolean;
 11041:S
 11042:S           {procedure dummy_crthook(dum_op : crtllops; anyvar dum_pos : integer;
 11043:S                                    dum_c : char);
 11044:S           begin
 11045:D -1510  1 end;}
 11046:S
 11047:D        1 procedure termraster ( anyvar iocb_ptr : anyptr );
 11048:S
 11049:D        2 var
 11050:D    -4  2   ptr : raster_device_rec_ptr;
 11051:D    -5  2   charvar : char;
 11052:S
 11053:C        2 begin
 11054:C        2   with gle_gcb^ do
 11055:C        3     begin
 11056:C        3       ptr := dev_dep_stuff;
 11057:C        3       if reduced_screen then
 11058:C        4         begin
 11059:C        4           with gcb^,raster_device_rec_ptr(dev_dep_stuff)^ do
 11060:C        5             begin
 11061:C        5               reduced_screen := true;
 11062:C        5               n_glines := 752;
 11063:C        5               hard_ymax := 751;
 11064:C        5             end;
 11065:C        4         end;
 11066:C        3       if took_type_ahead then
```

```
11067:C        4         begin
11068:C        4           crtllhook := save_crthook;
11069:C        4           keybufops(kdisplay,charvar);
11070:C        4         end;
11071:C        3       {dispose(ptr);}
11072:C        3     end;
11073:C        2 end;
11074:S
11075:S
11076:S          {Look for gator present.  Assume only one gator is on the bus and that
11077:D -1510  1  an internal one overrides an external one if multiples present.}
11078:S
11079:D        1 procedure gatorcrttype(var found_gator,has_color:boolean;
11080:D        2                        var  frame_buffer :integer;
11081:D        2                        var select_code,int_ext_gator :shortint);
11082:S
11083:D        2 const
11084:D        2     gatorid=25;
11085:D        2 var
11086:D    -2  2     i: shortint;
11087:D    -4  2     dummy: shortint;
11088:S
11089:S
11090:C        2 begin
11091:C        2   control_space:=0;
11092:C        2   found_gator:=false;
11093:C        2   int_ext_gator := 0;
11094:C        2   ptr:=anyptr(hex('560000'));
11095:S
11096:C        2  try
11097:C        3   dummy:=ptr^;
11098:C        3   if (dummy mod 128) = gatorid then          {look for internal gator }
11099:C        4     begin
11100:C        4      found_gator:=true;
11101:C        4      control_space:=integer(ptr);
11102:C        4     end;
11103:C        3  recover
11104:C        3   if escapecode<>-12 then escape(escapecode);
11105:C        2
11106:C        2  if found_gator then                        {if there, find frame buffer}
11107:C        3       begin
11108:C        3         int_ext_gator := 1;
11109:C        3         stat := anyptr(control_space + 16384);
11110:C        3         frame_buffer := ((stat^) mod 16)*1048576;
11111:C        3         end
11112:S
11113:C        3  else
11114:C        3   begin
11115:C        3    ptr:=anyptr(hex('680000'));
11116:C        3    i:=8;
11117:C        3    while (i<=31) and not found_gator  do begin  {look for external gator}
11118:C        4      select_code := i;
11119:C        4      try
11120:C        5        dummy:=ptr^;
11121:C        5        if (dummy mod 128) = gatorid then begin
11122:C        6          found_gator:=true;
11123:C        6          control_space:=integer(ptr);
11124:C        6        end;
11125:C        5      recover
11126:C        5        if escapecode<>-12 then escape(escapecode);
```



```
11127:S
11128:C        4      ptr:=anyptr(integer(ptr)+65536);
11129:C        4      i:=i+1;
11130:C        4    end;
11131:C        3    if found_gator then
11132:C        4     begin
11133:C        4       stat := anyptr(control_space + 16384);
11134:C        4       frame_buffer := ((stat^) mod 16)*1048576;
11135:C        4       int_ext_gator := 2;
11136:C        4     end;
11137:C        3   end;
11138:S
11139:S
11140:S          {if found_gator then begin
11141:S            try
11142:S              has_color:=false;                      assume no color
11143:S              ptr:=anyptr(control_space+hex('410E'));  read colormap id
11144:S              dummy:=ptr^;
11145:S              has_color:=true;                       set true if color there
11146:S            recover
11147:S              if escapecode<>-12 then escape(escapecode);
11148:C        2 end;}
11149:S
11150:C        2 end;
11151:S
11152:S
11153:D        1 procedure setupraster (  gcb : graphics_control_block_ptr );
11154:S
11155:D        2 var
11156:D        2   graphics_base ['GRAPHICSBASE'] : anyptr;
11157:D    -4  2   device_work_area : raster_device_rec_ptr;
11158:D    -6  2   cnt              : gle_shortint;
11159:D   -10  2   address          : integer;
11160:D   -14  2   control          : integer;
11161:D   -15  2   knob_echo_gcb    : boolean;
11162:D   -20  2   g_ptr            : ^shortint;
11163:D   -22  2   g_dummy          : shortint;
11164:D   -23  2   graphics_bd      : boolean;
11165:D   -23  2   graphicsstate ['GRAPHICSFLAG'] : boolean;
11166:S
11167:D        2   procedure dummy1 ( anyvar iocb_ptr, data_ptr : anyptr );
11168:C        3   begin
11169:C        3   end;
11170:S
11171:D        2   procedure expand_screen;
11172:S
11173:C        3   begin
11174:C        3     with gcb^ do
11175:C        4       begin
11176:C        4         info3 := 0;
11177:C        4         if (int_ext_gator = 1) or (irt_ext_gator = 2) then
11178:C        5          begin
11179:C        5            reduced_screen := false;
11180:C        5            info3 := 1;                  {1=expand; 0=leave reduced}
11181:C        5          end;                           {send on to expand screen}
11182:C        4       end;
11183:C        3     if (currentcrt = bitmaptype) and (int_ext_gator =1) then
11184:C        4       begin
11185:C        4         save_crthook := crtllhook;
11186:C        4         crtllhook := dummycrtll;
```

```
11187:C  4        {gator_clear (gcb);}
11188:C  4        took_type_ahead := true;
11189:C  4      end;
11190:C  3  end;
11191:S
11192:D  2   procedure ck_for_graphics_board;
11193:S
11194:C  3   begin
11195:C  3     graphics_bd := true;
11196:C  3     if graphicstate then g_ptr := anyptr(hex('530000'))
11197:C  4      else g_ptr := anyptr(hex('538000'));
11198:C  3     try
11199:C  4       g_dummy := g_ptr^;
11200:C  4     recover
11201:C  4       begin
11202:C  4         if escapecode <> -12 then escape(escapecode)
11203:C  5         else  graphics_bd := false;
11204:C  4       end;
11205:C  3   end;
11206:S
11207:S
11208:S
11209:S
11210:D  2   procedure setup_internal;
11211:S
11212:D  3     procedure toggle_graphics;
11213:D  4     var gon  [5439488  {530000 HEX}] : shortint;
11214:D  4         goff [5472256  {538000 HEX}] : shortint;
11215:D  4         g_on36c [ hex('51FFFC')]: shortint;
11216:D  4         gbase['GRAPHICSBASE'] : ^shortint;
11217:C  4     begin
11218:C  4       if gcb^.info1 = m9836c then begin
11219:C  5         if graphicstate then g_on36c:=1
11220:C  6         else g_on36c:=0;
11221:C  5         gbase:=anyptr(hex('520000'));
11222:C  5       end
11223:C  5       else begin
11224:C  5         if graphicstate then gbase := addr(gon)
11225:C  6                         else gbase := addr(goff);
11226:C  5         gbase^ := gbase^;
11227:C  5       end;
11228:C  4     end;
11229:S
11230:C  3   begin
11231:C  3     with gcb^ do
11232:C  4      begin
11233:C  4        graphicstate := true;
11234:C  4        info1 := return_machine_type;
11235:C  4        toggle_graphics;
11236:C  4        info_ptr1 := addr(graphics_base);
11237:C  4        info_ptr2 := anyptr(0);
11238:C  4        if info1 = m9836c then
11239:C  5          begin
11240:C  5            info2 := hex('51fffd');
11241:C  5            info3 := hex('51fb00');
11242:C  5          end;
11243:C  4      end;
11244:C  3   end;
11245:S
11246:S
```



```
11247:D  2   procedure set_gator_vals;
11248:S
11249:C  3   begin
11250:C  3     with gcb^ do
11251:C  4       begin
11252:S            {if has_color then
11253:C  4            info1 := mgator_c)
11254:C  4           (else)
11255:C  4         info1 := m9837a;                    {display type}
11256:C  4         info2 := control_space;             {top of control space}
11257:C  4         info3 := 0;                         {By default dont expand ! BJS 5-29-84)
11258:C  4         info_ptr1 := addr(frame_buffer); {start of control space}
11259:C  4         info_ptr2 := anyptr(0);
11260:C  4       end;
11261:C  3   end;
11262:S
11263:S
11264:C  2 begin
11265:C  2   with gcb^ do
11266:C  3     if spooling = 0 then
11267:C  4       try
11268:C  5         gatorcrttype(found_gator, has_color,
11269:C  5                      frame_buffer, select_code,int_ext_gator);
11270:C  5         ck_for_graphics_board;
11271:C  5         secondary := false;
11272:C  5         moon := false;
11273:C  5         reduced_screen := true;
11274:C  5         took_type_ahead := false;
11275:C  5         control := info1;  { control passed in info1 }
11276:C  5         knob_echo_gcb := info2 = 1; { GCB for knob echos }
11277:C  5         io_write := dummy1;
11278:C  5         io_term  := termraster;
11279:C  5         address := gle_read_integer(device_info_char_count,device_info,cnt);
11280:C  5         device_work_area := addr(raster_device_rec_space);
11281:C  5         dev_dep_stuff := device_work_area;
11282:C  5
11283:S
11284:C  5         if address = 3 then                        {indicates primary}
11285:C  6           begin                                    {display}
11286:C  6             if ((currentcrt = alphatype)  or (currentcrt = nocrt))
11287:C  7                  and (graphics_bd) then setup_internal
11288:C  7              else if ((currentcrt = bitmaptype) and (int_ext_gator = 1))
11289:C  8                       or ((not graphics_bd) and (int_ext_gator = 1)) then
11290:C  8                  set_gator_vals;
11291:C  6           end
11292:S
11293:C  6         else if (address = 6) then                 {indicates secondary }
11294:C  7           begin                                    {display}
11295:C  7             secondary := true;
11296:C  7             if ((currentcrt = alphatype)  or (currentcrt = nocrt)
11297:C  8                  and (int_ext_gator <>0)) then
11298:C  8                set_gator_vals
11299:C  8                else
11300:C  8                     if (graphics_bd) then setup_internal;
11301:S
11302:C  7             if (currentcrt = bitmaptype)  then           {console = gator so}
11303:C  8               begin                                      {set secondary to  }
11304:C  8                                                          {small screen}
11305:C  8                    if (graphics_bd) then setup_internal
11306:C  9                    else                                  {if fails set second.}
```

```
11307:C        9                       if (int_ext_gator =1) then          {to gator.}
11308:C       10                         set_gator_vals;
11309:C        8                   end;
11310:C        7                 end
11311:C        7               else       { must be moonunit or gator }
11312:C        7                 begin
11313:C        7                   if (address < minrealisc) or (address > maxrealisc) then escape(1);
11314:C        7                   if (address = select_code) and (int_ext_gator = 2) then
11315:C        8                     set_gator_vals
11316:C        8                   else
11317:C        8                    begin
11318:C        8                      info3 := control div 256;  { get monitor type information (part of control) }
11319:C        8                      if (info3 > 6) or (info3 < 1) then info3 := 1;
11320:C        8                      moon := true;
11321:C        8                      info1 := m98627a;          { set display type to 98627A }
11322:C        8                      info2 := address * 65536 + 6291456;  { i/o card address }
11323:C        8                      hp98627a_address := anyptr(info2 + hex('8000')); { first plane adr }
11324:C        8                      info_ptr1 := addr(hp98627a_address);
11325:C        8                      info_ptr2 := anyptr(0);
11326:C        8                    end;
11327:C        7                 end;
11328:S
11329:C        5               {control set}
11330:C        5               if (odd(control DIV 256)) and (not moon)  then
11331:C        6                if
11332:C        7                  ((currentcrt = bitmaptype) and (int_ext_gator = 1) and  {is bitmap, primary,  }
11333:C        7                    (not secondary))                                      { gator there         }
11334:C        7                                    or                                    {          or         }
11335:C        7                  (((currentcrt = alphatype) or (currentcrt = nocrt)) and {is alpha/none,gr bd, }
11336:C        7                   (graphics_bd) and (secondary) and (int_ext_gator = 1)) {second,gator there   }
11337:C        7                                    or                                    {          or         }
11338:C        7                  (address > 6) and (int_ext_gator = 2)                   {ext. gator there     }
11339:C        7                then expand_screen;
11340:S
11341:C        5               {control not set, but gator is not console}
11342:C        5               if (((currentcrt = alphatype) or (currentcrt = nocrt)) and
11343:C        6                   (int_ext_gator <>0) and
11344:C        6                   (secondary))
11345:C        6                                or
11346:C        6                  (((currentcrt = alphatype) or (currentcrt = nocrt)) and
11347:C        6                   (not graphics_bd) and (int_ext_gator <>0))
11348:C        6               then expand_screen;
11349:S
11350:S
11351:C        5               gle_init_raster_output (gcb);
11352:S
11353:S
11354:C        5               if (error_return = 0) and (not knob_echo_gcb) then dgl_raster_init(control);
11355:S
11356:S                         {if error_return <> 0 then
11357:C        5                   dispose(device_work_area);} { clean up }
11358:C        5             recover
11359:S                         { ignore all escapes (except stop key), user may look at
11360:C        5                 escapecode to determine error }
11361:C        5               if escapecode = -20 then escape(-20)
11362:C        6               else error_return := 1
11363:C        6           else
11364:C        4             error_return := 1; { raster devices may not be spooled }
11365:C        2 end;
11366:S
```



```
11367:D        1 procedure termhpgl ( anyvar iocb_ptr : anyptr );
11368:S
11369:D        2 var
11370:D    -4  2   iocb_ptr_file   : file_iocb_ptr;
11371:D    -8  2   iocb_ptr_hpib   : hpib_iocb_ptr;
11372:D   -12  2   buf             : ascii_buffer_ptr;
11373:D   -16  2   device_work_area : hpgl_device_rec_ptr;
11374:D   -20  2   save_ioresult   : integer;            { | fix clobbering ioresult -- 12/83}
11375:S
11376:C        2 begin
11377:C        2   with gle_gcb^ do
11378:C        3     begin
11379:C        3       if spooling <> 0 then
11380:C        4         begin
11381:C        4           save_ioresult := ioresult;   { | ioresult problem fix 12/83--BDS}
11382:C        4           file_term(iocb_ptr);         { perform io term then release mem }
11383:C        4           ioresult := save_ioresult;   { | ioresult problem fix 12/83--BDS}
11384:C        4           iocb_ptr_file := iocb;
11385:C        4           {dispose(iocb_ptr_file);}
11386:C        4         end
11387:C        4       else
11388:C        4         begin
11389:C        4           hpib_term(iocb_ptr);         { perform io term then release mem }
11390:C        4           iocb_ptr_hpib := iocb;
11391:C        4           {dispose(iocb_ptr_hpib);}
11392:C        4         end;
11393:C        3       buf := device_buf;
11394:C        3       device_work_area := dev_dep_stuff;
11395:C        3       {dispose(buf); dispose(device_work_area);}
11396:C        3     end;
11397:C        2 end;
11398:S
11399:D        1 procedure setuphpgl (  gcb : graphics_control_block_ptr );
11400:S
11401:D        2 var
11402:D    -4  2   iocb_ptr_file    : file_iocb_ptr;
11403:D    -8  2   iocb_ptr_hpib    : hpib_iocb_ptr;
11404:D   -12  2   buf             : ascii_buffer_ptr;
11405:D   -16  2   device_work_area : hpgl_device_rec_ptr;
11406:D   -18  2   cnt             : gle_shortint;
11407:D   -22  2   address         : integer;
11408:D   -23  2   address_found   : boolean;
11409:D   -28  2   control         : integer;
11410:D   -32  2   save_ioresult   : integer;              { | fix clobbering ioresult -- 12/83}
11411:D   -36  2   save            : integer;
11412:S
11413:C        2 begin
11414:C        2   with gcb^ do
11415:C        3     begin
11416:C        3       control := info1; { control passed in info1 }
11417:C        3       address_found := false;
11418:C        3       try
11419:C        4         address := gle_read_integer(device_info_char_count,device_info,cnt);
11420:C        4         address_found := true;
11421:C        4       recover
11422:C        4         if escapecode <> -8 { value range error } then escape(escapecode);
11423:S
11424:C        3       buf := addr(ascii_buffer_space);
11425:C        3       device_buf := buf;
11426:C        3       device_work_area := addr(hpgl_device_rec_space);
```

```
11427:C        3       dev_dep_stuff := device_work_area;
11428:S
11429:C        3       if spooling = 1 then
11430:C        4         begin
11431:C        4           iocb_ptr_file := addr(file_iocb_space);
11432:C        4           iocb := iocb_ptr_file;
11433:C        4           io_write := file_write;
11434:C        4           io_term := termhpgl;
11435:C        4           io_inq_timeout := file_inq_timeout;
11436:C        4           io_set_timeout := file_set_timeout;
11437:C        4           with iocb_ptr_file^ do
11438:C        5             begin
11439:C        5               file_name := device_info;
11440:C        5               name_size := device_info_char_count;
11441:C        5               try
11442:C        6                 lock_on_close := 0;                 { do not save file by default }
11443:C        6                 file_init ( iocb_ptr_file );
11444:C        6                 gle_init_hpgl_output (gcb);
11445:C        6                 if error_return = 0 then
11446:C        7                   begin
11447:C        7                     dgl_hpgl_init(control);
11448:C        7                     lock_on_close := 1;           { save file }
11449:C        7                   end
11450:C        7                 else
11451:C        7                   begin
11452:C        7                     save_ioresult := ioresult;    {| ioresult fix 12/83--BDS}
11453:C        7                     file_term ( iocb_ptr_file );
11454:C        7                     ioresult := save_ioresult;    {| ioresult fix 12/83--BDS}
11455:C        7                   end;
11456:C        6               recover
11457:C        6                 if escapecode <> -10 then escape(escapecode)
11458:C        7                 else                      error_return := 1;
11459:C        5             end;
11460:S
11461:C        4           if error_return <> 0 then
11462:C        5             begin { clean up }
11463:C        5               save_ioresult := ioresult;          {| ioresult fix 1/84--BDS}
11464:C        5              {dispose(iocb_ptr_file); dispose(buf); dispose(device_work_area);}
11465:C        5               ioresult := save_ioresult;          {| ioresult fix 1/84--BDS}
11466:C        5             end;
11467:C        4         end
11468:C        4       else
11469:C        4       if address_found then
11470:C        5         begin
11471:C        5           iocb_ptr_hpib := addr(hpib_iocb_space);
11472:C        5           iocb := iocb_ptr_hpib;
11473:C        5           io_write := hpib_write;
11474:C        5           io_read := hpib_read;
11475:C        5           io_term := termhpgl;
11476:C        5           io_inq_timeout := hpib_inq_timeout;
11477:C        5           io_set_timeout := hpib_set_timeout;
11478:C        5           with iocb_ptr_hpib^ do
11479:C        6             begin
11480:C        6               device_addr := device_info;
11481:C        6               name_size   := device_info_char_count;
11482:C        6             end;
11483:C        5           hpib_init ( iocb_ptr_hpib );
11484:C        5           if iocb_ptr_hpib^.error = 0 then
11485:C        6             begin
11486:C        6               gle_init_hpgl_output (gcb);
```



```
11487:C        6               if error_return = 0 then dgl_hpgl_init(control)
11488:C        7                { if error then clean up hpib bus (2.1 bug fix) }
11489:C        7               else                      hpib_init ( iocb_ptr_hpib );
11490:C        6             end
11491:C        6           else error_return := 1;
11492:C        5           if error_return <> 0 then
11493:C        6             begin { clean up }
11494:C        6              {dispose(iocb_ptr_hpib); dispose(buf); dispose(device_work_area);}
11495:C        6             end;
11496:C        5         end
11497:C        5       else
11498:C        5         error_return := 1;
11499:C        3     end;
11500:C        2 end;
11501:S
11502:D        1 procedure configure_gle (  gcb : graphics_control_block_ptr );
11503:S
11504:D        2 VAR
11505:D    -4  2     save : integer;
11506:S
11507:C        2 begin
11508:C        2   with gcb^ do
11509:C        3     begin
11510:C        3       setupraster ( gcb );
11511:C        3       if error_return <> 0 then setuphpgl ( gcb );
11512:C        3     end;
11513:C        2 end;
11514:S
11515:C        1 end. { of module }
11516:S
11517:S

No errors. No warnings.
          ***** Nonstandard language features enabled *****
```

DGL_HPGL

Description

DGL_HPGL provides device-dependent initialization of HPGL devices, input/output escape functions, and HPGL plotter setup utilities.

Requirements

DGL_TYPES, DGL_TYPES, DGL_VARS, GLE_TYPES, GLE_GEN, GLE_UTLS, SYSGLOBALS, and LOADER.

```
     1 D        0 {                                                                    }
     2:D        0 { DGL device dependent init routine                                  }
     3:D        0 {                                                                    }
     4:D        0 { Module    = DGL_HPGL                                               }
     5 D        0 { Programer = BJS                                                    }
     6:D        0 { Date      = 10- 5-82                                               }
     7:D        0 {                                                                    }
     8:D        0 { Purpose: To provide device dependent initialization for HPGL devices.  }
     9:S
    10:D        0 { Rev history                                                        }
    11 D        0 {  Created  - 1 - 5-82  BJS                                          }
    12:D        0 {  Modified - 4 -03-84  BDS Changes dynamic allocations to globals for 3.0  }
    13 D        0 {                           Added identifiers for 7586,7550,7475     }
    14:S
    15:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    16:S                All rights are reserved.  Copying or other
    17:S                reproduction of this program except for archival
    18:S                purposes is prohibited without the prior
    19:S                written consent of Hewlett-Packard Company.
    20:S
    21:S
    22:S                            RESTRICTED RIGHTS LEGEND
    23:S
    24:S                Use, duplication, or disclosure by the Government
    25:S                is subject to restrictions as set forth in
    26:S                paragraph (b) (3) (B) of the Rights in Technical
    27:S                Data and Computer Software clause in
    28:S                DAR 7-104.9(a).
    29:S
    30:S                HEWLETT-PACKARD COMPANY
    31:D        0       Fort Collins, Colorado                              }
    32:S
    33:D        0 $search 'GLE_LIB',
    34:D        0         'TYPES',
    35:D        0         'DGL_VARS'$
    36:D        0 $modcal$
    36:D        0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS ****************** }
    37:S
    38:S           { This include file specifies range checking, debug and other compiler
    39:D        0    options for the graphics library                                 }
    40:S
    41:D        0 $debug OFF$
    42:D        0 $range OFF$
    43:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    44:D        0 $FLOAT_HDW TEST$
    45:S
    46:S
    47:S
    48:S
    49:D        0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS ****************** }
 17000:D        0 $linenum 17000$
 17001:S
 17002:D        0 module DGL_HPGL;
 17003:S
 17004:D        1 export
 17005:S
 17006:D        1 procedure dgl_hpgl_init(control : integer);
 17007:S
 17008:D        1 implement
 17009:S
```



```
 17010:D        1 import dgl_types,
 17011:D        1        dgl_vars,
 17012:D        1        gle_gen,
 17013:D        1        gle_utls,
 17014:D        1        asm;
 17015:S
 17016:D        1 procedure hpgl_input_esc (       opcode : integer;
 17017:D        2                                  isize  : integer;
 17018:D        2                                  rsize  : integer;
 17019:D        2                           anyvar ilist  : gint_list;
 17020:D        2                           anyvar rlist  : greal_list;
 17021:D        2                           var    ierr   : integer  );
 17022:D        2
 17023:S
 17024:D        2 { Purpose : To perform an input escape function                      }
 17025:S
 17026:D        2 var
 17027:D   -42  2   s : string[40];
 17028:D   -46  2   cnt : integer;
 17029:D   -47  2   c : char;
 17030:S
 17031:C        2 begin
 17032:C        2   with gle_gcb^ do
 17033:C        3     begin
 17034:C        3       if spooling = 1 then ierr := 4;
***WARNING: (line17035): 'ADDR' of a constant may not be supported on other implementations
 17035:C        3       if ((gle_match(4,addr(display_name),4,addr('7580')) or
***WARNING: (line17036): 'ADDR' of a constant may not be supported on other implementations
 17036:C        4           gle_match(4,addr(display_name),4,addr('7550')) or
***WARNING: (line17037): 'ADDR' of a constant may not be supported on other implementations
 17037:C        4           gle_match(4,addr(display_name),4,addr('7475')) or
***WARNING: (line17038): 'ADDR' of a constant may not be supported on other implementations
 17038:C        4           gle_match(4,addr(display_name),4,addr('7090')) or
***WARNING: (line17039): 'ADDR' of a constant may not be supported on other implementations
 17039:C        4           gle_match(4,addr(display_name),4,addr('7585')) or
***WARNING: (line17040): 'ADDR' of a constant may not be supported on other implementations
 17040:C        4           gle_match(4,addr(display_name),4,addr('7586')))) and
 17041:C        4            (opcode = 2050) then
 17042:C        4          begin
 17043:C        4            if (ierr = 0) then
 17044:C        5              begin
***WARNING: (line17045): 'ADDR' of a constant may not be supported on other implementations
 17045:C        5                info_ptr1 := addr('OT');
 17046:C        5                info1     := 2;
 17047:C        5                gle_output_escapeo( gle_gcb );
 17048:C        5                gle_flush_buffer (gle_gcb);
 17049:C        5                info_ptr1 := addr(s[1]);
 17050:C        5                gle_output_escapei ( gle_gcb );
 17051:C        5                setstrlen(s,info1);
 17052:C        5                strread(s,1,cnt,ilist[1],c,ilist[2]);
 17053:C        5              end;
 17054:C        4          end
 17055:C        4        else
 17056:C        4          ierr := 1;
 17057:C        3      end;
 17058:S
 17059:C        2 end; { input_esc }
 17060:S
 17061:D        1 procedure hpgl_output_esc (      opcode : integer;
 17062:D        2                                  isize  : integer;
```

```
17063:D        2                                  rsize  : integer;
17064:D        2                            anyvar ilist  : gint_list;
17065:D        2                            anyvar rlist  : greal_list;
17066:D        2                            var    ierr   : integer  );
17067:S
17068:D        2 { Purpose : To perform an output escape funtion                          }
17069:S
17070:D        2 var
17071:D   -22  2   s : string[20];
17072:D   -26  2   cnt : integer;
17073:S
17074:D        2   procedure set_auto_pen;
17075:S
17076:C        3   begin
17077:C        3     strwrite(s,1,cnt,'AP',ilist[1]:1);
17078:C        3   end;
17079:S
17080:D        2   procedure control_cutter;
17081:S
17082:C        3   begin
17083:C        3     s := 'EC';
17084:C        3     if ilist[1] = 0 then
17085:C        4       begin setstrlen(s,3); s[3] := '0'; end;
17086:C        3   end;
17087:S
17088:D        2   procedure advance_page;
17089:S
17090:C        3   begin
17091:C        3     if ilist[1] = 0 then s := 'AH'
17092:C        4     else s := 'AF';
17093:C        3   end;
17094:S
17095:D        2   procedure set_velocity;
17096:S
17097:C        3   begin
17098:C        3     strwrite(s,1,cnt,'VS',ilist[1]:1);
17099:C        3     if (ilist[2] > 0) and (ilist[2] <= gle_gcb^.gamut) then
17100:C        4       strwrite(s,cnt,cnt,',',ilist[2]:1);
17101:C        3   end;
17102:S
17103:D        2   procedure set_force;
17104:S
17105:C        3   begin
17106:C        3     strwrite(s,1,cnt,'FS',ilist[1]:1);
17107:C        3     if (ilist[2] > 0) and (ilist[2] < gle_gcb^.gamut) then
17108:C        4       strwrite(s,cnt,cnt,',',ilist[2]:1);
17109:C        3   end;
17110:S
17111:D        2   procedure set_acceleration;
17112:S
17113:C        3   begin
17114:C        3     strwrite(s,1,cnt,'AS',ilist[1]:1);
17115:C        3     if (ilist[2] > 0) and (ilist[2] < gle_gcb^.gamut) then
17116:C        4       strwrite(s,cnt,cnt,',',ilist[2]:1);
17117:C        3   end;
17118:S
17119:C        2 begin
17120:C        2   with gle_gcb^ do
17121:C        3     begin
17122:C        3       s := '';
```



```
***WARNING: (line17123): 'ADDR' of a constant may not be supported on other implementations
17123:C        3       if gle_match(4,addr(display_name),4,addr('9872')) then
17124:C        4         begin
17125:C        4           if (opcode = 1052) then
17126:C        5             begin if (ierr = 0) then control_cutter; end
17127:C        5           else
17128:C        5           if (opcode = 1053) then
17129:C        6             begin if (ierr = 0) then advance_page; end
17130:C        6           else
17131:C        6           if (opcode = 2050) then
17132:C        7             begin if (ierr = 0) then set_velocity; end
17133:C        7           else
17134:C        7             ierr := 1;
17135:C        4         end
17136:C        4       else
***WARNING: (line17137): 'ADDR' of a constant may not be supported on other implementations
17137:C        4       if ((gle_match(4,addr(display_name),4,addr('7470'))) or
***WARNING: (line17138): 'ADDR' of a constant may not be supported on other implementations
17138:C        5         (gle_match(4,addr(display_name),4,addr('7440')))) then
17139:C        5       begin
17140:C        5         if (opcode = 2050) then
17141:C        6           begin if (ierr = 0) then set_velocity; end
17142:C        6         else
17143:C        6           ierr := 1;
17144:C        5       end
17145:C        5     else
***WARNING: (line17146): 'ADDR' of a constant may not be supported on other implementations
17146:C        5     if (gle_match(4,addr(display_name),4,addr('7580')) or
***WARNING: (line17147): 'ADDR' of a constant may not be supported on other implementations
17147:C        6       gle_match(4,addr(display_name),4,addr('7550'))  or
***WARNING: (line17148): 'ADDR' of a constant may not be supported on other implementations
17148:C        6       gle_match(4,addr(display_name),4,addr('7475'))  or
***WARNING: (line17149): 'ADDR' of a constant may not be supported on other implementations
17149:C        6       gle_match(4,addr(display_name),4,addr('7090'))  or
***WARNING: (line17150): 'ADDR' of a constant may not be supported on other implementations
17150:C        6       gle_match(4,addr(display_name),4,addr('7585'))  or
***WARNING: (line17151): 'ADDR' of a constant may not be supported on other implementations
17151:C        6       gle_match(4,addr(display_name),4,addr('7586'))) then
17152:C        6     begin
17153:C        6       if (opcode = 1052) then
17154:C        7         begin if (ierr = 0) then set_auto_pen; end
17155:C        7       else
17156:C        7       if ((opcode = 1053) and
***WARNING: (line17157): 'ADDR' of a constant may not be supported on other implementations
17157:C        8           ((gle_match(4,addr(display_name),4,addr('7586'))) or
***WARNING: (line17158): 'ADDR' of a constant may not be supported on other implementations
17158:C        8             (gle_match(4,addr(display_name),4,addr('7550')))))
17159:C        8         then
17160:C        8           begin if (ierr = 0) then advance_page; end
17161:C        8         else
17162:C        8         if (opcode = 2050) then
17163:C        9           begin if (ierr = 0) then set_velocity; end
17164:C        9         else
17165:C        9         if (opcode = 2051) then
17166:C       10           begin if (ierr = 0) then set_force; end
17167:C       10         else
17168:C       10         if (opcode = 2052) then
17169:C       11           begin if (ierr = 0) then set_acceleration; end
17170:C       11         else
17171:C       11           ierr := 1;
```

```
1717:C        6         end
1717:C        6       else
1717:C        6         ierr := 1;
1717:C        3       if ierr = 0 then
1717:C        4         begin
1717:C        4           info_ptr1 := addr(s[1]);
1717:C        4           info1 := strlen(s);
1717:C        4           gle_output_escapeo(gle_gcb);
1718:C        4         end;
1718:C        3     end;
1718:C        2 end; { output_esc }
1718:S
1718:D        1  procedure hpgl_linestyle ( index : integer);
1718:S
1718:D        2 { Purpose:  To set the linestyle that primitives are drawn with             }
1718:S
1718:D        2 type
1718:D        2   ls_map_def = packed array [1..13] of gbyte;
1719:D        2 const
1719:D        2   ls_map = ls_map_def [0,2,3,4,5,6,1,2,3,4,5,6,1];
1719:S
1719:C        2 begin
1719:C        2   with gle_gcb^ do
1719:C        3     begin
1719:C        3       info1 := ls_map[index];                { map DGL to GLE def }
1719:C        3       info2 := 4;                            { repeat rate 4% }
1719:C        3       if (index > 7) then info3 := 1
1719:C        4       else info3 := 0;                       { linestyle mode }
1720:C        3       info4 := 0;
1720:C        3       gle_linestyle ( gle_gcb );
1720:C        3     end;
1720:C        2 end; { hpgl_linestyle }
1720:S
1720:D        1 procedure hpgl_color ( index : integer );
1720:S
1720:C        2 begin
1720:C        2   gle_gcb^.info1 := index;
1720:C        2   gle_index_color ( gle_gcb );
1721:C        2 end;
1721:S
1721:D        1 procedure hpgl_color_table ( index : integer;
1721:D        2                              parm1 : real;
1721:D        2                              parm2 : real;
1721:D   -24  2                              parm3 : real);
1721:S
1721:C        2 begin
1721:C        2 end;
1721:S
1722:D        1 procedure dgl_hpgl_init(control : integer);
1722:S
1722:D        2 type
1722:D        2   default_poly_table_def = array[1..16] of poly_entry_def;
1722:S
1722:D        2 const
1722:D        2   default_poly_table = default_poly_table_def [
1722:D        2     poly_entry_def [ density :   0.0  , orient :   0.0, edge : true ], { 1 }
1722:D        2     poly_entry_def [ density :  0.125, orient :  90.0, edge : true ], { 2 }
1722:D        2     poly_entry_def [ density :  0.125, orient :   0.0, edge : true ], { 3 }
1723:D        2     poly_entry_def [ density : -0.125, orient :   0.0, edge : true ], { 4 }
1723:D        2     poly_entry_def [ density :  0.125, orient :  45.0, edge : true ], { 5 }
```



```
1723:D        2     poly_entry_def [ density :  0.125, orient : -45.0, edge : true ], { 6 }
1723:D        2     poly_entry_def [ density : -0.125, orient :  45.0, edge : true ], { 7 }
1723:D        2     poly_entry_def [ density :  0.25 , orient :  90.0, edge : true ], { 8 }
1723:D        2     poly_entry_def [ density :  0.25 , orient :   0.0, edge : true ], { 9 }
1723:D        2     poly_entry_def [ density : -0.25 , orient :   0.0, edge : true ], { 10 }
1723:D        2     poly_entry_def [ density :  0.25 , orient :  45.0, edge : true ], { 11 }
1723:D        2     poly_entry_def [ density :  0.25 , orient : -45.0, edge : true ], { 12 }
1723:D        2     poly_entry_def [ density : -0.25 , orient :  45.0, edge : true ], { 13 }
1724:D        2     poly_entry_def [ density : -0.5  , orient :   0.0, edge : true ], { 14 }
1724:D        2     poly_entry_def [ density :  1.0  , orient :   0.0, edge : false], { 15 }
1724:D        2     poly_entry_def [ density :  1.0  , orient :   0.0, edge : true ]];{ 16 }
1724:S
1724:D        2 type
1724:D        2   control_def = packed record
1724:D        2                   case gshortint of
1724:D        2                     0 : (whole : gshortint);
1724:D        2                     1 : (part  : packed record
1724:D        2                                    b15,b14,b13,b12,
1725:D        2                                    b11,b10,b9, b8,
1725:D        2                                    clr_inhibit,b6,b5,b4,
1725:D        2                                    b3,b2,b1,b0 : boolean;
1725:D        2                                  end);
1725:D        2                   end;
1725:S
1725:D        2 var
1725:D   -2   2   temp_control : control_def;
1725:D   -6   2   i : integer;
1725:S
1726:C        2 begin
1726:C        2   with gcb^ do
1726:C        3     begin
1726:C        3       disp_just := lowerleft;
1726:C        3       clipping_support := true;
1726:C        3       retroactive_color_support := false;
1726:C        3       retroactive_polygon_support := false;
1726:C        3       maximum_polygon_vertices := 0; { no hardware support }
1726:C        3       if gle_gcb^.vect_linestyles <> 0 then number_dgl_linestyles := 13
1726:C        4       else                                  number_dgl_linestyles := 7;
1727:C        3       number_markers := 19;
1727:S
1727:C        3       proc_output_esc  := hpgl_output_esc;
1727:C        3       proc_input_esc   := hpgl_input_esc;
1727:C        3       proc_linestyle   := hpgl_linestyle;
1727:C        3       proc_color       := hpgl_color;
1727:C        3       proc_color_table := hpgl_color_table;
1727:C        3       color_table_size := 0;
1727:S
1727:C        3       { allocate polygon table space }
1728:S
1728:C        3       number_polygon_styles := 16;
1728:C        3       (newbytes(poly_table_ptr,number_polygon_styles * 18);)
1728:C        3       poly_table_ptr := addr(poly_table_def_space);
1728:C        3       for i := 1 to poly_table_size do
1728:C        4         poly_table_ptr^[i] := default_poly_table[i];
1728:S
1728:C        3       display_echo_mult := 8;
1728:S
1728:C        3       temp_control.whole := control;
1729:C        3       if not temp_control.part.clr_inhibit then
1729:C        4         with gle_gcb^ do
```

```
17292:C         5           begin
17293:C         5             info1 := -1;                    { clear all planes }
17294:C         5             info2 := dgl_background_index;
17295:C         5             gle_clear ( gle_gcb );
17296:C         5           end;
17297:C         3     end;
17298:C         2 end;
17299:S
17300:C         1 end. { dgl_hpgl }
17301:S

No errors. 18 warnings.
             ***** Nonstandard language features enabled *****
```

```
    1:D         0 {                                                                    }
    2:D         0 { DGL device dependent init routine                                  }
    3:D         0 {                                                                    }
    4:D         0 { Module    = DGL_HPGLI                                              }
    5:D         0 { Programer = BJS                                                    }
    6:D         0 { Date      = 2 -10-83                                               }
    7:D         0 {                                                                    }
    8:D         0 { Purpose: To provide device dependent initialization for the knob.  }
    9:S
   10:D         0 { Rev history                                                        }
   11:D         0 {  Created  - 2 -10-82  BJS                                          }
   12:D         0 {  Modified - XX-XX-XX                                               }
   13:S
   14:S           {     (c) Copyright Hewlett-Packard Company, 1983.
   15:S                 All rights are reserved.  Copying or other
   16:S                 reproduction of this program except for archival
   17:S                 purposes is prohibited without the prior
   18:S                 written consent of Hewlett-Packard Company.
   19:S
   20:S
   21:S                             RESTRICTED RIGHTS LEGEND
   22:S
   23:S                 Use, duplication, or disclosure by the Government
   24:S                 is subject to restrictions as set forth in
   25:S                 paragraph (b) (3) (B) of the Rights in Technical
   26:S                 Data and Computer Software clause in
   27:S                 DAR 7-104.9(a).
   28:S
   29:S                 HEWLETT-PACKARD COMPANY
   30:D         0       Fort Collins, Colorado                                 }
   31:S
   32:D         0 $search 'GLE_LIB',
   33:D         0         'TYPES',
   34:D         0         'DGL_VARS',
   35:D         0         'GEN',
   36:D         0         'DGL_C_OUT'$
   37:D         0 $modcal$
   37:D         0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS ****************** }
   38:S
   39:S            { This include file specifies range checking, debug and other compiler
   40:D         0    options for the graphics library                                  }
   41:S
   42:D         0 $debug OFF$
   43:D         0 $range OFF$
   44:D         0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
   45:D         0 $FLOAT_HDW TEST$
   46:S
   47:S
   48:S
   49:S
   50:D         0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS ****************** }
18000:D         0 $linenum 18000$
18001:S
18002:D         0 module DGL_HPGLI;
18003:S
18004:D         1 export
18005:S
18006:D         1 procedure dgl_hpgli_init;
18007:S
18008:D         1 implement
```



```
18009:S
18010:D         1 import dgl_types,
18011:D         1        dgl_vars,
18012:D         1        gle_types,
18013:D         1        gle_gen,
18014:D         1        gle_geni,
18015:D         1        dgl_gen;
18016:S
18017:D         1 procedure hpgli_sample_locator(    echo   : integer;
18018:D         2                                var rx,ry  : real    );
18019:S
18020:D         2 { Purpose : To sample the locator device                             }
18021:S
18022:D         2 var
18023:D    -8   2   dx, dy : integer;
18024:S
18025:C         2 begin
18026:C         2   with gle_gcbi^ do
18027:C         3     begin
18028:C         3       gle_sample(gle_gcbi);
18029:C         3       convert_itod(info1,info2,dx,dy);
18030:C         3       convert_dtow(dx,dy,rx,ry);
18031:C         3       info1 := echo;
18032:C         3       gle_input_echo (gle_gcbi);      { echo on locator device }
18033:C         3     end;
18034:C         2 end; { sample_locator }
18035:S
18036:D         1 procedure hpgli_await_locator(var echo   : integer;
18037:D         2                               var button : integer;
18038:D         2                               var rx,ry  : real    );
18039:S
18040:S
18041:D         2 { Purpose : To activate the locator, and wait for operator termination      }
18042:S
18043:D         2 var
18044:D    -1   2   echoerror : boolean;
18045:D   -10   2   dx,dy : integer;
18046:D   -18   2   last_x,last_y : integer;
18047:S
18048:C         2 begin
18049:C         2   if (echo < 0) or (echo > 8) then echo := 1;  { ck echo range }
18050:C         2   { ck for display echo, and display not enabled }
18051:C         2   if (not disp_init) and (echo >1)  then
18052:C         3     begin
18053:C         3       echoerror := true;
18054:C         3       echo := 1;
18055:C         3     end
18056:C         3   else
18057:C         3     echoerror := false;
18058:S
18059:C         2   with gcb^,gle_gcbi^ do
18060:C         3     begin
18061:C         3       current_echo_type := echo;
18062:S
18063:C         3       info2 := 0;
18064:C         3       gle_start_digitize (gle_gcbi);
18065:S
18066:C         3       last_x := -32768;
18067:C         3       last_y := -32768;
18068:S
```

```
18069:C         3       if echo > 1 then
18070:C         4         with gle_gcb^ do
18071:C         5           begin
18072:C         5             info1 := d_loc_echo_x;
18073:C         5             info2 := d_loc_echo_y;
18074:C         5             info3 := 1; { on }
18075:C         5             gle_cursor ( gle_gcb );   { perform first echo at lep }
18076:C         5           end;
18077:S
18078:C         3       repeat
18079:C         4         gle_sample ( gle_gcbi );
18080:C         4         convert_ltod(info1,info2,dx,dy);
18081:C         4         button := info3;
18082:S
18083:C         4         if (dx <> last_x) or (dy <> last_y) then
18084:C         5           begin
18085:C         5             last_x := dx;
18086:C         5             last_y := dy;
18087:C         5             if (echo > 1) and (not disp_eq_loc) then
18088:C         6               with gle_gcb^ do
18089:C         7                 begin
18090:C         7                   info1 := dx;
18091:C         7                   info2 := dy;
18092:C         7                   info3 := 1;
18093:C         7                   gle_cursor ( gle_gcb );
18094:C         7                 end;
18095:C         5           end;
18096:C         4       until button = -1;
18097:S
18098:C         3       gle_get_digitize(gle_gcbi);
18099:C         3       button := info3;
18100:S
18101:C         3       convert_ltod(info1,info2,dx,dy);
18102:C         3       convert_dtow(dx,dy,rx,ry);
18103:C         3       adjust_return_echo ( rx, ry );
18104:S
18105:C         3       if echo > 1 then
18106:C         4         with gle_gcb^ do
18107:C         5           begin
18108:C         5             info3 := 0;
18109:C         5             gle_cursor(gle_gcb);          { remove cursor from screen }
18110:C         5           end;
18111:S
18112:C         3       if echo = 1 then
18113:C         4         begin
18114:C         4           info1 := echo;
18115:C         4           gle_input_echo ( gle_gcbi );
18116:C         4         end;
18117:C         3   end;
18118:S
18119:C         2   if echoerror then error (err_echo_dis_int);
18120:S
18121:C         2 end; { await_locator }
18122:S
18123:D         1 procedure dgl_hpgli_init;
18124:S
18125:C         2 begin
18126:C         2   with gcb^ do
18127:C         3     begin
18128:C         3       proc_await_locator := hpgli_await_locator;
```



```
18129:C         3       proc_sample_locator := hpgli_sample_locator;
18130:C         3     end;
18131:C         2 end;
18132:S
18133:C         1 end. { dgl_hpgli_init }
18134:S

No errors. No warnings.
              ***** Nonstandard language features enabled *****
```

DGL_INQ

Description

DGL_INQ provides user inquiry code for DGL.

Requirements

DGL_TYPES, DGL_VARS, GLE_TYPES, GLE_RAS_OUT, SYSGLOBALS, SYSDEVS, GLE_ARAS_OUT, GLE_ASCLIP, GLE_STEXT, GLE_ASTEXT, GLE_SCLIP, GLE_ASCLIP, GLE_SMARK, and GLE_AUTL.

```
     1:D        0 {                                                                    }
     2:D        0 { Pascal work station graphics library                               }
     3:D        0 {                                                                    }
     4:D        0 { Module    = DGL_INQ                                                }
     5:D        0 { Programer = BJS                                                    }
     6:D        0 { Date      = 4/13/82                                                }
     7:S
     8:D        0 { Purpose:    Holds all code for user inquires to the DGL system.    }
     9:S
    10:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    11:S                All rights are reserved.  Copying or other
    12:S                reproduction of this program except for archival
    13:S                purposes is prohibited without the prior
    14:S                written consent of Hewlett-Packard Company.
    15:S
    16:S
    17:S                            RESTRICTED RIGHTS LEGEND
    18:S
    19:S                Use, duplication, or disclosure by the Government
    20:S                is subject to restrictions as set forth in
    21:S                paragraph (b) (3) (B) of the Rights in Technical
    22:S                Data and Computer Software clause in
    23:S                DAR 7-104.9(a).
    24:S
    25:S                HEWLETT-PACKARD COMPANY
    26:D        0       Fort Collins, Colorado                                 }
    27:S
    28:D        0 $modcal$
    29:D        0 $search 'TYPES',
    30:D        0         'DGL_VARS',
    31:D        0         'GEN',
    32:D        0         'GLE_LIB'$
    33:D        0 $include 'OPTIONS'$
    34:S
    35:S           { This include file specifies range checking, debug and other compiler
    36:D        0    options for the graphics library                                  }
    37:S
    38:D        0 $debug OFF$
    39:D        0 $range OFF$
    40:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    41:D        0 $FLOAT_HDW TEST$
    42:S
    43:S
    44:S
    45:D        0 $include 'OPTIONS'$
  6000:D        0 $linenum 6000$
  6001:S
  6002:D        0 module dgl_inq;
  6003:S
  6004:D        1 import dgl_types;
  6005:S
  6006:D        1 export
  6007:S
  6008:D        1 procedure inq_ws (       opcode : integer;
  6009:D        2                          ssize  : integer;
  6010:D        2                          isize  : integer;
  6011:D        2                          rsize  : integer;
  6012:D        2                 anyvar slist  : gchar_list;
  6013:D        2                 anyvar ilist  : gint_list;
```



```
  6014:D        2                 anyvar rlist  : greal_list;
  6015:D        2                    var ierr   : integer);
  6016:S
  6017:D        1 procedure inq_color_table ( index : integer;
  6018:D        2                             var parm1 : real;
  6019:D        2                             var parm2 : real;
  6020:D        2                             var parm3 : real);
  6021:D        1
  6022:D        1 procedure inq_pgn_table ( index : integer;
  6023:D        2                           var pdensity : real;
  6024:D        2                           var porient : real;
  6025:D        2                           var pedge   : integer);
  6026:S
  6027:D        1 implement
  6028:S
  6029:D        1 import dgl_vars,
  6030:D        1        gle_ras_out,
  6031:D        1        dgl_gen;
  6032:S
  6033:S
  6034:D        1 procedure inq_color_table ( index : integer;
  6035:D        2                             var parm1 : real;
  6036:D        2                             var parm2 : real;
  6037:D        2                             var parm3 : real);
  6038:S
  6039:D        2 { Purpose:  To return information about a color table location       }
  6040:S
  6041:C        2 begin
  6042:C        2   ck_system_init;
  6043:C        2   ck_display_init;
  6044:C        2   with gcb^ do
  6045:C        3     begin
  6046:C        3       if color_table_size = 0 then error(err_no_ctable);
  6047:C        3       if (index < 0) or (index > color_table_size) then error(err_bad_parms);
  6048:C        3       with color_table_ptr^[index] do
  6049:C        4         if dgl_current_color_model = 1 { rgb } then
  6050:C        5           begin
  6051:C        5             parm1 := red;
  6052:C        5             parm2 := green;
  6053:C        5             parm3 := blue;
  6054:C        5           end
  6055:C        5         else
  6056:C        5           convert_rgb_to_hsl(red,green,blue,parm1,parm2,parm3);
  6057:C        3     end;
  6058:C        2 end;
  6059:S
  6060:D        1  procedure inq_pgn_table ( index : integer;
  6061:D        2                           var pdensity : real;
  6062:D        2                           var porient : real;
  6063:D        2                           var pedge   : integer);
  6064:S
  6065:D        2 { Purpose:  To return information about a polygon table location     }
  6066:S
  6067:C        2 begin
  6068:C        2   ck_system_init;
  6069:C        2   ck_display_init;
  6070:S
  6071:C        2   with gcb^ do
  6072:C        3     begin
  6073:C        3       if (index < 1) or (index > number_polygon_styles) then error(err_bad_parms);
```

```
6074:C        3       with poly_table_ptr^ [ index ] do
6075:C        4         begin
6076:C        4           pdensity := density;
6077:C        4           porient := orient;
6078:C        4           if edge then pedge := 1
6079:C        5           else pedge := 0;
6080:C        4         end;
6081:C        3     end;
6082:C        2 end;
6083:S
6084:D        1 procedure inq_ws (       opcode : integer;
6085:D        2                          ssize  : integer;
6086:D        2                          isize  : integer;
6087:D        2                          rsize  : integer;
6088:D        2                  anyvar slist  : gchar_list;
6089:D        2                  anyvar ilist  : gint_list;
6090:D        2                  anyvar rlist  : greal_list;
6091:D        2                     var ierr   : integer);
6092:S
6093:D        2 { Purpose:  To return information some part of the DGL graphics system.   }
6094:S
6095:D        2 label 1;
6096:S
6097:D        2 const numopcodes = 40;
6098:S
6099:D        2 type
6100:D        2   opcode_type_def = (inq_250,   inq_251,   inq_252,   inq_253,   inq_254,
6101:D        2                      inq_255,   inq_256,   inq_257,   inq_258,   inq_259,
6102:D        2                      inq_450,   inq_451,
6103:D        2                      inq_1050,  inq_1051,  inq_1052,  inq_1053,  inq_1054,
6104:D        2                                 inq_1056,  inq_1057,             inq_1059,
6105:D        2                      inq_1060,             inq_1062,  inq_1063,  inq_1064,
6106:D        2                      inq_1065,  inq_1066,  inq_1067,  inq_1068,  inq_1069,
6107:D        2                      inq_1070,  inq_1071,  inq_1072,  inq_1073,  inq_1074,
6108:D        2                      inq_1075,  inq_1076,
6109:D        2                      inq_11050, inq_11052,
6110:D        2                      inq_12050,
6111:D        2                      inq_13052                                          );
6112:S
6113:D        2   opcode_def = 0..32767;
6114:D        2   opcode_list_entry_def = packed record
6115:D        2       opcode      : opcode_def;
6116:D        2       opcode_type : opcode_type_def;
6117:D        2     end;
6118:D        2   opcode_list_def = packed array [1..numopcodes] of opcode_list_entry_def;
6119:S
6120:D        2 const
6121:D        2   opcode_list = opcode_list_def[
6122:D        2       opcode_list_entry_def[opcode : 250,   opcode_type : inq_250   ],
6123:D        2       opcode_list_entry_def[opcode : 251,   opcode_type : inq_251   ],
6124:D        2       opcode_list_entry_def[opcode : 252,   opcode_type : inq_252   ],
6125:D        2       opcode_list_entry_def[opcode : 253,   opcode_type : inq_253   ],
6126:D        2       opcode_list_entry_def[opcode : 254,   opcode_type : inq_254   ],
6127:D        2       opcode_list_entry_def[opcode : 255,   opcode_type : inq_255   ],
6128:D        2       opcode_list_entry_def[opcode : 256,   opcode_type : inq_256   ],
6129:D        2       opcode_list_entry_def[opcode : 257,   opcode_type : inq_257   ],
6130:D        2       opcode_list_entry_def[opcode : 258,   opcode_type : inq_258   ],
6131:D        2       opcode_list_entry_def[opcode : 259,   opcode_type : inq_259   ],
6132:D        2       opcode_list_entry_def[opcode : 450,   opcode_type : inq_450   ],
6133:D        2       opcode_list_entry_def[opcode : 451,   opcode_type : inq_451   ],
```



```
6134:D        2       opcode_list_entry_def[opcode : 1050,  opcode_type : inq_1050  ],
6135:D        2       opcode_list_entry_def[opcode : 1051,  opcode_type : inq_1051  ],
6136:D        2       opcode_list_entry_def[opcode : 1052,  opcode_type : inq_1052  ],
6137:D        2       opcode_list_entry_def[opcode : 1053,  opcode_type : inq_1053  ],
6138:D        2       opcode_list_entry_def[opcode : 1054,  opcode_type : inq_1054  ],
6139:D        2       opcode_list_entry_def[opcode : 1056,  opcode_type : inq_1056  ],
6140:D        2       opcode_list_entry_def[opcode : 1057,  opcode_type : inq_1057  ],
6141:D        2       opcode_list_entry_def[opcode : 1059,  opcode_type : inq_1059  ],
6142:D        2       opcode_list_entry_def[opcode : 1060,  opcode_type : inq_1060  ],
6143:D        2       opcode_list_entry_def[opcode : 1062,  opcode_type : inq_1062  ],
6144:D        2       opcode_list_entry_def[opcode : 1063,  opcode_type : inq_1063  ],
6145:D        2       opcode_list_entry_def[opcode : 1064,  opcode_type : inq_1064  ],
6146:D        2       opcode_list_entry_def[opcode : 1065,  opcode_type : inq_1065  ],
6147:D        2       opcode_list_entry_def[opcode : 1066,  opcode_type : inq_1066  ],
6148:D        2       opcode_list_entry_def[opcode : 1067,  opcode_type : inq_1067  ],
6149:D        2       opcode_list_entry_def[opcode : 1068,  opcode_type : inq_1068  ],
6150:D        2       opcode_list_entry_def[opcode : 1069,  opcode_type : inq_1069  ],
6151:D        2       opcode_list_entry_def[opcode : 1070,  opcode_type : inq_1070  ],
6152:D        2       opcode_list_entry_def[opcode : 1071,  opcode_type : inq_1071  ],
6153:D        2       opcode_list_entry_def[opcode : 1072,  opcode_type : inq_1072  ],
6154:D        2       opcode_list_entry_def[opcode : 1073,  opcode_type : inq_1073  ],
6155:D        2       opcode_list_entry_def[opcode : 1074,  opcode_type : inq_1074  ],
6156:D        2       opcode_list_entry_def[opcode : 1075,  opcode_type : inq_1075  ],
6157:D        2       opcode_list_entry_def[opcode : 1076,  opcode_type : inq_1076  ],
6158:D        2       opcode_list_entry_def[opcode : 11050, opcode_type : inq_11050 ],
6159:D        2       opcode_list_entry_def[opcode : 11052, opcode_type : inq_11052 ],
6160:D        2       opcode_list_entry_def[opcode : 12050, opcode_type : inq_12050 ],
6161:D        2       opcode_list_entry_def[opcode : 13052, opcode_type : inq_13052 ]];
6162:S
6163:D        2 var
6164:D    -2  2   index : gshortint;
6165:D   -14  2   workstring : string[10];
6166:D   -18  2   strcnt : integer;
6167:S
6168:C        2 begin
6169:C        2   ck_system_init;
6170:S
6171:C        2   ierr := opcode_ck ( opcode,isize,rsize);  { ck for good parms }
6172:S
6173:C        2   { Find opcode in list }
6174:C        2   for index := 1 to numopcodes do
6175:C        3     if opcode_list[index].opcode = opcode then goto 1;
6176:C        2   ierr := 1;
6177:S
6178:C        2 1:
6179:S
6180:C        2   if ierr = 0 then
6181:C        3     with gcb^,gle_gcb^ do
6182:C        4       case opcode_list[index].opcode_type of
6183:C        5         inq_250 : { return cell size (250) }
6184:C        5           begin
6185:C        5             rlist[1] := gcb^.dgl_char_width;
6186:C        5             rlist[2] := gcb^.dgl_char_height;
6187:C        5           end;
6188:C        5
6189:C        5         inq_251 : { return marker cell size (251) }
6190:C        5           begin
6191:C        5             rlist[1] := marker_size_x * xdtow_scale;
6192:C        5             rlist[2] := marker_size_y * ydtow_scale;
6193:C        5           end;
```

```
6194:S
6195:C          5          inq_252 : { return display resolution (252) }
6196:C          5            if disp_init then
6197:C          6              begin
6198:C          6                with gcb^,raster_device_rec_ptr(dev_dep_stuff)^ do
6199:C          7                  begin
6200:C          7                    rlist[1] := display_res_x;
6201:C          7                    rlist[2] := display_res_y;
6202:C          7                  end;
6203:C          6              end
6204:C          6            else
6205:C          6              begin
6206:C          6                rlist[1] := 0;
6207:C          6                rlist[2] := 0;
6208:C          6              end;
6209:S
6210:C          5          inq_253 : { return max display dimensions (253) }
6211:C          5            if disp_init then
6212:C          6              with max_disp_lim do
6213:C          7                begin
6214:C          7                with gcb^,raster_device_rec_ptr(dev_dep_stuff)^ do
6215:C          8                  begin
6216:C          8                    rlist[1] := (xmax - xmin) / display_res_x;
6217:C          8                    rlist[2] := (ymax - ymin) / display_res_y;
6218:C          8                  end;
6219:C          7                end
6220:C          7            else
6221:C          6              begin
6222:C          6                rlist[1] := 0;
6223:C          6                rlist[2] := 0;
6224:C          6              end;
6225:S
6226:C          5          inq_254 : { return aspect ratios (254) }
6227:C          5            begin
6228:C          5              rlist[1] := aspect_ratio;
6229:C          5              rlist[2] := log_aspect;
6230:C          5            end;
6231:S
6232:C          5          inq_255 : { return locator resolution (255) }
6233:C          5            if loc_init then
6234:C          6              with gle_gcbi^ do
6235:C          7                begin
6236:C          7                  rlist[1] := input_res_x;
6237:C          7                  rlist[2] := input_res_y;
6238:C          7                end
6239:C          7            else
6240:C          6              begin
6241:C          6                rlist[1] := 0;
6242:C          6                rlist[2] := 0;
6243:C          6              end;
6244:S
6245:C          5          inq_256 : { max locator dimensions (256) }
6246:C          5            if loc_init then
6247:C          6              with gle_gcbi^, max_loc_lim do
6248:C          7                begin
6249:C          7                  rlist[1] := (xmax - xmin) / input_res_x;
6250:C          7                  rlist[2] := (ymax - ymin) / input_res_y;
6251:C          7                end
6252:C          7            else
6253:C          6              begin
```



```
6254:C          6                rlist[1] := 0;
6255:C          6                rlist[2] := 0;
6256:C          6              end;
6257:S
6258:C          5          inq_257 : { locator echo pos (257) }
6259:C          5            begin
6260:C          5              rlist[1] := w_loc_echo_x;
6261:C          5              rlist[2] := w_loc_echo_y;
6262:C          5            end;
6263:S
6264:C          5          inq_258 : { current virtual limits (258) }
6265:C          5            with cur_vir_lim do
6266:C          6              begin
6267:C          6                rlist[1] := xlim;
6268:C          6                rlist[2] := ylim;
6269:C          6              end;
6270:S
6271:C          5          inq_259 : { return cp (259) }
6272:C          5            begin
6273:C          5              if int_cp then
6274:C          6                begin
6275:C          6                  rlist[1] := world_int_cpx;
6276:C          6                  rlist[2] := world_int_cpy;
6277:C          6                end
6278:C          6              else
6279:C          6                begin
6280:C          6                  rlist[1] := world_real_cpx;
6281:C          6                  rlist[2] := world_real_cpy;
6282:C          6                end;
6283:C          5            end;
6284:S
6285:C          5          inq_450: { return window limits (450) }
6286:C          5            with window_lim do
6287:C          6              begin
6288:C          6                rlist[1] := xmin;
6289:C          6                rlist[2] := xmax;
6290:C          6                rlist[3] := ymin;
6291:C          6                rlist[4] := ymax;
6292:C          6              end;
6293:S
6294:C          5          inq_451: { return viewport limits (451) }
6295:C          5            with viewport_lim do
6296:C          6              begin
6297:C          6                rlist[1] := xmin;
6298:C          6                rlist[2] := xmax;
6299:C          6                rlist[3] := ymin;
6300:C          6                rlist[4] := ymax;
6301:C          6              end;
6302:S
6303:C          5          inq_1050: { Clipping supported at physical limits (1050) }
6304:C          5            begin
6305:C          5              if clipping_support then ilist [1] := 1
6306:C          6              else                     ilist [1] := 0;
6307:C          5            end;
6308:S
6309:C          5          inq_1051: { return display justification info (1051) }
6310:C          5            begin
6311:C          5              if disp_just = centered then ilist [1] := 0
6312:C          6              else                         ilist [1] := 1;
6313:C          5            end;
```

```
6314:S
6315:C      5         inq_1052: { return info about drawing in the background color (1052) }
6316:C      5           with gle_gcb^ do
6317:C      6             begin
6318:C      6               if disp_init then ilist [1] := background
6319:C      7               else              ilist [1] := 0;
6320:C      6             end;
6321:S
6322:C      5         inq_1053: { return color palette info (1053) }
6323:C      5           begin
6324:C      5             if disp_init then ilist[1] := pallette
6325:C      6             else              ilist[1] := 0;
6326:C      5           end;
6327:S
6328:C      5         inq_1054: { return color gamut info (1054) }
6329:C      5           begin
6330:C      5             if disp_init then ilist[1] := gamut
6331:C      6             else              ilist[1] := 0;
6332:C      5           end;
6333:S
6334:C      5         inq_1056: { return number linestyles (1056) }
6335:C      5           begin
6336:C      5             if disp_init then ilist[1] := number_dgl_linestyles
6337:C      6             else              ilist[1] := 0;
6338:C      5           end;
6339:S
6340:C      5         inq_1057: { return number linewidths (1057) }
6341:C      5           begin
6342:C      5             if disp_init then ilist[1] := linewidths
6343:C      6             else              ilist[1] := 0;
6344:C      5           end;
6345:S
6346:C      5         inq_1059: { return number markers (1059) }
6347:C      5           begin
6348:C      5             if disp_init then ilist[1] := number_markers
6349:C      6             else              ilist[1] := 0;
6350:C      5           end;
6351:S
6352:C      5         inq_1060: { return current color (1060) }
6353:C      5           begin
6354:C      5             ilist[1] := dgl_current_color;
6355:C      5           end;
6356:S
6357:C      5         inq_1062: { return current linestyle (1062) }
6358:C      5           begin
6359:C      5             ilist[1] := dgl_current_linestyle;
6360:C      5           end;
6361:S
6362:C      5         inq_1063: { return current linewidth (1063) }
6363:C      5           begin
6364:C      5             ilist[1] := dgl_current_linewidth;
6365:C      5           end;
6366:S
6367:C      5         inq_1064: { return current timming mode }
6368:C      5           begin
6369:C      5             ilist[1] := dgl_current_timming_mode;
6370:C      5           end;
6371:S
6372:C      5         inq_1065: { return number polygon styles supported ( 1065) }
6373:C      5           begin
```



```
6374:C      5             if disp_init then ilist[1] := number_polygon_styles
6375:C      6             else              ilist[1] := 0;
6376:C      5           end;
6377:S
6378:C      5         inq_1066: { return current polygon color (1066) }
6379:C      5           begin
6380:C      5             ilist[1] := dgl_current_polygon_color;
6381:C      5           end;
6382:S
6383:C      5         inq_1067: { return current polygon style (1067) }
6384:C      5           begin
6385:C      5             ilist[1] := dgl_current_polygon_style;
6386:C      5           end;
6387:S
6388:C      5         inq_1068: { return maximum polygon vertices supported (1068) }
6389:C      5           begin
6390:C      5             if disp_init then ilist[1] := maximum_polygon_vertices
6391:C      6             else              ilist[1] := 0;
6392:C      5           end;
6393:S
6394:C      5         inq_1069: { retroactive polygon support (1069) }
6395:C      5           begin
6396:C      5             if disp_init and retroactive_polygon_support then ilist[1] := 1
6397:C      6             else                                              ilist[1] := 0;
6398:C      5           end;
6399:S
6400:C      5         inq_1070: { device dependent polygons ( 1070 ) }
6401:C      5           begin
6402:C      5             if disp_init and (polygon_support = 1) then ilist [1] := 1
6403:C      6             else                                        ilist [1] := 0;
6404:C      5           end;
6405:S
6406:C      5         inq_1071: { retroactive color support ( 1071 ) }
6407:C      5           begin
6408:C      5             if disp_init and retroactive_color_support then ilist [1] := 1
6409:C      6             else                                            ilist [1] := 0;
6410:C      5           end;
6411:S
6412:C      5         inq_1072: { redef of background ( 1072 ) }
6413:C      5           begin
6414:C      5             if disp_init then ilist [1] := redef_background
6415:C      6             else              ilist [1] := 0;
6416:C      5           end;
6417:S
6418:C      5         inq_1073: { redef of color capability table ( 1073 ) }
6419:C      5           begin
6420:C      5             if disp_init and (color_table_size > 0) then ilist [1] := 1
6421:C      6             else                                         ilist [1] := 0;
6422:C      5           end;
6423:S
6424:C      5         inq_1074: { return current color model ( 1074 ) }
6425:C      5           begin
6426:C      5             ilist [1] := dgl_current_color_model;
6427:C      5           end;
6428:S
6429:C      5         inq_1075: { return color capability table size ( 1075 ) }
6430:C      5           ilist [1] := color_table_size;
6431:S
6432:C      5         inq_1076: { return current polygon linestyle ( 1076 ) }
6433:C      5           ilist [1] := dgl_current_polygon_linestyle;
```

```
6434:S
6435:C  5         inq_11050: { return display device association (11050) }
6436:C  5           begin
6437:C  5             if not disp_init then
6438:C  6               begin
6439:C  6                 if ssize >= 1 then
6440:C  7                   begin
6441:C  7                     slist[1] := '0';
6442:C  7                     ilist[1] := 1;
6443:C  7                   end
6444:C  7                 else ierr := 4;
6445:C  6               end
6446:C  6             else
6447:C  6             if disp_file_name <> '' then
6448:C  7               begin
6449:C  7                 for index := 1 to strlen(disp_file_name) do
6450:C  8                   slist[index] := disp_file_name[index];
6451:C  7                 ilist [1] := strlen(disp_file_name);
6452:C  7               end
6453:C  7             else
6454:C  7               begin
6455:C  7                 setstrlen(workstring,0);
6456:C  7                 strwrite(workstring,1,strcnt,disp_dev_adr:0);
6457:C  7                 strcnt := strcnt - 1;
6458:C  7                 if ssize >= strcnt then
6459:C  8                   begin
6460:C  8                     for index := 1 to strcnt do
6461:C  9                       slist[index] := workstring[index];
6462:C  8                     ilist[1] := strcnt;
6463:C  8                   end
6464:C  8                 else
6465:C  8                   ierr := 4;
6466:C  7               end;
6467:C  5           end;
6468:S
6469:C  5         inq_11052: { return locator device association (11052) }
6470:C  5           begin
6471:C  5             if not loc_init then
6472:C  6               begin
6473:C  6                 if ssize >= 1 then
6474:C  7                   begin
6475:C  7                     slist[1] := '0';
6476:C  7                     ilist[1] := 1;
6477:C  7                   end
6478:C  7                 else ierr := 4;
6479:C  6               end
6480:C  6             else
6481:C  6               begin
6482:C  6                 setstrlen(workstring,0);
6483:C  6                 strwrite(workstring,1,strcnt,loc_dev_adr:0);
6484:C  6                 strcnt := strcnt - 1;
6485:C  6                 if ssize >= strcnt then
6486:C  7                   begin
6487:C  7                     for index := 1 to strcnt do
6488:C  8                       slist[index] := workstring[index];
6489:C  7                     ilist[1] := strcnt;
6490:C  7                   end
6491:C  7                 else
6492:C  7                   ierr := 4;
6493:C  6               end;
```



```
6494:C  5           end;
6495:S
6496:C  5         inq_12050: { return display info (12050) }
6497:C  5           begin
6498:C  5             if not disp_init then
6499:C  6               begin
6500:C  6                 if ssize >= 6 then
6501:C  7                   begin
6502:C  7                     for index := 1 to 6 do
6503:C  8                       slist[index] := ' ';
6504:C  7                     ilist[1] := 6;
6505:C  7                     ilist[2] := 0; { disabled }
6506:C  7                   end
6507:C  7                 else ierr := 4;
6508:C  6               end
6509:C  6             else
6510:C  6               begin
6511:C  6                 if ssize >= display_name_char_count then
6512:C  7                   begin
6513:C  7                     for index := 1 to display_name_char_count do
6514:C  8                       slist[index] := display_name[index];
6515:C  7                     ilist[1] := display_name_char_count;
6516:C  7                     ilist[2] := 1; { enabled }
6517:C  7                   end
6518:C  7                 else
6519:C  7                   ierr := 4;
6520:C  6               end;
6521:C  5           end;
6522:S
6523:C  5         inq_13052: { return locator info (13052) }
6524:C  5           begin
6525:C  5             if not loc_init then
6526:C  6               begin
6527:C  6                 if ssize >= 6 then
6528:C  7                   begin
6529:C  7                     for index := 1 to 6 do
6530:C  8                       slist[index] := ' ';
6531:C  7                     ilist[1] := 6;
6532:C  7                     ilist[2] := 0; { disabled }
6533:C  7                     ilist[3] := 0;
6534:C  7                   end
6535:C  7                 else ierr := 4;
6536:C  6               end
6537:C  6             else
6538:C  6               begin
6539:C  6                 with gle_gcbi^ do
6540:C  7                   if ssize >= input_name_char_count then
6541:C  8                     begin
6542:C  8                       for index := 1 to input_name_char_count do
6543:C  9                         slist[index] := input_name[index];
6544:C  8                       ilist[1] := input_name_char_count;
6545:C  8                       ilist[2] := 1; { enabled }
6546:C  8                       if gle_gcbi^.input_handler_name = 'KNOB   ' then
6547:C  9                       ilist[3] := 255
6548:C  9                       else ilist[3] := 1;
6549:C  8                     end
6550:C  8                   else
6551:C  8                     ierr := 4;
6552:C  6               end;
6553:C  5           end;
```

```
6554:C          5         end; { of case }
6555:S
6556:C          2 end;
6557:S
6558:C          1 end.  { of module }
6559:S

No errors. No warnings.
              ***** Nonstandard language features enabled *****
```

DGL_KNOB

Description

DGL_KNOB provides DGL-level knob handler and initialization.

Requirements

DGL_TYPES, DGL_VARS, GLE_TYPES, GLE_GEN, GLE_GENI, DGL_GEN, ASM, and SYSGLOBALS.

```
    1:D        0 {                                                                    }
    2:D        0 { DGL device dependent init routine                                  }
    3:D        0 {                                                                    }
    4:D        0 { Module    = DGL_KN0B                                               }
    5:D        0 { Programer = BJS                                                    }
    6:D        0 { Date      = 2 -10-83                                               }
    7:D        0 {                                                                    }
    8:D        0 { Purpose: To provide device dependent initialization for the knob.  }
    9:S
   10:D        0 { Rev history                                                        }
   11:D        0 {  Created  - 2 -10-82  BJS                                          }
   12:D        0 {  Modified - XX-XX-XX                                               }
   13:S
   14:S          {     (c) Copyright Hewlett-Packard Company, 1983.
   15:S                All rights are reserved.  Copying or other
   16:S                reproduction of this program except for archival
   17:S                purposes is prohibited without the prior
   18:S                written consent of Hewlett-Packard Company.
   19:S
   20:S
   21:S                             RESTRICTED RIGHTS LEGEND
   22:S
   23:S                Use, duplication, or disclosure by the Government
   24:S                is subject to restrictions as set forth in
   25:S                paragraph (b) (3) (B) of the Rights in Technical
   26:S                Data and Computer Software clause in
   27:S                DAR 7-104.9(a).
   28:S
   29:S                HEWLETT-PACKARD COMPANY
   30:D        0       Fort Collins, Colorado                                 }
   31:S
   32:D        0 $search 'GLE_LIB',
   33:D        0         'TYPES',
   34:D        0         'DGL_VARS',
   35:D        0         'GEN',
   36:D        0         'DGL_C_OUT'$
   37:D        0 $modcal$
   37:D        0 $include 'OPTIONS'$  { ********************* COMPILER OPTIONS ****************** }
   38:S
   39:S           { This include file specifies range checking, debug and other compiler
   40:D        0    options for the graphics library                                  }
   41:S
   42:D        0 $debug OFF$
   43:D        0 $range OFF$
   44:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
   45:D        0 { LOAT_HDW TEST$
   46:S
   47:S
   48:S
   49:S
   50:D        0 $include 'OPTIONS'$  { ********************* COMPILER OPTIONS ****************** }
18000:D        0 $linenum 18000$
18001:S
18002:D        0 module DGL_KN0B;
18003:S
18004:D        1 export
18005:S
18006:D        1 procedure dgl_knob_init;
18007:S
18008:D        1 implement
```



```
18009:S
18010:D        1 import dgl_types,
18011:D        1        dgl_vars,
18012:D        1        gle_types,
18013:D        1        gle_gen,
18014:D        1        gle_geni,
18015:D        1        dgl_gen,
18016:D        1        dgl_config_out;
18017:S
18018:D        1 var
18019:D        1   last_knob_rx,
18020:D   -16  1   last_knob_ry : real;
18021:D   -16  1   last_knob_dx,
18022:D   -24  1   last_knob_dy : integer;
18023:S
18024:D        1 procedure knob_sample_locator(    echo    : integer;
18025:D        2                               var rx,ry  : real    );
18026:S
18027:D        2 { Purpose : To sample the locator device                             }
18028:S
18029:C        2 begin
18030:C        2   rx := last_knob_rx;
18031:C        2   ry := last_knob_ry;
18032:C        2   gle_gcbi^.info1 := echo;
18033:C        2   gle_input_echo (gle_gcbi);      { echo on locator device }
18034:C        2 end; { sample_locator }
18035:S
18036:D        1 procedure knob_await_locator(  var   echo    : integer;
18037:D        2                                var button    : integer;
18038:D        2                                var rx,ry     : real    );
18039:S
18040:S
18041:D        2 { Purpose : To activate the locator, and wait for operator termination      }
18042:S
18043:D        2 var
18044:D    -1  2   echoerror : boolean;
18045:D   -10  2   dx,dy : integer;
18046:D   -18  2   last_x,last_y : integer;
18047:D   -22  2   echo_gcb       : graphics_control_block_ptr;
18048:S
18049:C        2 begin
18050:C        2   if (echo < 0) or (echo > 8) then echo := 1;  { ck echo range }
18051:C        2   { ck for display echo, and display not enabled }
18052:C        2   if (not disp_init) and (echo >1)  then
18053:C        3     begin
18054:C        3       echoerror := true;
18055:C        3       echo := 1;
18056:C        3     end
18057:C        3   else
18058:C        3     echoerror := false;
18059:S
18060:C        2   with gcb^,gle_gcbi^ do
18061:C        3     begin
18062:C        3       current_echo_type := echo;
18063:S
18064:C        3       if echo < 2 then
18065:C        4         begin
18066:C        4           input_cpx := last_knob_dx;
18067:C        4           input_cpy := last_knob_dy;
18068:C        4           echo_gcb := gle_knob_echo_gcb;
```

```
 18069:S
 18070:C        4           if echo = 1 then
 18071:C        5             begin
 18072:C        5               with gle_knob_echo_gcb^ do
 18073:C        6                 begin
 18074:C        6                   info2 := 1;  { do not init DGL stuff }
***WARNING: (line18075): 'ADDR' of a constant may not be supported on other implementations
 18075:C        6                   device_info := ADDR('3 ');
 18076:C        6                   device_info.char_count := 1;
 18077:C        6                   configure_gle ( gle_knob_echo_gcb );
 18078:C        6                   if error_return <> 0 then error (err_no_display_hardware);
 18079:S
 18080:C        6                   { define echo edges }
 18081:C        6                   input_min_x := 0;
 18082:C        6                   input_max_x := display_max_x;
 18083:C        6                   input_min_y := 0;
 18084:C        6                   input_max_y := display_max_y;;
 18085:C        6                 end;
 18086:C        5               end;
 18087:C        4             info2 := 1;
 18088:C        4           end
 18089:C        4         else
 18090:C        4           begin
 18091:C        4             echo_gcb := gle_gcb;
 18092:C        4             with cur_disp_lim do
 18093:C        5               begin
 18094:C        5                 input_min_x := trunc(xmin+0.5);
 18095:C        5                 input_max_x := trunc(xmax+0.5);
 18096:C        5                 input_min_y := trunc(ymin+0.5);
 18097:C        5                 input_max_y := trunc(ymax+0.5);
 18098:C        5                 input_cpx := d_loc_echo_x;
 18099:C        5                 input_cpy := d_loc_echo_y;
 18100:C        5                 info2 := display_echo_mult;
 18101:C        5               end;
 18102:C        4           end;
 18103:S
 18104:C        3         gle_start_digitize (gle_gcbi);
 18105:S
 18106:C        3         last_x := -32768;
 18107:C        3         last_y := -32768;
 18108:S
 18109:C        3         repeat
 18110:C        4           gle_sample ( gle_gcbi );
 18111:C        4           button := gle_gcbi^.info3;
 18112:C        4           dx := info1;
 18113:C        4           dy := info2;
 18114:S
 18115:C        4           if (dx <> last_x) or (dy <> last_y) then
 18116:C        5             begin
 18117:C        5               last_x := dx;
 18118:C        5               last_y := dy;
 18119:C        5               if (echo > 0) then
 18120:C        6                 with echo_gcb^ do
 18121:C        7                   begin
 18122:C        7                     info1 := dx;
 18123:C        7                     info2 := dy;
 18124:C        7                     info3 := 1;
 18125:C        7                     gle_cursor ( echo_gcb );
 18126:C        7                   end;
 18127:C        5             end;
```



```
 18128:C        4         until button = -1;
 18129:S
 18130:C        3         gle_get_digitize(gle_gcbi);
 18131:C        3         button := info3;
 18132:S
 18133:C        3         if echo > 1 then
 18134:C        4           begin
 18135:C        4             dx := info1;
 18136:C        4             dy := info2;
 18137:C        4           end
 18138:C        4         else
 18139:C        4           begin
 18140:C        4             last_knob_dx := dx;                    { save before convert }
 18141:C        4             last_knob_dy := dy;
 18142:C        4             convert_ltod(info1,info2,dx,dy);
 18143:C        4           end;
 18144:S
 18145:C        3         convert_dtow(dx,dy,rx,ry);
 18146:C        3         adjust_return_echo ( rx, ry );
 18147:S
 18148:C        3         last_knob_rx := rx;
 18149:C        3         last_knob_ry := ry;
 18150:S
 18151:C        3         if echo > 0 then
 18152:C        4           with echo_gcb^ do
 18153:C        5             begin
 18154:C        5               info3 := 0;
 18155:C        5               gle_cursor(echo_gcb);          { remove cursor from screen }
 18156:C        5             end;
 18157:S
 18158:C        3         if echo = 1 then
 18159:C        4           begin
 18160:C        4             info1 := echo;
 18161:C        4             gle_input_echo ( gle_gcbi );
 18162:C        4           end;
 18163:C        3     end;
 18164:S
 18165:C        2     if echoerror then error (err_echo_dis_int);
 18166:S
 18167:C        2   end; { await_locator }
 18168:S
 18169:D        1   procedure dgl_knob_init;
 18170:S
 18171:C        2   begin
 18172:C        2     with gcb^ do
 18173:C        3       begin
 18174:C        3         proc_await_locator := knob_await_locator;
 18175:C        3         proc_sample_locator := knob_sample_locator;
 18176:C        3         last_knob_rx := 0;
 18177:C        3         last_knob_ry := 0;
 18178:C        3         last_knob_dx := 0;
 18179:C        3         last_knob_dy := 0;
 18180:C        3       end;
 18181:C        2   end;
 18182:S
 18183:C        1   end. { dgl_knob }
 18184:S

No errors. 1 warrings.
            ***** Nonstandard language features enabled *****
```

DGL_POLY

Description

DGL_POLY contains the polygon routines except low-level code for device-dependent polygons.

Requirements

GLE_TYPES, SYSDEVS, SYSGLOBALS, GLE_HPGL_OUT, GLE_STEXT, GLE_ASTEXT, GLE_ASCLIP, GLE_SCLIP, GLE_SMARK, GLE_AUTL, GLE_UTLS, GLE_RAS_OUT, GLE_ARAS_OUT, GLE_FILE_IO, GLE_HPIB_IO, GENERAL_0, IODECLARATIONS, IOCOMASM, DGL_TOOLS, DGL_RASTER, DGL_TYPES, DGL_VARS, ASM, LOADER, GLE_GEN, DGL_GEN, DGL_HPGL, GLE_HPGL_IN, GLE_KNOB_IN, DGL_KNOB, GLE_GENI, and DGL_CONFG_OUT.

```
     1:D        0 {                                                                     }
     2:D        0 { Pascal work station graphics library                                }
     3:D        0 {                                                                     }
     4:D        0 { Module    = DGL_POLY                                                }
     5:D        0 { Programer = BJS                                                     }
     6:D        0 { Date      = 11/10/82                                                }
     7:S
     8:D        0 { Rev history:                                                        }
     9:S
    10:D        0 { Purpose: Hold polygon user routines                                 }
    11:S
    12:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    13:S                All rights are reserved.  Copying or other
    14:S                reproduction of this program except for archival
    15:S                purposes is prohibited without the prior
    16:S                written consent of Hewlett-Packard Company.
    17:S
    18:S
    19:S                            RESTRICTED RIGHTS LEGEND
    20:S
    21:S                Use, duplication, or disclosure by the Government
    22:S                is subject to restrictions as set forth in
    23:S                paragraph (b) (3) (B) of the Rights in Technical
    24:S                Data and Computer Software clause in
    25:S                DAR 7-104.9(a).
    26:S
    27:S                HEWLETT-PACKARD COMPANY
    28:D        0       Fort Collins, Colorado                          }
    29:S
    30:D        0 $modcal$
    30:D        0 $include 'OPTIONS'$
    31:S
    32:S           { This include file specifies range checking, debug and other compiler
    33:D        0    options for the graphics library                                   }
    34:S
    35:D        0 $debug OFF$
    36:D        0 $range OFF$
    37:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    38:D        0 $FLOAT_HDW TEST$
    39:S
    40:S
    41:S
    42:S
    43:D        0 $include 'OPTIONS'$
 20000:D        0 $linenum 20000$
 20001:D        0 $search 'TYPES',
 20002:D        0         'DGL_VARS',
 20003:D        0         'GEN',
 20004:D        0         'GLE_LIB',
 20005:D        0         'LIB'$
 20006:S
 20007:S
 20008:D        0 module DGL_POLY;
 20009:S
 20010:D        1 import dgl_types;
 20011:S
 20012:D        1 export
 20013:D        1   procedure set_pgn_ls ( index : integer);
 20014:D        1   procedure set_pgn_color ( index : integer);
 20015:D        1   procedure set_pgn_table ( index : integer;
```



```
 20016:D        2                           pdensity : real;
 20017:D        2                           porient : real;
 20018:D        2                           pedge   : integer);
 20019:S
 20020:D        1   procedure set_pgn_style ( index : integer );
 20021:D        1   procedure int_polygon_dd ( num_points : integer;
 20022:D        2                           anyvar xvec, yvec : gshortint_list;
 20023:D        2                           anyvar opcodes    : gshortint_list );
 20024:D        1   procedure polygon_dev_dep ( num_points : integer;
 20025:D        2                           anyvar xvec, yvec : greal_list;
 20026:D        2                           anyvar opcodes    : gshortint_list );
 20027:D        1   procedure int_polygon ( num_points : integer;  anyvar xvec,yvec : gshortint_list;
 20028:D        2                              anyvar opcodes   : gshortint_list );
 20029:D        1 procedure polygon ( num_points : integer;
 20030:D        2                           anyvar xvec, yvec : greal_list;
 20031:D        2                           anyvar opcodes    : gshortint_list );
 20032:D        1 implement
 20033:S
 20034:D        1 import dgl_vars,
 20035:D        1        dgl_gen,
 20036:D        1        gle_types,
 20037:D        1        gle_GEN,
 20038:D        1        dgl_lib,
 20039:D        1        asm;
 20040:S
 20041:D        1 const
 20042:D        1   deg_to_rad     = 0.01745329252;
 20043:D        1   normalized_one = 32768;
 20044:S
 20045:D        1 type vec_ptr_def  = ^gint_list;
 20046:D        1      work_ptr_def = ^gshortint_list;
 20047:S
 20048:D        1 var
 20049:D    -4  1   saved_linestyle : integer;
 20050:D    -8  1   saved_linewidth : integer;
 20051:D   -12  1   stack_ptr       : work_ptr_def;
 20052:S
 20053:D        1 procedure set_polygon_color;
 20054:S
 20055:D        2 { Purpose:  To set the color polygons will be drawn with (send to gle)       }
 20056:S
 20057:D        2 var
 20058:D    -1  2   pass_rgb : boolean;
 20059:D   -26  2   h,s,i   : real;
 20060:S
 20061:C        2 begin
 20062:C        2   with gcb^,gle_gcb^ do
 20063:C        3     begin
 20064:S               { If dither is to be used (b&w, moonunit, or 9836C with index > 15) then
 20065:C        3       RGB values need to be passed.  Otherwise the index is passed.     }
 20066:S
 20067:C        3       pass_rgb :=    (dither_support = 1) and
 20068:C        3                    ( ((color_map_support = 1) and
 20069:C        3                       (dgl_current_polygon_color > gamut)) or
 20070:C        3                     ( color_map_support = 0));
 20071:C        3       if pass_rgb then
 20072:C        4         begin
 20073:C        4           info1 := 0;
 20074:C        4           with color_table_ptr^[dgl_current_polygon_color] do
 20075:C        5             if gamut = 1 { black and white } then
```

```
20076:C        6               begin
20077:C        6                 { define only one parm for dither, use lit to set }
20078:C        6                 { calc brite as defined from CIE diagram          }
20079:C        6                 info2 := trunc((0.3*red+0.59*green+0.11*blue)*1023+0.5);
20080:C        6                 info3 := 0;
20081:C        6                 info4 := 0;
20082:C        6                 gle_fill_index_color(gle_gcb);
20083:C        6               end
20084:C        6             else          { multi color device }
20085:C        6               begin
20086:C        6                 info2 := trunc(red*1023+0.5);
20087:C        6                 info3 := trunc(green*1023+0.5);
20088:C        6                 info4 := trunc(blue*1023+0.5);
20089:C        6                 gle_fill_index_color(gle_gcb);
20090:C        6               end;
20091:C        4           end
20092:C        4         else
20093:C        4           begin
20094:C        4             info1 := 1;
20095:C        4             info2 := dgl_current_polygon_color;
20096:C        4             gle_fill_index_color(gle_gcb);
20097:C        4           end;
20098:C        3         dgl_polygon_color_current := true;
20099:C        3       end;
20100:C        2 end; { set_polygon_color }
20101:S
20102:D        1 procedure set_pgn_color (index : integer);
20103:S
20104:D        2 { Purpose:  To set the color polygons will be drawn with                }
20105:S
20106:D        2 var
20107:D     -1 2   pass_rgb : boolean;
20108:D    -26 2   h,s,I    : real;
20109:S
20110:C        2 begin
20111:C        2   ck_system_init;
20112:C        2   ck_display_init;
20113:S
20114:C        2   with gcb^,gle_gcb^ do
20115:C        3     begin                    { Bad values of index are set to index 1 }
20116:C        3       if  (index < 0) or
20117:C        4           ((index > gamut) and
20118:C        4           ((color_table_size = 0) or (index > color_table_size))) then index := 1;
20119:C        3       dgl_current_polygon_color := index;
20120:C        3       dgl_polygon_color_current := false;
20121:C        3     end;
20122:S
20123:C        2 end; { set_pgn_color }
20124:S
20125:S
20126:D        1 procedure set_pgn_ls ( index : integer);
20127:S
20128:D        2 { Purpose:  To set the linestyle that polygons are drawn with           }
20129:S
20130:C        2 begin
20131:C        2   ck_system_init;
20132:C        2   ck_display_init;
20133:S
20134:C        2   with gcb^ do
20135:C        3     begin
```



```
20136:C        3       if (index < 1) or (index > number_dgl_linestyles) then index := 1;
20137:C        3       dgl_current_polygon_linestyle := index;
20138:C        3     end;
20139:S
20140:C        2 end; { set_pgn_line_style }
20141:S
20142:D        1 procedure set_pgn_style ( index : integer );
20143:S
20144:D        2 { Purpose:  To set the polygon style that polygons will be drawn with }
20145:S
20146:C        2 begin
20147:C        2   ck_system_init;
20148:C        2   ck_display_init;
20149:S
20150:C        2   with gcb^ do
20151:C        3     begin
20152:C        3       if (index < 1) or (index > number_polygon_styles) then index := 1;
20153:C        3       with poly_table_ptr^ [ index ] do
20154:C        4         begin
20155:C        4           { decode table and setup local vars }
20156:C        4           dgl_current_polygon_crosshatch := density < 0;
20157:C        4           dgl_current_polygon_density := density;
20158:C        4           dgl_current_polygon_angle := orient;
20159:C        4           dgl_current_polygon_edge  := edge;
20160:C        4           dgl_current_polygon_style := index;
20161:C        4         end;
20162:C        3     end;
20163:C        2 end;
20164:S
20165:D        1 procedure set_pgn_table ( index : integer;
20166:D        2                           pdensity : real;
20167:D        2                           porient : real;
20168:D    -16 2                           pedge   : integer);
20169:S
20170:D    -16 2 { Purpose:  To define an entry in the polygon table  }
20171:S
20172:C        2 begin
20173:C        2   ck_system_init;
20174:C        2   ck_display_init;
20175:S
20176:C        2   with gcb^ do
20177:C        3     begin
20178:C        3       if ((index < 1) or (index > number_polygon_styles)) or
20179:C        4          ((pedge <> 0) and (pedge <> 1)) or
20180:C        4          ((pdensity < -1) or (pdensity > 1)) or
20181:C        4          ((porient < -90) or (porient > 90)) then error(err_bad_parms);
20182:C        3       with poly_table_ptr^ [ index ] do
20183:C        4         begin
20184:C        4           density := pdensity;
20185:C        4           orient := porient;
20186:C        4           edge := pedge = 1;
20187:C        4           if index = dgl_current_polygon_style then set_pgn_style(dgl_current_polygon_style);
20188:C        4         end;
20189:C        3     end;
20190:C        2 end;
20191:S
20192:D        1 procedure edge_polygon (        num_points         : integer;
20193:D        2                          anyvar vector             : gint_list;
20194:D        2                          anyvar opcodes            : gshortint_list;
20195:D        2                                 polygon_simulation : boolean );
```

```
20195:S
20197:D        2 { Purpose : To draw edges around the specified polygon }
20198:S
20199:D        2 var
20200:D    -4  2   vector_count    : integer;
20201:D    -8  2   next_subpolygon : integer;
20202:D   -12  2   i               : integer;
20203:D   -16  2   saved_color     : integer;
20204:D   -20  2   saved_linestyle : integer;
20205:D   -24  2   saved_linewidth : integer;
20206:S
20207:C        2 begin
20208:C        2   with gcb^, gle_gcb^ do
20209:C        3     begin
20210:C        3       if (polygon_simulation) and (dgl_current_polygon_density <> 0) then
20211:C        4         begin
20212:C        4           saved_color := dgl_current_color;
20213:C        4           saved_linestyle := dgl_current_linestyle;
20214:C        4           saved_linewidth := dgl_current_linewidth;
20215:C        4           set_color(dgl_current_polygon_color);
20216:C        4           set_line_style(dgl_current_polygon_linestyle);
20217:C        4           set_line_width(1);
20218:C        4         end;
20219:S
20220:C        3       vector_count    := 1;
20221:C        3       next_subpolygon := 1;
20222:S
20223:C        3       for i := 1 to num_points do
20224:C        4         begin
20225:C        4           if vector_count = next_subpolygon then
20226:C        5             begin
20227:C        5               end_x := vector[vector_count+1];
20228:C        5               end_y := vector[vector_count+2];
20229:C        5               gle_move ( gle_gcb );
20230:C        5               next_subpolygon := vector_count + vector[vector_count] * 2 + 1;
20231:C        5               vector_count := vector_count + 3;
20232:C        5             end
20233:C        5           else
20234:C        5             begin
20235:C        5               end_x := vector[vector_count];
20236:C        5               end_y := vector[vector_count+1];
20237:C        5               if opcodes[i] = 1 then gle_draw ( gle_gcb )
20238:C        6               else                   gle_move ( gle_gcb );
20239:C        5               vector_count := vector_count + 2;
20240:C        5             end;
20241:C        4         end;
20242:S
20243:C        3       if (polygon_simulation) and (dgl_current_polygon_density <> 0) then
20244:C        4         begin
20245:C        4           set_color(saved_color);
20246:C        4           set_line_style(saved_linestyle);
20247:C        4           set_line_width(saved_linewidth);
20248:C        4         end;
20249:C        3     end;
20250:C        2 end;
20251:S
20252:D        1 function int_div ( a, b : integer ) : integer;
20253:S
20254:D        2 { Purpose : To perform an integer div with rounding }
20255:S
```



```
20256:D    -4  2 var temp : integer;
20257:S
20258:C        2 begin
20259:C        2   temp := (2 * a ) div b;
20260:C        2   if odd ( temp ) then
20261:C        3     if temp > 0 then temp := temp + 1
20262:C        4     else             temp := temp - 1;
20263:C        2   int_div := temp div 2;
20264:C        2 end;
20265:S
20266:D        1 procedure line_line_intersection ( p1x, p1y, p2x, p2y,
20267:D        2                                    p3x, p3y, p4x, p4y : integer;
20268:D        2                                var ix,  iy            : integer );
20269:S
20270:D        2 { Purpose : To calculate the intersection of two lines }
20271:D        2 {           Note: The two lines must intersect.        }
20272:S
20273:D        2 var
20274:D        2   num, denom,
20275:D        2   delta_x_21,
20276:D        2   delta_y_21,
20277:D        2   delta_x_31,
20278:D        2   delta_y_31,
20279:D        2   delta_x_43,
20280:D  -32   2   delta_y_43  : integer;
20281:D  -32   2   real_num,
20282:D  -32   2   real_denom,
20283:D  -56   2   real_factor : real;
20284:S
20285:C        2 begin
20286:C        2 $range on$
20287:C        2   delta_x_21 := p2x - p1x;
20288:C        2   delta_y_21 := p2y - p1y;
20289:S
20290:C        2   delta_x_31 := p3x - p1x;
20291:C        2   delta_y_31 := p3y - p1y;
20292:S
20293:C        2   delta_x_43 := p4x - p3x;
20294:C        2   delta_y_43 := p4y - p3y;
20295:S
20296:C        2   try
20297:C        3     denom := delta_y_21 * delta_x_43 - delta_x_21 * delta_y_43;
20298:C        3     num   := delta_x_21 * delta_y_31 - delta_y_21 * delta_x_31;
20299:S
20300:C        3     ix := p3x + int_div((p4x-p3x)*num, denom);
20301:C        3     iy := p3y + int_div((p4y-p3y)*num, denom);
20302:C        3 $range off$
20303:S
20304:C        3   recover
20305:C        3     if escapecode = -4 ( integer overflow ) then
20306:C        4       begin
20307:C        4         real_denom := 1.0 * delta_y_21 * delta_x_43 - 1.0 * delta_x_21 * delta_y_43;
20308:C        4         real_num   := 1.0 * delta_x_21 * delta_y_31 - 1.0 * delta_y_21 * delta_x_31;
20309:C        4         real_factor := real_num / real_denom;
20310:C        4         ix := trunc(p3x + (p4x-p3x) * real_factor + 0.5);
20311:C        4         iy := trunc(p3y + (p4y-p3y) * real_factor + 0.5);
20312:C        4       end
20313:C        4     else
20314:C        4       escape(escapecode);
20315:C        2 end;
```

```
20316:S
20317:D       1 procedure draw_pg ( anyvar vector, work                  : gint_list;
20318:D       2                           dgl_current_polygon_color,
20319:D       2                           dgl_current_polygon_linestyle,
20320:D       2                           normalized_sin,normalized_cos  : integer;
20321:D       2                           dgl_current_polygon_crosshatch : boolean;
20322:D       2                           dgl_current_polygon_spacing    : integer);
20323:S
20324:D       2 { PURPOSE:  To draw a polygon using the current polygon attributes     }
20325:S
20326:S         {           The input format for vector is as follows (GLE polygon format):
20327:S
20328:S                       VECTOR [ Number of pts in segment 1 ( 1st subpolygon ) ]
20329:S                              [          X1                                   ]
20330:S                              [          Y1                                   ]
20331:S                              [          X2                                   ]
20332:S                              [          Y2                                   ]
20333:S                              [           :                                   ]
20334:S                              [           :                                   ]
20335:S                              [          Xn                                   ]
20336:S                              [          Yn                                   ]
20337:S
20338:S                              [ Number of pts in segment 2 ( 2nd subpolygon   ]
20339:S                              [          X1                                   ]
20340:S                              [          Y1                                   ]
20341:S                              [           :                                   ]
20342:S                              [           :                                   ]
20343:S                              [          Xm                                   ]
20344:S                              [          Ym                                   ]
20345:S
20346:S                                          :
20347:S                                          :
20348:S
20349:D       2                      [           0                                   ] }
20350:S
20351:S         {       The basic algorithm is as follow:
20352:S
20353:S                   - Calculate dist between fill lines
20354:S                   - Calculate fill line slope in terms of dx, dy
20355:S                   - For every edge in the polygon, calculate the x intercept
20356:S                     ( or y intercept for x major fill slope ) along a line parallel
20357:S                     to the fill lines for each end point.  With this information
20358:S                     build a record with minimum intercept, maximum intercept, and
20359:S                     both end points ordered by maximum intercept value.
20360:S
20361:S                     Maintain a minimum and maximum intercept value for all edge
20362:S                     end points in the polygon.  This information will be used to
20363:S                     indicate where to start filling the polygon with fill lines.
20364:S
20365:S                   - Calculate using the minimum intercept value the first fill
20366:S                     line that may intersect the polygon.
20367:S
20368:S                   - For each possible fill line do the following:
20369:S
20370:S                       - For each intercept record look for intersections.  An
20371:S                         intersection is determined by the current intercept value
20372:S                         of the current fill line, falling between the minimum
20373:S                         and maximum intercept values of the record.
20374:S
20375:S                         If an intersection is found, find the end point of
```



```
20376:S                           the intersection and save the point.
20377:S
20378:S                           After all intersections for a given fill line are found,
20379:S                           sort the end points.  The sort alternates between top
20380:S                           down, and bottom up for each fill line.  This minimizes
20381:S                           motion on mechanical devices.
20382:S
20383:D       2                   Plot the end points alternating between moves and draws. }
20384:S
20385:D       2 const
20386:D       2   normal_vertex = 0;
20387:D       2   short_vertex  = 1;
20388:D       2   edge_vertex   = 2;
20389:S
20390:D       2   edge_index    = 2;
20391:S
20392:D       2 type
20393:D       2   point_def1  = array [0..1] of integer;
20394:D       2   point_def   = array [0..2] of integer;
20395:S
20396:D       2   point_array = array [1..maxint] of point_def1;
20397:S
20398:D       2   intercept_rec_def = record
20399:D       2             intercept_p_min,
20400:D       2             intercept_p_max      : integer;
20401:D       2             intercept_min_points : point_def;
20402:D       2             intercept_max_points : point_def;
20403:D       2           end;
20404:S
20405:D       2   intercept_array = array [1..maxint] of intercept_rec_def;
20406:S
20407:D       2 var
20408:D    -4 2   intercept_list_ptr  : ^intercept_array;
20409:D    -8 2   p_list_ptr          : ^point_array;
20410:D   -12 2   p_count             : integer;
20411:D   -24 2   t, i, j             : integer;
20412:D   -28 2   intercept_count     : integer;
20413:D   -32 2   intercept_min       : integer;
20414:D   -36 2   intercept_max       : integer;
20415:D   -40 2   intercept_inc       : integer;
20416:D   -48 2   xmin, ymin          : integer;
20417:D   -56 2   dx, dy              : integer;
20418:D   -60 2   local_spacing       : integer;
20419:D   -64 2   vector_index        : integer;
20420:D   -68 2   num_vert            : integer;
20421:D   -72 2   last_index          : integer;
20422:D   -73 2   move_it             : boolean;
20423:D   -74 2   top_down_sort       : boolean;
20424:D   -75 2   x_major             : boolean;
20425:D   -76 2   hatch               : boolean;
20426:D   -80 2   saved_color         : integer;
20427:D   -84 2   vedge_index         : integer;
20428:D   -88 2   major_index         : integer;
20429:D   -92 2   first_index         : integer;
20430:D   -96 2   nxt_edge            : integer;
20431:D   -97 2   found_fill_line_on_edge : boolean;
20432:S
20433:D       2   procedure calc_intercept (  pt_1_index : integer;
20434:D       3                               pt_2_index : integer;
20435:D       3                               switch_xy  : boolean);
```

```
204 6:S
204 7:S             { Purpose : For each end point of the edge defined by pt_1_index and
204 8:S                         pt_2_index, calculate the intercept of a line which runs
204 9:D                         though the end point and is parallel with the fill line. }
204 0:S
204 1:D         3   var
204 2:D   -12   3     p1,p2,tp    : integer;
204 3:D   -20   3     pt1x,pt1y   : integer;
204 4:D   -28   3     pt2x,pt2y   : integer;
204 5:D   -36   3     ix,iy       : integer;
204 6:D   -44   3     tdx,tdy     : integer;
204 7:S
204 8:C         3   begin
204 9:C         3     if switch_xy then
204 0:C         4       begin
204 1:C         4         ix := 1;    iy := 0;
204 2:C         4         tdx := dy;  tdy := dx;
204 3:C         4       end
204 4:C         4     else
204 5:C         4       begin
204 6:C         4         ix := 0;    iy := 1;
204 7:C         4         tdx := dx;  tdy := dy;
204 8:C         4       end;
204 9:S
204 0:C         3     with intercept_list_ptr^[intercept_count] do
204 1:C         4       begin
204 2:C         4         pt1x := vector[pt_1_index+ix];
204 3:C         4         pt1y := vector[pt_1_index+iy];
204 4:C         4         pt2x := vector[pt_2_index+ix];
204 5:C         4         pt2y := vector[pt_2_index+iy];
204 6:S
204 7:C         4         p1 := pt1y - int_div(tdy * pt1x,tdx);       { calc intercept }
204 8:C         4         p2 := pt2y - int_div(tdy * pt2x,tdx);
204 9:S
204 0:C         4         if p1 > p2 then
204 1:C         5           begin                                     { swap points }
204 2:C         5             tp := p2;   p2  := p1;   p1  := tp;
204 3:C         5             tp := pt2x; pt2x := pt1x; pt1x := tp;
204 4:C         5             tp := pt2y; pt2y := pt1y; pt1y := tp;
204 5:C         5           end;
204 6:C         4                                                     { save intercepts }
204 7:C         4         intercept_count := intercept_count + 1;
204 8:C         4         intercept_p_min := p1;
204 9:C         4         intercept_p_max := p2;
204 0:C         4         intercept_max := max(intercept_max,intercept_p_max);
204 1:C         4         intercept_min := min(intercept_min,intercept_p_min);
204 2:C         4         if p1 = p2 then
204 3:C         5           begin
204 4:C         5             intercept_min_points[edge_index] := edge_vertex;
204 5:C         5             intercept_max_points[edge_index] := edge_vertex;
204 6:C         5             if pt1x <= pt2x then
204 7:C         6               begin
204 8:C         6                 intercept_min_points[ix] := pt1x;
204 9:C         6                 intercept_min_points[iy] := pt1y;
205 0:C         6                 intercept_max_points[ix] := pt2x;
205 1:C         6                 intercept_max_points[iy] := pt2y;
205 2:C         6               end
205 3:C         5             else
205 4:C         6               begin
205 5:C         6                 intercept_max_points[ix] := pt1x;
```



```
204 6:C         6                 intercept_max_points[iy] := pt1y;
204 7:C         6                 intercept_min_points[ix] := pt2x;
204 8:C         6                 intercept_min_points[iy] := pt2y;
204 9:C         6               end;
205 0:C         5           end
205 1:C         5         else
205 2:C         5           begin
205 3:C         5             intercept_min_points[edge_index] := normal_vertex;
205 4:C         5             intercept_max_points[edge_index] := normal_vertex;
205 5:C         5             intercept_min_points[ix] := pt1x;
205 6:C         5             intercept_min_points[iy] := pt1y;
205 7:C         5             intercept_max_points[ix] := pt2x;
205 8:C         5             intercept_max_points[iy] := pt2y;
205 9:C         5           end;
205 0:C         4       end;
205 1:C         3   end;
205 2:S
205 3:D         2   procedure calc_vertex_info (  edge_a : integer;
205 4:D         3                                 edge_b : integer);
205 5:S
205 6:D         3   { Purpose : To mark points which should not be used when calc fill line }
205 7:D         3   {           end points.                                                  }
205 8:S
205 9:D         3   var
205 0:D         3     a_min,
205 1:D         3     a_max,
205 2:D         3     b_min,
205 3:D   -16   3     b_max  : integer;
205 4:S
205 5:C         3   begin
205 6:C         3     with intercept_list_ptr^[edge_b] do
205 7:C         4       begin
205 8:C         4         b_min  := intercept_p_min;
205 9:C         4         b_max  := intercept_p_max;
205 0:C         4       end;
205 1:S
205 2:C         3     with intercept_list_ptr^[edge_a] do
205 3:C         4       begin
205 4:C         4         a_min := intercept_p_min;
205 5:C         4         a_max := intercept_p_max;
205 6:C         4         if (intercept_max_points[edge_index] <> edge_vertex) then
205 7:C         5           begin
205 8:C         5             if a_min = b_max then
205 9:C         6               intercept_min_points[edge_index] := short_vertex
206 0:C         6             else
206 1:C         6             if a_max = b_min then
206 2:C         7               intercept_max_points[edge_index] := short_vertex;
206 3:C         5           end;
206 4:C         4       end;
206 5:C         3   end;
206 6:S
206 7:D         2   procedure sort( starting, ending, inc : integer );
206 8:S
206 9:D         3   { Purpose : To sort the P_LIST array.  }
207 0:S
207 1:D         3   var
207 2:D    -2   3     sx          : gle_shortint;
207 3:D    -4   3     sy          : gle_shortint;
207 4:D    -6   3     index       : gle_shortint;
207 5:D   -10   3     test_point  : integer;
```

```
20556:D   -14  3     temp_point : integer;
20557:D   -15  3     done       : boolean;
20558:S
20559:C        3   begin
20560:C        3     if x_major then                ( sort by x )
20561:C        4       begin sx := 1; sy := 0; end
20562:C        4     else                           ( sort by y )
20563:C        4       begin sx := 0; sy := 1; end;
20564:S
20565:C        3     repeat
20566:C        4       index := starting + inc;
20567:C        4       done := true;
20568:C        4       test_point := p_list_ptr^[starting,sx];
20569:C        4       while index <> ending + inc do
20570:C        5         begin
20571:C        5           temp_point := p_list_ptr^[index,sx];
20572:C        5           if test_point > temp_point then
20573:C        6             begin
20574:C        6               p_list_ptr^[index,sx] := p_list_ptr^[index-inc,sx];
20575:C        6               p_list_ptr^[index-inc,sx] := temp_point;
20576:S
20577:C        6               temp_point := p_list_ptr^[index,sy];
20578:C        6               p_list_ptr^[index,sy] := p_list_ptr^[index-inc,sy];
20579:C        6               p_list_ptr^[index-inc,sy] := temp_point;
20580:S
20581:C        6               done := false;
20582:C        6             end
20583:C        6           else
20584:C        6             test_point := temp_point;
20585:C        5           index := index + inc;
20586:C        5         end;
20587:C        4     until done
20588:C        4   end;          ( sort )
20589:S
20590:C        2 begin ( poly )
20591:S
20592:C        2   with gcb^ do
20593:C        3     begin
20594:C        3       intercept_list_ptr := addr(work);
20595:C        3       hatch          := dgl_current_polygon_crosshatch;
20596:C        3       local_spacing  := dgl_current_polygon_spacing;
20597:S
20598:C        3       saved_color    := dgl_current_color;
20599:C        3       if dgl_current_color <> dgl_current_polygon_color then set_color(dgl_current_polygon_color);
20600:S
20601:C        3       with gle_gcb^ do
20602:C        4         repeat                       ( cross hatching loop )
20603:S
20604:C        5           hatch := not hatch;
20605:S
20606:C        5           { Calc slope in terms of dx, dy }
20607:C        5           { Calc x or y spacing  (intercept_inc)   }
20608:S
20609:C        5           x_major := true;
20610:C        5           major_index := 1;
20611:C        5           if abs(normalized_sin) = normalized_one  { 90 deg } then
20612:C        6             begin
20613:C        6               dy := display_max_y;
20614:C        6               dx := 0;
20615:C        6               intercept_inc := local_spacing;
```



```
20616:C        6             end
20617:C        6           else
20618:C        6           if abs(normalized_sin) <= abs(normalized_cos) { <= 45 deg } then
20619:C        7             begin
20620:C        7               dx := display_max_x;
20621:C        7               dy := int_div(dx * normalized_sin,normalized_cos);
20622:C        7               x_major := false;
20623:C        7               major_index := 0;
20624:C        7               intercept_inc := abs(int_div(local_spacing * normalized_one,normalized_cos));
20625:C        7             end
20626:C        7           else
20627:C        7             begin
20628:C        7               dy := display_max_y;
20629:C        7               dx := int_div(dy * normalized_cos,normalized_sin);
20630:C        7               intercept_inc := abs(int_div(local_spacing * normalized_one,normalized_sin));
20631:C        7             end;
20632:S
20633:C        5           if intercept_inc < 1 then intercept_inc := 1;
20634:S
20635:C        5           { Calc end point intercepts }
20636:S
20637:C        5           intercept_count := 1;
20638:C        5           intercept_min := maxint;
20639:C        5           intercept_max := minint;
20640:S
20641:C        5           vector_index := 1;
20642:C        5           while vector[vector_index] <> 0 do
20643:C        6             begin
20644:C        6               num_vert := vector[vector_index];
20645:C        6               vector_index := vector_index + 1;
20646:C        6               for i := 2 to num_vert do
20647:C        7                 calc_intercept ( vector_index+(i-1)*2,vector_index+(i-2)*2,x_major);
20648:C        6               last_index := vector_index+((num_vert-1)*2);
20649:C        6               calc_intercept ( vector_index,last_index,x_major);
20650:C        6               vector_index := last_index + 2;
20651:C        6             end;
20652:S
20653:C        5           intercept_count := intercept_count - 1;
20654:S
20655:C        5           vector_index := 1;
20656:C        5           vedge_index := 1;
20657:C        5           while vector[vector_index] <> 0 do
20658:C        6             begin
20659:C        6               num_vert := vector[vector_index];
20660:C        6               first_index := vedge_index;
20661:C        6               last_index := vedge_index + num_vert;
20662:C        6               for i := 1 to num_vert do
20663:C        7                 begin
20664:C        7                   nxt_edge := vedge_index + 1;
20665:C        7                   { The following while statement should read }
20666:C        7                   {  'while (nxt_edge < last_index) and ...'  }
20667:C        7                   { however this "bug" was not found until after }
20668:C        7                   { QA.  It will not proceduce a user bug though }
20669:C        7                   { since the following 'if' stmt with not use the }
20670:C        7                   { bad results                                    }
20671:C        7                   while (nxt_edge < last_index) and
20672:C        8                         (intercept_list_ptr^[nxt_edge].
20673:C        8                          intercept_max_points[edge_index] = edge_vertex) do
20674:C        8                     nxt_edge := nxt_edge + 1;
20675:C        7                   if nxt_edge >= last_index then
```

```
20678:C       8                    begin
20677:C       8                      nxt_edge := first_index;
20678:C       8                        while (nxt_edge < vedge_index) and
20679:C       9                              (intercept_list_ptr^[nxt_edge].
20680:C       9                               intercept_max_points[edge_index] = edge_vertex) do
20681:C       9                          nxt_edge := nxt_edge + 1;
20682:C       8                    end;
20683:C       7                  if (intercept_list_ptr^[nxt_edge].
20684:C       8                      intercept_max_points[edge_index] <> edge_vertex) then
20685:C       8                        calc_vertex_info ( vedge_index,nxt_edge);
20686:C       7                  vedge_index := vedge_index + 1;
20687:C       7                end;
20688:C       6              vector_index := vector_index + num_vert * 2 + 1;
20689:C       6            end;
20690:S
20691:S                  { Calc first fill line intercept value, adjust with lower left
20692:C       5            of display                                                  }
20693:S
20694:C       5          p_list_ptr := addr(work,(intercept_count+1)*32);
20695:S
20696:C       5          intercept_min := intercept_min - (intercept_min mod intercept_inc);
20697:C       5          xmin := 0;
20698:C       5          ymin := 0;
20699:C       5          top_down_sort := true;
20700:S
20701:C       5          { Fill polygon }
20702:C       5          while intercept_min <= intercept_max do
20703:C       6            begin
20704:C       6              if x_major then
20705:C       7                xmin := intercept_min
20706:C       7              else
20707:C       7                ymin := intercept_min;
20708:S
20709:C       6              { Find intersections }
20710:C       6              p_count := 0;
20711:C       6              found_fill_line_on_edge := false;
20712:C       6              for i := 1 to intercept_count do
20713:C       7                begin
20714:C       7                  with intercept_list_ptr^[i] do
20715:C       8                    begin
20716:C       8                      if (intercept_min = intercept_p_min) and
20717:C       9                         (intercept_min = intercept_p_max) then
20718:C       9                        found_fill_line_on_edge := true
20719:C       9                      else
20720:C       9                      if (intercept_min >= intercept_p_min) and
20721:C      10                         (intercept_min <= intercept_p_max) then
20722:C      10                        { intersection }
20723:C      10                        begin
20724:C      10                          if (intercept_min = intercept_p_min) then
20725:C      11                            begin
20726:C      11                             if (intercept_min_points[edge_index] = normal_vertex) then
20727:C      12                               begin
20728:C      12                                 p_count := p_count + 1;
20729:C      12                                 p_list_ptr^[p_count,0] := intercept_min_points[0];
20730:C      12                                 p_list_ptr^[p_count,1] := intercept_min_points[1];
20731:C      12                               end;
20732:C      11                            end
20733:C      11                          else
20734:C      11                          if (intercept_min = intercept_p_max) then
20735:C      12                            begin
```



```
20736:C      12                             if (intercept_max_points[edge_index] = normal_vertex) then
20737:C      13                               begin
20738:C      13                                 p_count := p_count + 1;
20739:C      13                                 p_list_ptr^[p_count,0] := intercept_max_points[0];
20740:C      13                                 p_list_ptr^[p_count,1] := intercept_max_points[1];
20741:C      13                               end;
20742:C      12                            end
20743:C      12                          else
20744:C      12                            begin
20745:C      12                              p_count := p_count + 1;
20746:C      12                              line_line_intersection(xmin,ymin,xmin+dx,ymin+dy,
20747:C      12                                intercept_min_points[0],intercept_min_points[1],
20748:C      12                                intercept_max_points[0],intercept_max_points[1],
20749:C      12                                p_list_ptr^[p_count,0],p_list_ptr^[p_count,1]);
20750:C      12                            end;
20751:C      10                        end;
20752:C       8                    end;
20753:C       7                end;
20754:S
20755:C       6              if found_fill_line_on_edge then  { add edge points }
20756:C       7                begin
20757:C       7                  if p_count > 1 then sort(1,p_count,1); { sort bottom up }
20758:C       7                  if odd(p_count) then p_count := p_count-1; { remove last move }
20759:C       7                  for i := 1 to intercept_count do
20760:C       8                    with intercept_list_ptr^[i] do
20761:C       9                      begin
20762:C       9                        if (intercept_min = intercept_p_min) and
20763:C      10                          (intercept_min = intercept_p_max) then
20764:C      10                        begin
20765:C      10                          p_count := p_count + 1;
20766:C      10                          p_list_ptr^[p_count,0] := intercept_min_points[0];
20767:C      10                          p_list_ptr^[p_count,1] := intercept_min_points[1];
20768:C      10                          p_count := p_count + 1;
20769:C      10                          p_list_ptr^[p_count,0] := intercept_max_points[0];
20770:C      10                          p_list_ptr^[p_count,1] := intercept_max_points[1];
20771:C      10                        end;
20772:C       9                    end;
20773:C       7                  i := 1;
20774:C       7                  repeat
20775:C       8                    j := i + 2;
20776:C       8                    while j < p_count do
20777:C       9                      begin
20778:C       9                        if (p_list_ptr^[i,major_index] <=
20779:C      10                           p_list_ptr^[j+1,major_index]) and
20780:C      10                          (p_list_ptr^[j,major_index] <=
20781:C      10                           p_list_ptr^[i+1,major_index]) then
20782:C      10                        begin
20783:C      10                          if p_list_ptr^[i,major_index] >
20784:C      11                            p_list_ptr^[j,major_index] then
20785:C      11                            begin
20786:C      11                              p_list_ptr^[i,0] := p_list_ptr^[j,0];
20787:C      11                              p_list_ptr^[i,1] := p_list_ptr^[j,1];
20788:C      11                            end;
20789:C      10                          if p_list_ptr^[i+1,major_index] <
20790:C      11                            p_list_ptr^[j+1,major_index] then
20791:C      11                            begin
20792:C      11                              p_list_ptr^[i+1,0] := p_list_ptr^[j+1,0];
20793:C      11                              p_list_ptr^[i+1,1] := p_list_ptr^[j+1,1];
20794:C      11                            end;
20795:C      10                          for t := j to p_count-2 do
```

```
20796:C       11                                begin
20797:C       11                                  p_list_ptr^[t,0] := p_list_ptr^[t+2,0];
20798:C       11                                  p_list_ptr^[t,1] := p_list_ptr^[t+2,1];
20799:C       11                                end;
20800:C       10                              p_count := p_count - 2;
20801:C       10                              j := i + 2;
20802:C       10                            end
20803:C       10                          else
20804:C       10                            j := j + 2;
20805:C        9                        end;
20806:C        8                      i := i + 2;
20807:C        8                    until i > p_count;
20808:C        7                  end;
20809:S
20810:C        6                { Sort points }
20811:C        6                if p_count > 1 then
20812:C        7                  begin
20813:C        7                    top_down_sort := not top_down_sort;
20814:C        7                    if top_down_sort then sort(p_count,1,-1)
20815:C        8                    else sort(1,p_count,1);
20816:C        7                  end;
20817:S
20818:C        6                { draw a fill line }
20819:C        6                move_it := true;
20820:C        6                for i := 1 to p_count do
20821:C        7                  begin
20822:C        7                    end_x := p_list_ptr^[i,0];
20823:C        7                    end_y := p_list_ptr^[i,1];
20824:C        7                    if move_it then call ( move,gle_gcb )
20825:C        8                    else            call ( draw,gle_gcb );
20826:C        7                    move_it := not move_it;
20827:C        7                  end;
20828:S
20829:C        6                intercept_min := intercept_min + intercept_inc;
20830:C        6              end;  { of filling loop }
20831:S
20832:C        5            if (not hatch) and dgl_current_polygon_crosshatch then
20833:C        6              begin
20834:C        6                t := normalized_sin;
20835:C        6                normalized_sin := -normalized_cos;
20836:C        6                normalized_cos := t;
20837:C        6              end;
20838:C        5          until hatch; { end of hatching loop }
20839:S
20840:C        3        if saved_color <> dgl_current_polygon_color then set_color(saved_color);
20841:C        3      end;
20842:C        2 end;
20843:S
20844:D        1 procedure draw_polygon ( anyvar vector, work : gint_list );
20845:S
20846:D        2 { PURPOSE:  To draw a polygon using the current polygon attributes     }
20847:S
20848:S          {           The input format for vector is as follows (GLE polygon format):
20849:S
20850:S                        VECTOR [ Number of pts in segment 1 ( 1st subpolygon ) ]
20851:S                               [          X1                                   ]
20852:S                               [          Y1                                   ]
20853:S                               [          X2                                   ]
20854:S                               [          Y2                                   ]
20855:S                               [           :                                   ]
```



```
20856:S                               [           :                                   ]
20857:S                               [          Xn                                   ]
20858:S                               [          Yn                                   ]
20859:S
20860:S                               [ Number of pts in segment 2 ( 2nd subpolygon   ]
20861:S                               [          X1                                   ]
20862:S                               [          Y1                                   ]
20863:S                               [           :                                   ]
20864:S                               [           :                                   ]
20865:S                               [          Xm                                   ]
20866:S                               [          Ym                                   ]
20867:S
20868:S                                :
20869:S                                :
20870:S
20871:D        2                      [           0                                   ] }
20872:S
20873:D        2   { normalized_one = 32768 }
20874:S
20875:D        2 var
20876:D    -8  2   local_angle   : real;
20877:D    -9  2   hatch         : boolean;
20878:D   -14  2   local_spacing : integer;
20879:D   -22  2   rad_angle     : real;
20880:D   -22  2   sin_angle,
20881:D   -30  2   cos_angle     : integer         { normalized fixed point numbers };
20882:S
20883:C        2 begin { draw_polygon }
20884:S
20885:C        2   with gcb^ do
20886:C        3     begin
20887:C        3       if dgl_current_polygon_density <> 0 then
20888:C        4         begin
20889:C        4           local_angle := dgl_current_polygon_angle;
20890:C        4           hatch := dgl_current_polygon_crosshatch and (dgl_current_polygon_density <> 1);
20891:C        4           if dgl_current_polygon_density = 1 then
20892:C        5             local_spacing := gle_gcb^.polygon_solid_fill
20893:C        5           else
20894:C        5             local_spacing :=
20895:C        5               abs(trunc(1/dgl_current_polygon_density * gle_gcb^.polygon_fill_factor));
20896:C        4           if local_spacing < 1 then local_spacing := 1;
20897:C        4           if hatch then local_spacing := local_spacing * 2;
20898:C        4           if local_spacing = 1 then local_angle := 0;
20899:C        4           rad_angle := deg_to_rad * local_angle;
20900:C        4           sin_angle := trunc(sin(rad_angle) * normalized_one);
20901:C        4           cos_angle := trunc(cos(rad_angle) * normalized_one);
20902:C        4           draw_pg(vector,work,dgl_current_polygon_color,dgl_current_polygon_linestyle,
20903:C        4                   sin_angle,cos_angle,hatch,local_spacing);
20904:C        4         end;
20905:C        3     end;
20906:C        2 end;
20907:S
20908:D        1 procedure setup_for_polygon ( real_format  : boolean;
20909:D        2                               num_points   : integer;
20910:D        2                         anyvar xvec, yvec  : gshortint_list;
20911:D        2                         anyvar rxvec, ryvec : greal_list;
20912:D        2                         anyvar opcodes     : gshortint_list;
20913:D        2                         anyvar vector      : vec_ptr_def;
20914:D        2                                work_mult   : integer;
20915:D        2                            anyvar work_ptr  : work_ptr_def;
```

```
                                var last_subpoly : integer );

{ Purpose : To prepare for drawing a polygon set.  This includes          }
{           setting up attributes, creating work space, performing error  }
{           checks, and creating a GLE format polygon.                    }

var
  i              : integer;
  sub_poly_start : integer;
  sub_poly_count : integer;
  point_count    : integer;
  vector_count   : integer;
  local_angle    : real;
  rad_angle      : real;

begin
  ck_system_init;
  ck_display_init;
$ovflcheck on$
  if num_points <= 0 then error (err_neg_points);
  if opcodes[1] <> 2 then error (err_bad_parms);

  { allocate worst possible space for vector array }
  mark(stack_ptr);                          { mark current base }
  newbytes(vector,12*num_points+4);         { alocate worst case space }

  with gcb^ do
    begin
      saved_linestyle := dgl_current_linestyle;
      saved_linewidth := dgl_current_linewidth;

      if dgl_current_linestyle <> dgl_current_polygon_linestyle then
        set_line_style ( dgl_current_polygon_linestyle );
      if dgl_current_linewidth <> 1 then set_line_width(1);
      if not dgl_polygon_color_current then set_polygon_color;
    end;

  sub_poly_start := 1;
  i := 1;
  sub_poly_count := 1;
  point_count := 1;
  last_subpoly := 1;   { last subpolygon in polygon }
  vector_count := 2;      { first spot will hold count }

  while i <= num_points do
    begin
      if (opcodes[I]=2) and (i<>1)  then
        begin
          vector^[sub_poly_start] := point_count-1;
          sub_poly_start := vector_count;  { save space to hold count }
          vector_count := vector_count + 1;
          point_count := 1;
          sub_poly_count := sub_poly_count + 1;
          last_subpoly := i;
        end;
      point_count := point_count + 1;
      if real_format then
        convert_wtod(rxvec[i],ryvec[i],vector^[vector_count],vector^[vector_count+1])
      else
      if short_flag then
```



```
        convert_intwtod(xvec[i],yvec[i],vector^[vector_count],vector^[vector_count+1])
      else
        convert_wtod(xvec[i],yvec[i],vector^[vector_count],vector^[vector_count+1]);
      vector_count := vector_count + 2;
      i := i + 1;
    end;
  vector^[sub_poly_start] := point_count-1;
  vector^[vector_count] := 0;

  newbytes(work_ptr,work_mult*vector_count);     { alocate work space for gle }

  with gcb^,gle_gcb^ do
    begin
      if dgl_current_polygon_crosshatch then info1 := 1
      else                                   info1 := 0;

      local_angle := dgl_current_polygon_angle;
      rad_angle := deg_to_rad * local_angle;
      info3 := trunc(sin(rad_angle) * normalized_one);
      info4 := trunc(cos(rad_angle) * normalized_one);

      if dgl_current_polygon_density = 1 then
        info2 := polygon_solid_fill
      else
      if dgl_current_polygon_density = 0 then
        info2 := 0
      else
        begin
          info2 := abs(trunc(1/dgl_current_polygon_density * polygon_fill_factor));
          if info2 < 1 then info2 := 1;
        end;
    end;
end;


procedure finish_polygon;

{ Purpose: Restore linestyle and line width to current values }

begin
  with gcb^ do
    begin
      if saved_linestyle <> dgl_current_polygon_linestyle then
        set_line_style(saved_linestyle);
      if saved_linewidth <> 1 then
        set_line_width(saved_linewidth);
    end;
end;

procedure int_polygon_dd ( num_points : integer;
                           anyvar xvec, yvec : gshortint_list;
                           anyvar opcodes    : gshortint_list );

{ Purpose : To output a device dependent polygon set }

var
  t              : array[1..1] of real;
  work_ptr       : work_ptr_def;
  vector_ptr     : vec_ptr_def;
  last_subpolygon : integer;
```

```
21036:D   -21  2   use_simulation  : boolean;
21037:S
21038:C        2 begin
21039:C        2   try { must return 'new' space if escape occurs }
21040:C        3     setup_for_polygon ( false, num_points, xvec, yvec, t, t,
21041:C        3                         opcodes, vector_ptr,14,work_ptr,last_subpolygon);
21042:S
21043:C        3     with gle_gcb^ do
21044:C        4       begin
21045:C        4         use_simulation := true;
21046:C        4         if polygon_support = 1 then
21047:C        5           begin
21048:C        5             gle_get_polygon_info ( gle_gcb );
21049:C        5             if error_return = 0 then
21050:C        6               begin
21051:C        6                 info_ptr1 := vector_ptr;
21052:C        6                 info_ptr2 := work_ptr;
21053:C        6                 gle_polygon ( gle_gcb );
21054:C        6                 use_simulation := false;
21055:C        6               end;
21056:C        5           end;
21057:C        4       end;
21058:C        3     finish_polygon;
21059:C        3     if gcb^.dgl_current_polygon_edge or (use_simulation) then
21060:C        4       edge_polygon ( num_points, vector_ptr^[1], opcodes,use_simulation);
21061:C        3     int_move(xvec[last_subpolygon],yvec[last_subpolygon]); { update cp }
21062:S
21063:C        3     release(stack_ptr);                    { return all space }
21064:C        3   recover
21065:C        3     begin
21066:C        3       if escapecode <> -27 then release (stack_ptr);
21067:C        3       escape(escapecode);
21068:C        3     end;
21069:C        2 end;
21070:S
21071:D        1 procedure polygon_dev_dep ( num_points : integer;
21072:D        2                             anyvar xvec, yvec : greal_list;
21073:D        2                             anyvar opcodes    : gshortint_list );
21074:S
21075:D        2 { Purpose : To output a device dependent polygon set }
21076:S
21077:D        2 var
21078:D    -8  2   t              : array[1..1] of real;
21079:D   -12  2   work_ptr       : work_ptr_def;
21080:D   -16  2   vector_ptr     : vec_ptr_def;
21081:D   -20  2   last_subpolygon : integer;
21082:D   -21  2   use_simulation  : boolean;
21083:S
21084:C        2 begin
21085:C        2   try { must return 'new' space if escape occurs }
21086:C        3     setup_for_polygon ( true, num_points, t, t, xvec, yvec,
21087:C        3                         opcodes, vector_ptr,14,work_ptr,last_subpolygon);
21088:C        3     with gle_gcb^ do
21089:C        4       begin
21090:C        4         use_simulation := true;
21091:C        4         if polygon_support = 1 then
21092:C        5           begin
21093:C        5             gle_get_polygon_info ( gle_gcb );
21094:C        5             if error_return = 0 then
21095:C        6               begin
```



```
21096:C        6                 info_ptr1 := vector_ptr;
21097:C        6                 info_ptr2 := work_ptr;
21098:C        6                 gle_polygon ( gle_gcb );
21099:C        6                 use_simulation := false;
21100:C        6               end;
21101:C        5           end;
21102:C        4       end;
21103:C        3     finish_polygon;
21104:C        3     if gcb^.dgl_current_polygon_edge or (use_simulation) then
21105:C        4       edge_polygon ( num_points, vector_ptr^[1], opcodes, use_simulation);
21106:C        3     move(xvec[last_subpolygon],yvec[last_subpolygon]); { update cp }
21107:S
21108:C        3     release(stack_ptr);                    { return all space }
21109:C        3   recover
21110:C        3     begin
21111:C        3       if escapecode <> -27 then release (stack_ptr);
21112:C        3       escape(escapecode);
21113:C        3     end;
21114:C        2 end;
21115:S
21116:D        1 procedure int_polygon ( num_points : integer;  anyvar xvec,yvec : gshortint_list;
21117:D        2                         anyvar opcodes   : gshortint_list );
21118:S
21119:S
21120:D        2 { Purpose : To output a device independent polygon set }
21121:S
21122:D        2 var
21123:D    -8  2   t          : array[1..1] of real;
21124:D   -12  2   work_ptr   : work_ptr_def;
21125:D   -16  2   vector_ptr : vec_ptr_def;
21126:D   -20  2   last_subpolygon : integer;
21127:S
21128:C        2 begin
21129:C        2   try { must return 'new' space if escape occurs }
21130:C        3     setup_for_polygon ( false, num_points, xvec, yvec, t, t,
21131:C        3                         opcodes, vector_ptr, 40, work_ptr,last_subpolygon );
21132:S
21133:C        3     draw_polygon ( vector_ptr^[1], work_ptr^[1] );
21134:C        3     finish_polygon;
21135:C        3     if gcb^.dgl_current_polygon_edge then
21136:C        4       edge_polygon ( num_points, vector_ptr^[1], opcodes, false );
21137:C        3     int_move(xvec[last_subpolygon],yvec[last_subpolygon]); { update cp }
21138:S
21139:C        3     release(stack_ptr);                    { return all space }
21140:C        3   recover
21141:C        3     begin
21142:C        3       if escapecode <> -27 then release (stack_ptr);
21143:C        3       escape(escapecode);
21144:C        3     end;
21145:C        2 end;
21146:S
21147:D        1 procedure polygon ( num_points : integer;
21148:D        2                     anyvar xvec, yvec : greal_list;
21149:D        2                     anyvar opcodes    : gshortint_list );
21150:S
21151:D        2 { Purpose : To output a device independent polygon set }
21152:S
21153:D        2 var
21154:D    -8  2   t          : array[1..1] of real;
21155:D   -12  2   work_ptr   : work_ptr_def;
```

```
21156:D   -16  2   vector_ptr : vec_ptr_def;
21157:D   -20  2   last_subpolygon : integer;
21158:S
21159:C        2 begin
21160:C        2   try { must return 'new' space if escape occurs }
21161:C        3     setup_for_polygon ( true, num_points, t, t, xvec, yvec,
21162:C        3                         opcodes, vector_ptr,40, work_ptr,last_subpolygon );
21163:S
21164:C        3     draw_polygon ( vector_ptr^[1], work_ptr^[1] );
21165:C        3     finish_polygon;
21166:C        3     if gcb^.dgl_current_polygon_edge then
21167:C        4       edge_polygon ( num_points, vector_ptr^[1], opcodes, false );
21168:C        3     move(xvec[last_subpolygon],yvec[last_subpolygon]); { update cp }
21169:S
21170:C        3     release(stack_ptr);                    { return all space }
21171:C        3   recover
21172:C        3     begin
21173:C        3       if escapecode <> -27 then release (stack_ptr);
21174:C        3       escape(escapecode);
21175:C        3     end;
21176:C        2 end;
21177:S
21178:C        1 end. {module DGL_POLY}
21179:S
21180:D        1
21181:S
```

No errors. No warnings.
***** Nonstandard language features enabled *****

DGL_RAS

Description

DGL_RAS provides device-dependent initialization for raster devices, as well as raster utilities.

Requirements

DGL_TYPES, DGL_VARS, SYSDEVS, SYSGLOBALS, ASM, LOADERS, GLE_GEN, GLE_TYPES, GLE_GEN, GLE_AUTL, GLE_RAS_OUT, and DGL_GEN.

```
    1:D        0 {                                                                          }
    2:D        0 { DGL device dependent init routine                                        }
    3:D        0 {                                                                          }
    4:D        0 { Module    = DGL_RASTER                                                   }
    5:D        0 { Programer = BJS                                                          }
    6:D        0 { Date      = 1 - 5-83                                                     }
    7:D        0 {                                                                          }
    8:D        0 { Purpose: To provide device dependent initialization for raster devices.  }
    9:S
   10:D        0 { Rev history                                                              }
   11:D        0 {  Created  - 1 - 5-83  BJS                                                }
   12:D        0 {  Modified - 02-17-84  BDS Changed allocations from dynamic to global.    }
   13:S
   14:S          {     (c) Copyright Hewlett-Packard Company, 1983.
   15:S           All rights are reserved.  Copying or other
   16:S           reproduction of this program except for archival
   17:S           purposes is prohibited without the prior
   18:S           written consent of Hewlett-Packard Company.
   19:S
   20:S
   21:S                        RESTRICTED RIGHTS LEGEND
   22:S
   23:S           Use, duplication, or disclosure by the Government
   24:S           is subject to restrictions as set forth in
   25:S           paragraph (b) (3) (B) of the Rights in Technical
   26:S           Data and Computer Software clause in
   27:S           DAR 7-104.9(a).
   28:S
   29:S           HEWLETT-PACKARD COMPANY
   30:D        0  Fort Collins, Colorado                                  }
   31:S
   32:D        0 $search 'GLE_LIB',
   33:D        0         'TYPES',
   34:D        0         'DGL_VARS',
   35:D        0         'GEN'$
   36:D        0 $modcal$
   36:D        0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS ****************** }
   37:S
   38:S          { This include file specifies range checking, debug and other compiler
   39:D        0   options for the graphics library                                        }
   40:S
   41:D        0 $debug OFF$
   42:D        0 $range OFF$
   43:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
   44:D        0 $FLOAT_HDW TEST$
   45:S
   46:S
   47:S
   48:S
   49:D        0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS ****************** }
17000:D        0 $linenum 17000$
17001:S
17002:D        0 module DGL_RASTER;
17003:S
17004:D        1 export
17005:S
17006:D        1 procedure dgl_raster_init ( control : integer );
17007:S
17008:D        1 implement
17009:S
```



```
17010:D        1 import dgl_types,
17011:D        1        dgl_vars,
17012:D        1        sysdevs,
17013:D        1        asm,
17014:D        1        sysglobals,
17015:D        1        gle_types,
17016:D        1        gle_gen,
17017:D        1        gle_autl,
17018:D        1        gle_ras_out,
17019:D        1        dgl_gen;
17020:S
17021:D        1 type
17022:D        1   init_color_table_def = ARRAY [0..15] of c_def;
17023:S
17024:D        1 const
17025:D        1   init_color_table = init_color_table_def [
17026:D        1 c_def [ red :  0, green :  0,  blue :  0 ],  {  0 }
17027:D        1 c_def [ red :  1, green :  1,  blue :  1 ],  {  1 }
17028:D        1 c_def [ red :  1, green :  0,  blue :  0 ],  {  2 }
17029:D        1 c_def [ red :  1, green :  1,  blue :  0 ],  {  3 }
17030:D        1 c_def [ red :  0, green :  1,  blue :  0 ],  {  4 }
17031:D        1 c_def [ red :  0, green :  1,  blue :  1 ],  {  5 }
17032:D        1 c_def [ red :  0, green :  0,  blue :  1 ],  {  6 }
17033:D        1 c_def [ red :  1, green :  0,  blue :  1 ],  {  7 }
17034:D        1 c_def [ red :  0, green :  0,  blue :  0 ]]; {  8 }
17035:D        1 c_def[red:0.8               ,green:0.733333333333333,blue:0.2               ], {  9 }
17036:D        1 c_def[red:0.2               ,green:0.666666666666667,blue:0.466666666666667 ], { 10 }
17037:D        1 c_def[red:0.533333333333333,green:0.4               ,blue:0.666666666666667 ], { 11 }
17038:D        1 c_def[red:0.8               ,green:0.266666666666667,blue:0.4               ], { 12 }
17039:D        1 c_def[red:1.0               ,green:0.4               ,blue:0.2               ], { 13 }
17040:D        1 c_def[red:1.0               ,green:0.466666666666667,blue:0                 ], { 14 }
17041:D        1 c_def[red:0.866666666666667,green:0.533333333333333,blue:0.266666666666667 ]];{ 15 }
17042:S
17043:D        1 {procedure hpm_new(var opject:anyptr; numbytes : integer); external;}
17044:S
17045:D        1 procedure raster_linestyle ( index : integer);
17046:S
17047:D        2 { Purpose:  To set the linestyle that primitives are drawn with            }
17048:S
17049:D        2 type
17050:D        2   ls_map_def = packed array [1..13] of gbyte;
17051:D        2 const
17052:D        2   ls_map = ls_map_def [0,2,3,4,5,6,1,2,3,4,5,6,1];
17053:S
17054:C        2 begin
17055:C        2   with gle_gcb^ do
17056:C        3     begin
17057:C        3       info1 := ls_map[index];                 { map DGL to GLE def }
17058:C        3       info2 := 4;                             { repeat rate 4% }
17059:C        3       info3 := 0;                             { linestyle mode }
17060:C        3       info4 := raster_patterns[index-1];      { pattern }
17061:C        3       gle_linestyle ( gle_gcb );
17062:C        3     end;
17063:S
17064:S
17065:C        2 end; { set_line_style }
17066:S
17067:D        1 function return_closest_color ( r,g,b : real;          { target color }
17068:D   -24   2                                  c_table_ptr : anyptr ) { system color map } : integer;
17069:S
```

```
17070:D   -24  2 var
17071:D   -24  2   target_h,
17072:D   -24  2   target_s,
17073:D   -48  2   target_l : real;          { HSL target color values }
17074:D   -48  2   error_h,
17075:D   -48  2   error_h2,                 { hue error distance squared }
17076:D   -48  2   error_l,                  { lightness error distance squared }
17077:D   -48  2   error_s,
17078:D   -48  2   error_l2,
17079:D   -96  2   error_s2 : real;          { saturation error distance squared }
17080:D   -96  2   map_h,
17081:D   -96  2   map_l,
17082:D   -96  2   map_s,                    { Current color map entry in HSL }
17083:D  -128  2   error : real;             { Distance from target to current color map entry }
17084:D  -136  2   closest_error : real;
17085:D  -136  2   i,
17086:D  -144  2   closest_index : integer;  { Best fit color map index }
17087:S
17088:D  -144  2 { Find closest color from system color map, to match target color.  }
17089:D  -144  2 { The closest color is the color which 'looks' the closest.  This   }
17090:D  -144  2 { algorithm has been derived from a mixture of logic and            }
17091:D  -144  2 { experimentation.  The algorithm calculates for each entry in the  }
17092:D  -144  2 { color map an error factor indicating how far off the color map    }
17093:D  -144  2 { value is from the target color.  It then returns the color map   }
17094:D  -144  2 { index with the least error.                                       }
17095:S
17096:D  -144  2 { Experimentation has showed that the best looking color is normaly }
17097:D  -144  2 { the color with the least error in hue.  However when the target   }
17098:D  -144  2 { color is near black or white this is not true, and when many color}
17099:D  -144  2 { map entrys have a small hue error the closest hue does not produce}
17100:D  -144  2 { the best color.  The algorithm makes special cases out of the     }
17101:D  -144  2 { above cases and 'weights' the error result to reduce the effects  }
17102:D  -144  2 { of hue.                                                           }
17103:S
17104:C        2 begin
17105:C        2   convert_rgb_to_hsl(r,g,b,target_h,target_s,target_l);
17106:C        2   closest_index := 1;
17107:C        2   closest_error := maxint;                          { worst case error }
17108:C        2   for i := 0 to gle_gcb^.gamut do                    { for each CMAP entry }
17109:C        3     with color_table_ptr_def(c_table_ptr)^[i] do     { force anyptr to known type }
17110:C        4       begin
17111:C        4         convert_rgb_to_hsl(red,green,blue,map_h,map_s,map_l);
17112:S
17113:C        4         { Calc errors, note that since Hue is circular it must }
17114:C        4         { be calc as shortest dist of either direction         }
17115:S
17116:C        4         error_h  := abs(map_h-target_h);
17117:C        4         error_h2 := abs(map_h-1-target_h);
17118:C        4         if error_h2 <= error_h then error_h := error_h2;
17119:C        4         error_h2 := error_h * error_h;
17120:C        4         error_s := abs(map_s-target_s);
17121:C        4         error_l := abs(map_l-target_l);
17122:C        4         error_s2 := error_s * error_s;
17123:C        4         error_l2 := error_l * error_l;
17124:S
17125:C        4         if target_l < 0.1 then { special case where request is near black }
17126:C        5           begin
17127:C        5             { With small lum in cmap, sat and hue are undifined and can't }
17128:C        5             { be used in error calculation                               }
17129:C        5             if map_l < 0.01 then error := error_l
```



```
17130:C        6               else
17131:C        6               { With small sat in cmap, hue is undifined and can't be used  }
17132:C        6               { in error calculation                                        }
17133:C        6               if map_s < 0.01 then error := error_l + error_s2
17134:C        7               else                 error := error_l + error_h2 + error_s2;
17135:C        5             end
17136:C        5           else
17137:C        5           if target_s < 0.1 then { special casewhere request is near white }
17138:C        6             begin
17139:C        6               { With small lum in cmap, sat and hue are undifined and can't }
17140:C        6               { be used in error calculation                               }
17141:C        6               if map_l < 0.01 then error := 3
17142:C        7               else
17143:C        7               { With small sat in cmap, hue is undifined and can't be used  }
17144:C        7               { in error calculation                                        }
17145:C        7               if map_s < 0.01 then error := error_l2 + error_s
17146:C        8               else                 error := error_l2 + error_h2 + error_s;
17147:C        6             end
17148:C        6           else                   { normal case }
17149:C        6           { With small lum in cmap, sat and hue are undifined and can't }
17150:C        6           { be used in error calculation                               }
17151:C        6           if map_l < 0.01 then error := 3
17152:C        7           else
17153:C        7           { With small sat in cmap, hue is undifined and can't be used  }
17154:C        7           { in error calculation                                        }
17155:C        7           if map_s < 0.01 then error := 3
17156:C        8           else                 error := error_h2;
17157:S
17158:C        4           if error < closest_error then
17159:C        5               begin
17160:C        5                 closest_error := error;
17161:C        5                 closest_index := i;
17162:C        5               end;
17163:C        4         end;
17164:C        2   return_closest_color := closest_index;
17165:C        2 end;
17166:S
17167:S
17168:D        1 procedure raster_color ( index : integer );
17169:S
17170:D        2 var
17171:D     -8 2   intensity : real;
17172:D    -32 2   h,s,l     : real;
17173:S
17174:C        2 begin
17175:C        2   with gcb^,gle_gcb^ do
17176:C        3     begin
17177:C        3       with color_table_ptr^[index] do
17178:C        4         if gamut = 1 then { b&w }
17179:C        5           begin
17180:C        5             { Numbers from Dawn (HP-9000) DGL for 2648 terminal }
17181:C        5             intensity := 0.3*red + 0.59*green + 0.11*blue;
17182:C        5             if intensity < 0.06 then info1 := 0
17183:C        6             else                     info1 := 1;
17184:C        5           end
17185:C        5         else
17186:C        5         if raster_device_rec_ptr(dev_dep_stuff)^.devicetype = 2 then { moonunit }
***WARNING: (line17187): 'ADDR' of a constant may not be supported on other implementations
17187:C        6           info1 := return_closest_color(red,green,blue,addr(init_color_table))
17188:C        6         else                                  { 9836C }
```

```
17189:C        6          if (index <= gamut) then info1 := index
17190:C        7          else
17191:C        7            info1 := return_closest_color(red,green,blue,color_table_ptr);
17192:S
17193:C        3        gle_index_color ( gle_gcb ); { this function sets color }
17194:C        3      end;
17195:C        2 end;
17196:S
17197:D        1 procedure raster_color_table ( index : integer;
17198:D        2                                parm1 : real;
17199:D        2                                parm2 : real;
17200:D   -24  2                                parm3 : real);
17201:S
17202:D   -24  2 var
17203:D   -30  2   color_list : array [1..3] of gle_shortint;
17204:D   -38  2   intensity  : real;
17205:S
17206:C        2 begin
17207:C        2   with gcb^,gle_gcb^ do
17208:C        3     begin
17209:C        3       if dgl_current_color_model = 2 then
17210:C        4         convert_hsl_to_rgb(parm1,parm2,parm3,parm1,parm2,parm3);
17211:C        3       if (index <= gamut) and (color_map_support = 1) then
17212:C        4         begin
17213:C        4           info1 := index;
17214:C        4           info2 := index;
17215:C        4           color_list[1] := trunc(parm1 * 1023 + 0.5);
17216:C        4           color_list[2] := trunc(parm2 * 1023 + 0.5);
17217:C        4           color_list[3] := trunc(parm3 * 1023 + 0.5);
17218:C        4           info_ptr1 := addr(color_list);
17219:C        4           gle_define_color_map ( gle_gcb );
17220:C        4         end;
17221:C        3       with color_table_ptr^[index] do
17222:C        4         begin
17223:C        4           red := parm1;
17224:C        4           green := parm2;
17225:C        4           blue := parm3;
17226:C        4           if index = 0 then
17227:C        5             begin
17228:C        5               if color_map_support = 1 then
17229:C        6                 dgl_background_index := 0
17230:C        6               else
17231:C        6                 begin
17232:C        6                   if gamut = 1 then { b&w }
17233:C        7                     begin
17234:C        7                       { Numbers from Dawn (HP-9000) DGL for 2648 terminal }
17235:C        7                       intensity := 0.3*red + 0.59*green + 0.11*blue;
17236:C        7                       if intensity < 0.06 then dgl_background_index := 0
17237:C        8                       else                     dgl_background_index := 1;
17238:C        7                     end
17239:C        7                   else              { moonunit }
***WARNING: (line17240): 'ADDR' of a constant may not be supported on other implementations
17240:C        7                     dgl_background_index := return_closest_color(red,green,blue,addr(init_color_table)
)
17241:C        7                 end;
17242:C        5             end;
17243:C        4         end;
17244:C        3     end;
17245:C        2 end;
17246:S
```



```
17247:D        1 procedure set_all_color_table ( anyvar list : greal_list );
17248:S
17249:D        2 var
17250:D   -96  2   color_list : array[0..47] of gle_shortint;
17251:D   -96  2   parm1,
17252:D   -96  2   parm2,
17253:D  -120  2   parm3 : real;
17254:D  -120  2   i,
17255:D  -124  2   adr  : gshortint;
17256:S
17257:C        2 begin
17258:C        2   with gcb^,gle_gcb^ do
17259:C        3     begin
17260:C        3       for i := 0 to 15 do
17261:C        4         begin
17262:C        4           adr := i * 3;
17263:C        4           parm1 := list[adr];
17264:C        4           parm2 := list[adr+1];
17265:C        4           parm3 := list[adr+2];
17266:C        4           if dgl_current_color_model = 2 then
17267:C        5             convert_hsl_to_rgb(parm1,parm2,parm3,parm1,parm2,parm3);
17268:C        4           color_list[adr]   := trunc(parm1 * 1023 + 0.5);
17269:C        4           color_list[adr+1] := trunc(parm2 * 1023 + 0.5);
17270:C        4           color_list[adr+2] := trunc(parm3 * 1023 + 0.5);
17271:C        4           with color_table_ptr^[i] do
17272:C        5             begin
17273:C        5               red := parm1;
17274:C        5               green := parm2;
17275:C        5               blue := parm3;
17276:C        5             end;
17277:C        4         end;
17278:S
17279:C        3       info1 := 0;
17280:C        3       info2 := 15;
17281:C        3       info_ptr1 := addr(color_list);
17282:C        3       gle_define_color_map ( gle_gcb );
17283:C        3     end;
17284:C        2 end;
17285:S
17286:D        1 procedure dummy_on_off ( gcb : graphics_control_block_ptr );
17287:S
17288:C        2 begin
17289:C        2 end;
17290:S
17291:D        1 procedure graphics_on_off ( gcb : graphics_control_block_ptr );
17292:S
17293:D        2 var
17294:D    -1  2   on  : boolean;
17295:S
17296:C        2 begin
17297:C        2  with gcb^ do
17298:C        3    begin
17299:C        3      on := info1 <> 0;
17300:C        3      if ( on and not graphicstate ) or
17301:C        4         ( not on and graphicstate ) then
17302:C        4        call (togglegraphicshook);
17303:C        3    end;
17304:C        2 end;
17305:S
17306:D        1 procedure dump_graphics ( mask : integer );
```

```
17307:S
17308:D        2 { Purpose:  To dump bit map to standard printer }
17309:S
17310:D        2 label 1;
17311:S
17312:D        2 const
17313:D        2   gbuffersize = 255;
17314:S
17315:D        2 var
17316:D  -256  2   gbuffer : packed array [1..gbuffersize] of char;
17317:D  -260  2   y : integer;
17318:D  -264  2   bytes_wide : integer;
17319:S
17320:S
17321:C        2 begin
17322:C        2   gbuffer[1] := chr(27); { escape sequence for graphics }
17323:C        2   gbuffer[2] := '*';
17324:C        2   gbuffer[3] := 'b';
17325:C        2   gbuffer[4] := '6';
17326:C        2   gbuffer[5] := '4';
17327:C        2   gbuffer[6] := 'W';
17328:S
17329:C        2   with gle_gcb^ do
17330:C        3     begin
17331:C        3       bytes_wide := (display_max_x - display_min_x + 8) div 8;
17332:C        3       info_ptr1 := addr(gbuffer[7]);
17333:C        3       info1     := mask;
17334:S
17335:C        3       for y := display_min_y to display_max_y do
17336:C        4         begin
17337:C        4           info2 := y;
17338:C        4           gle_get_raster ( gle_gcb );
17339:C        4           write(gfiles[4]^,gbuffer:bytes_wide+6);
17340:C        4           if ioresult <> ord(inoerror) then goto 1;
17341:C        4         end;
17342:C        3     end;
17343:C        2   write(gfiles[4]^,#27'*rB');   { terminate graphics sequence }
17344:C        2 1:
17345:C        2 end;
17346:S
17347:D        1 procedure raster_input_esc (       opcode : integer;
17348:D        2                                    isize  : integer;
17349:D        2                                    rsize  : integer;
17350:D        2                             anyvar ilist  : gint_list;
17351:D        2                             anyvar rlist  : greal_list;
17352:D        2                             var    ierr   : integer  );
17353:D        2
17354:S
17355:D        2 { Purpose : To perform an input escape function                          }
17356:S
17357:C        2 begin
17358:C        2   ierr := 1;   { no input escape functions supported }
17359:S
17360:C        2 end; { input_esc }
17361:S
17362:S
17363:D        1   {********************** gator dumper ***************************}
17364:D        1   procedure dumpgat ;
17365:D        2   label 1;
17366:S
```



```
17367:D        2   const
17368:D        2     gwidthb = 128;
17369:D        2     gmaxheight = 512;
17370:D        2     gbuffersize = gwidthb + 7;
17371:S
17372:D        2   type
17373:D        2     gbyte = 0..255;
17374:D        2     row_def = packed array [0..(1024*768)-1] of gbyte;
17375:S
17376:D        2   var
17377:D    -4  2     row : ^row_def;
17378:S
17379:D  -140  2     gbuffer : packed array [1..gbuffersize] of char;
17380:D  -148  2     i,j : integer;
17381:D  -152  2     index : integer;
17382:D  -156  2     bit_mask : integer;
17383:D  -160  2     result : integer;
17384:D  -164  2     tmp : ^shortint;
17385:D  -168  2     tmp2 : integer;
17386:S
17387:C        2   begin
17388:S
17389:C        2     tmp := anyptr(control_space + 16384);
17390:C        2     tmp2 := ((tmp^) mod 16) * 1048576;
17391:C        2     row := anyptr(tmp2);
17392:S
17393:C        2     write(gfiles[4]^,#27'*rA');   { initiate graphics sequence }
17394:S
17395:C        2     gbuffer[1] := chr(27); { escape sequence for graphics }
17396:C        2     gbuffer[2] := '*';
17397:C        2     gbuffer[3] := 'b';
17398:C        2     gbuffer[4] := '1';
17399:C        2     gbuffer[5] := '2';
17400:C        2     gbuffer[6] := '8';
17401:C        2     gbuffer[7] := 'W';
17402:S
17403:C        2     for j := 0 to 767 do
17404:C        3       begin
17405:C        3         for i := 0 to 127 do
17406:C        4           begin
17407:C        4             result := 0;
17408:C        4             index := j*1024+i*8;
17409:C        4             bit_mask := 256;
17410:C        4             for index := index to index+7 do
17411:C        5               begin
17412:C        5                 bit_mask := bit_mask div 2;
17413:C        5                 if odd(row^[index])  then result := bit_mask+result;
17414:C        5               end;
17415:C        4             gbuffer[i+8] := chr(result);
17416:C        4           end;
17417:C        3         write(gfiles[4]^,gbuffer:gwidthb+7);
17418:C        3         if ioresult <> ord(inoerror) then goto 1;
17419:C        3       end;
17420:S
17421:C        2     write(gfiles[4]^,#27'*rB');   { terminate graphics sequence }
17422:C        2   1:
17423:C        2   end;{of crtb}
17424:D        1 {********************** end gator dumper ************************}
17425:S
17426:D        1 procedure raster_output_esc (       opcode : integer;
```

```
Pascal [Rev 3.0M  6/ 4/84] DGL_RAS.TEXT            26-Dec-84 22:40:37 Page 9   *** File name: DGL_RAS ***

17427:D        2                                       isize  : integer;
17428:D        2                                       rsize  : integer;
17429:D        2                                  anyvar ilist  : gint_list;
17430:D        2                                  anyvar rlist  : greal_list;
17431:D        2                                  var    ierr   : integer  );
17432:S
17433:D        2 { Purpose : To perform an output escape funtion                          }
17434:S
17435:D        2 var
17436:D    -1  2   on  : boolean;
17437:S
17438:S
17439:S
17440:C        2 begin
17441:C        2  with gle_gcb^,raster_device_rec_ptr(dev_dep_stuff)^  do
17442:C        3    if (opcode = 52) and (devicetype <>4 ) and (devicetype <> 2) then
17443:C        4       begin
17444:C        4         if ierr = 0 then call (dumpgraphicshook);
17445:C        4       end
17446:C        4    else
17447:C        4    if (opcode = 52) and (devicetype = 2) then
17448:C        5       begin
17449:C                  if ierr = 0 then dump_graphics(-1);
17450:C                end
17451:C        5    else
17452:C        5    if (opcode = 52) and (devicetype = 4) then   {added for gator case}
17453:C        6       begin
17454:C        6         if ierr = 0 then dumpgat;
17455:C        6       end
17456:C        6    else
17457:C        6    if (opcode = 53) and (devicetype = 3) then
17458:C        7       begin
17459:C        7         if ierr = 0 then gle_await_blanking ( gle_gcb );
17460:C        7       end
17461:C        7    else
17462:C        7    if (opcode = 250) and
17463:C        8       ((devicetype = 0) or (devicetype = 2)) then
17464:C        8       begin                 { marmot and aspen and moonunit}
17465:C        8         if ierr = 0 then
17466:C        9           begin
17467:C        9             if (rlist[1] > 0.0) and ( rlist[2] > 0.0 ) then
17468:C       10               begin
17469:C       10                 display_res_x := rlist[1];
17470:C       10                 display_res_y := rlist[2];
17471:C       10               end
17472:C       10             else
17473:C       10               ierr := 4;
17474:C        9           end;
17475:C        8       end
17476:C        8    else
17477:C        8    if (opcode = 1050) and (devicetype <> 4) then  { graphics on / off }
17478:C        9      begin
17479:C        9        if ierr = 0 then
17480:C       10          begin
17481:C       10            infc1 := ilist[1];
17482:C       10            gle_graphics_on_off ( gle_gcb );
17483:C       10          end;
17484:C        9      end
17485:C        9    else
17486:C        9    if (opcode = 1051) and ((devicetype <> 2) or (devicetype <> 4))
```

```
Pascal [Rev 3.0M  6/ 4/84] DGL_RAS.TEXT            26-Dec-84 22:40:37 Page 10   *** File name: DGL_RAS ***

17487:C       10                  then  { alpha on / off }
17488:C       10      begin
17489:C       10        if ierr = 0 then
17490:C       11          begin
17491:C       11            on := ilist[1] <> 0;
17492:C       11            if ( on and not alphastate ) or
17493:C       12               ( not on and alphastate ) then
17494:C       12              call(togglealphahook);
17495:C       11          end;
17496:C       10      end
17497:C       10    else
17498:C       10    if (opcode = 1052) then
17499:C       11      begin
17500:C       11        if ierr = 0 then
17501:C       12          begin
17502:C       12            infc1 := ilist[1];
17503:C       12            if (info1 < 0) or (info1 > 3) then info1 := 0;
17504:C       12            if info1 = 1 then info1 := 2
17505:C       13            else
17506:C       13            if info1 = 2 then info1 := 1;
17507:C       12            gle_define_drawing_mode ( gle_gcb );
17508:C       12          end;
17509:C       11      end
17510:C       11    else
17511:C       11    if (opcode = 1053) and
17512:C       12       ((devicetype = 3) or (devicetype = 2)) then
17513:C       12      begin
17514:C       12        if ierr = 0 then dump_graphics(ilist[1]);
17515:C       12      end
17516:C       12    else
17517:C       12    if (opcode = 1053) and (devicetype = 4) then       {for gator}
17518:C       13       begin
17519:C       13         if ierr = 0 then dumpgat;
17520:C       13       end
17521:C       13    else
17522:C       13    if (opcode = 1054) and (devicetype <> 4) then
17523:C       14      begin
17524:C       14        if ierr = 0 then
17525:C       15          begin
17526:C       15            info1 := ilist[1];
17527:C       15            info2 := 0;
17528:C       15            gle_clear(gle_gcb);
17529:C       15          end;
17530:C       14      end
17531:C       14    else
17532:C       14    if (opcode = 10050) and (devicetype = 3) then
17533:C       15      begin
17534:C       15        if (ierr = 3) and (rsize = 48) { opcode_ck gave real size error } then
17535:C       16          begin
17536:C       16            ierr := 0;
17537:C       16            set_all_color_table(rlist);
17538:C       16          end;
17539:C       15      end
17540:C       15    else
17541:C       15      ierr := 1;
17542:C        2 end; { raster_output_esc }
17543:S
17544:D        1 procedure dgl_raster_init ( control : integer );
17545:S
17546:D        2 type
```

```
17547:D         2   default_poly_table_def = array[1..16] of poly_entry_def;
17548:S
17549:D         2   control_def = packed record
17550:D         2                   case gshortint of
17551:D         2                     0 : (whole : gshortint);
17552:D         2                     1 : (part  : packed record
17553:D         2                                    b15,b14,b13,b12,
17554:D         2                                    b11,b10,b9, b8,
17555:D         2                                    clr_inhibit,b6,b5,b4,
17556:D         2                                    b3,b2,b1,b0 : boolean;
17557:D         2                                  end);
17558:D         2                   end;
17559:S
17560:D         2 const
17561:D         2   default_poly_table = default_poly_table_def [
17562:D         2     poly_entry_def [ density :  0.0  , orient :   0.0, edge : true ], { 1 }
17563:D         2     poly_entry_def [ density :  0.125, orient :  90.0, edge : true ], { 2 }
17564:D         2     poly_entry_def [ density :  0.125, orient :   0.0, edge : true ], { 3 }
17565:D         2     poly_entry_def [ density : -0.125, orient :   0.0, edge : true ], { 4 }
17566:D         2     poly_entry_def [ density :  0.125, orient :  45.0, edge : true ], { 5 }
17567:D         2     poly_entry_def [ density :  0.125, orient : -45.0, edge : true ], { 6 }
17568:D         2     poly_entry_def [ density : -0.125, orient :  45.0, edge : true ], { 7 }
17569:D         2     poly_entry_def [ density :  0.25 , orient :  90.0, edge : true ], { 8 }
17570:D         2     poly_entry_def [ density :  0.25 , orient :   0.0, edge : true ], { 9 }
17571:D         2     poly_entry_def [ density : -0.25 , orient :   0.0, edge : true ], { 10 }
17572:D         2     poly_entry_def [ density :  0.25 , orient :  45.0, edge : true ], { 11 }
17573:D         2     poly_entry_def [ density :  0.25 , orient : -45.0, edge : true ], { 12 }
17574:D         2     poly_entry_def [ density : -0.25 , orient :  45.0, edge : true ], { 13 }
17575:D         2     poly_entry_def [ density : -0.5  , orient :   0.0, edge : true ], { 14 }
17576:D         2     poly_entry_def [ density :  1.0  , orient :   0.0, edge : false], { 15 }
17577:D         2     poly_entry_def [ density :  1.0  , orient :   0.0, edge : true ]};{ 16 }
17578:S
17579:D         2 var
17580:D    -2   2   temp_control     : control_def;
17581:D    -6   2   i                : integer;
17582:D   -10   2   temp_color_model : integer;
17583:D   -18   2   c                : real;
17584:S
17585:C         2 begin
17586:C         2   if gle_gcb^.display_name = '9837a ' then
17587:C         3     gle_gcb^.graphics_on_off := dummy_on_off
17588:S
17589:C         3   else
17590:C         3   if gle_gcb^.display_name <> '98627A' then
17591:C         4     gle_gcb^.graphics_on_off := graphics_on_off;
17592:C         2   with gle_gcb^,gcb^,raster_device_rec_ptr(dev_dep_stuff)^ do
17593:C         3     begin
17594:C         3       disp_just := centered;
17595:C         3       clipping_support := true;
17596:C         3       retroactive_polygon_support := false;
17597:C         3       retroactive_color_support := color_map_support = 1;
17598:C         3       number_markers := 19;
17599:C         3       number_dgl_linestyles := 8;
17600:C         3       maximum_polygon_vertices := 32767;
17601:S
17602:C         3       proc_output_esc  := raster_output_esc;
17603:C         3       proc_input_esc   := raster_input_esc;
17604:C         3       proc_linestyle   := raster_linestyle;
17605:C         3       proc_color       := raster_color;
17606:C         3       proc_color_table := raster_color_table;
```



```
17607:S
17608:C         3       temp_control.whole := control;
17609:C         3       if not temp_control.part.clr_inhibit then
17610:C         4         with gle_gcb^ do
17611:C         5           begin
17612:C         5             info1 := -1;                  { clear all planes }
17613:C         5             info2 := 0;
17614:C         5             gle_clear ( gle_gcb );
17615:C         5           end;
17616:S
17617:C         3       { allocate color table space }
17618:S
17619:C         3       temp_color_model := dgl_current_color_model;
17620:C         3       dgl_current_color_model := 1; { rgb }
17621:S
17622:C         3       color_table_size := 31;
17623:C         3       color_table_ptr := addr(color_table_def_space);
17624:S
17625:C         3       if (gamut > 1) then
17626:C         4         begin
17627:C         4           for i := 0 to 15 do
17628:C         5             with init_color_table[i] do
17629:C         6               raster_color_table(i,red,green,blue);
17630:C         4           raster_color_table(16,1,1,1);
17631:C         4         end
17632:C         4       else
17633:C         4         begin
17634:C         4           raster_color_table(0,0,0,0);
17635:C         4           for i := 1 to 16 do
17636:C         5             begin
17637:C         5               c := ((17-i) / 16);
17638:C         5               raster_color_table(i,c,c,c);
17639:C         5             end;
17640:C         4         end;
17641:S
17642:C         3       for i := 17 to 31 do
17643:C         4         raster_color_table(i,1,1,1);
17644:S
17645:C         3       dgl_current_color_model := temp_color_model;
17646:S
17647:C         3       gle_gcb^.info1 := 1;
17648:C         3       gle_index_color( gle_gcb);
17649:S
17650:C         3       { allocate polygon table space }
17651:S
17652:C         3       number_polygon_styles := poly_table_size;
17653:C         3       poly_table_ptr := addr(poly_table_def_space);
17654:C         3       for i := 1 to poly_table_size do
17655:C         4         poly_table_ptr^[i] := default_poly_table[i];
17656:S
17657:C         3       display_echo_mult := 1;
17658:C         3     end;
17659:C         2 end;
17660:S
17661:C         1 end. { dgl_raster }
17662:S

No errors. 2 warnings.
            ***** Nonstandard language features enabled *****
```

DGL_TOOLS

Description

DGL_TOOLS provides the function RETURN_MACHINE_TYPE.

Requirements

GLE_TYPES.

```
Pascal [Rev 3.0M  6/ 4/84] DGL_TOOLS.TEXT          26-Dec-84 22:39:23 Page 1   *** File name: DGL_TOOLS ***

     1:D        0 {                                                                                 }
     2:D        0 { Graphics Low End                                                                }
     3:D        0 {                                                                                 }
     4:D        0 { Module    = DGL_TOOLS                                                           }
     5:D        0 { Programer = BJS                                                                 }
     6:D        0 { Date      = 12-07-82                                                            }
     7:D        0 {                                                                                 }
     8:D        0 { Purpose: To provide general tools for GLE and DGL.                              }
     9:S
    10:D        0 { Rev history                                                                     }
    11:D        0 {  Created  - 12-07-82                                                            }
    12:D        0 {  Modified -  9-07-83  BJS Changed to add check for marbox                       }
    13:S
    14:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    15:S                All rights are reserved.  Copying or other
    16:S                reproduction of this program except for archival
    17:S                purposes is prohibited without the prior
    18:S                written consent of Hewlett-Packard Company.
    19:S
    20:S
    21:S                            RESTRICTED RIGHTS LEGEND
    22:S
    23:S                Use, duplication, or disclosure by the Government
    24:S                is subject to restrictions as set forth in
    25:S                paragraph (b) (3) (B) of the Rights in Technical
    26:S                Data and Computer Software clause in
    27:S                DAR 7-104.9(a).
    28:S
    29:S                HEWLETT-PACKARD COMPANY
    30:D        0       Fort Collins, Colorado                                  }
    31:S
    32:D        0 $modcal$
    32:D        0 $include 'OPTIONS'$  { ********************* Compiler options *************** }
    33:S
    34:S           { This include file specifies range checking, debug and other compiler
    35:D        0    options for the graphics library                                              }
    36:S
    37:D        0 $debug OFF$
    38:D        0 $range OFF$
    39:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    40:D        0 $FLOAT_HDW TEST$
    41:S
    42:S
    43:S
    44:S
    45:D        0 $include 'OPTIONS'$  { ********************* Compiler options *************** }
 20000:D        0 $linenum 20000$
 20001:D        0 $search 'GLE_LIB'$
 20002:S
 20003:D        0 module DGL_TOOLS;
 20004:S
 20005:D        1 export
 20006:S
 20007:D        1 function return_machine_type : integer;
 20008:S
 20009:D        1 implement
 20010:S
 20011:D        1 import gle_types;
 20012:S
 20013:D        1 function return_machine_type : integer;
```

```
Pascal [Rev 3.0M  6/ 4/84] DGL_TOOLS.TEXT          26-Dec-84 22:39:23 Page 2   *** File name: DGL_TOOLS ***

 20014:S
 20015:D        2 type
 20016:D        2   sysflag_def = packed record
 20017:D        2                   bit7,bit6,hpib,crt_config,
 20018:D        2                   kbd,high,big_graph,alpha50 : boolean;
 20019:D        2                 end;
 20020:S
 20021:D        2   crt_reg_def = packed record
 20022:D        2                   selfinit,bit14,bit13,top1,top2,highlite,
 20023:D        2                   graph,alpha,
 20024:D        2                   bit7,bit6,bit5,bit4,bit3,bit2,bit1,bit0 : boolean;
 20025:D        2                 end;
 20026:S
 20027:D        2 var
 20028:D        2   sysflag [ hex('fffed2') ] : sysflag_def;
 20029:D        2   crt_reg [ hex('51fffe') ] : crt_reg_def;
 20030:S
 20031:C        2 begin
 20032:C        2   if sysflag.big_graph then
 20033:C        3     begin { could be 9836A or 9836C or Marbox }
 20034:C        3       if sysflag.crt_config then { might be 9836C or Marbox }
 20035:C        4         begin
 20036:C        4           {  CRT reg bits 12 (top1) and 11 (top2) are defined }
 20037:C        4           {  as follows:                                      }
 20038:C        4           {    00 - Monochrome                                }
 20039:C        4           {    01 - 4 planes starting at $520000              }
 20040:C        4           {           1 pixel / byte; 4 bits / pixel          }
 20041:C        4           {    10 - 3 planes starting at $528000              }
 20042:C        4           {           8 pixel / byte (moonunit like)          }
 20043:C        4           {    11 - 8 planes starting at $520000              }
 20044:C        4           {           1 pixel / byte; 8 bits / pixel          }
 20045:S
 20046:C        4           with crt_reg do
 20047:C        5             if (not top1) and (top2) then
 20048:C        6               return_machine_type := m9836c   { 9836C }
 20049:C        6             else
 20050:C        6             if (not top1) and (not top2) then
 20051:C        7               return_machine_type := m9836a   { Marbox }
 20052:C        7             else
 20053:C        7               return_machine_type := munknown; { unknown }
 20054:C        4         end
 20055:C        4       else
 20056:C        4         begin
 20057:C        4           return_machine_type := m9836a;      { 9836A }
 20058:C        4         end;
 20059:C        3     end
 20060:C        3   else
 20061:C        3     if sysflag.alpha50 then
 20062:C        4       begin
 20063:C        4         return_machine_type := m9826a;        { 9826A }
 20064:C        4       end
 20065:C        4     else
 20066:C        4       begin
 20067:C        4         return_machine_type := m9816a;        { 9816A }
 20068:C        4       end;
 20069:C        2 end;
 20070:S
 20071:C        1 end.

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

DGL_VARS

Description

DGL_VARS contains most of the graphics library global variables and DGL types.

Requirements

DGL_TYPES and GLE_TYPES.

```
                0 {                                                                     }
                0 { Pascal work station graphics library                                }
                0 {                                                                     }
                0 { Module    = DGL_VARS                                                }
                0 { Programer = BJS                                                     }
                0 { Date      = 2/1/82                                                  }
                0 {                                                                     }
                0 { Purpose: To hold graphics library global variables                  }

                0 { Rev history                                                         }
                0 {  5-13-82  BJS - Made changes to add hp9816 support                  }
                0 {  5-24-82  BJS - Made changes to support inq of world cp values      }
                0 {  7-01-82  BJS - Made changes to support 8 color HP 9836C            }
                0 {  8-25-82  BJS - Major mods for addition of GLE                      }
                0 {  2-17-84  BDS - Changed gcb vars to global for Pascal 3.0           }

                  {     (c) Copyright Hewlett-Packard Company, 1983.
                        All rights are reserved.  Copying or other
                        reproduction of this program except for archival
                        purposes is prohibited without the prior
                        written consent of Hewlett-Packard Company.

                                    RESTRICTED RIGHTS LEGEND

                        Use, duplication, or disclosure by the Government
                        is subject to restrictions as set forth in
                        paragraph (b) (3) (B) of the Rights in Technical
                        Data and Computer Software clause in
                        DAR 7-104.9(a).

                        HEWLETT-PACKARD COMPANY
                0       Fort Collins, Colorado                                  }

                0 $modcal$
                0 $search 'TYPES',
                0         'GLE_LIB'$
                0 $include 'OPTICNS'$

                   { This include file specifies range checking, debug and other compiler
                0    options for the graphics library                                   }

                0 $debug OFF$
                0 $range OFF$
                0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
                0 $FLOAT_HOW TEST$



                0 $include 'OPTIONS'$

                0 module DGL_VARS;

                1 import DGL_TYPES,GLE_TYPES;

                1 export

              1000 1 $linenum 1000$
```



```
                1 const
                1   graphics_rev = '3.0';      { revision number }
                1   default_color_table_size = 32;
                1   default_poly_table_size = 16;
                1   poly_table_size = 16;


                1 type
                1   bounds     = record
                1                  xmin : real;
                1                  xmax : real;
                1                  ymin : real;
                1                  ymax : real;
                1                end;

                1   vir_bounds = record
                1                  xlim : real;
                1                  ylim : real;
                1                end;

                1 type
                1   c_def = packed record
                1                  red : real;
                1                green : real;
                1                 blue : real;    { 24 bytes each }
                1               end;

                1   color_table_def = array [0..31] of c_def;
                1   color_table_ptr_def = ^color_table_def;

                1   poly_entry_def = packed record
                1                     density : real;
                1                     orient : real;
                1                     edge   : boolean;  { 18 bytes each }
                1                   end;

                1   poly_table_def = array [1..poly_table_size] of poly_entry_def;
                1   poly_table_ptr_def = ^poly_table_def;

                    { a large number of system variables are alocated in dynamic memory
                      to conserve the amount of global space used.  These variables
                1     are all part of the GCB (graphics control block)                  }

                1   graphics_control_block1 =
                1     record
                1       aspect_ratio : real;          { current aspect ratio }

                1       log_aspect   : real;     { aspect ratio of the logical display limits }

                1       log_disp_lim : bounds;        { logical display limits }

                1       max_disp_lim : bounds;        { maximum display limits }

                1       def_disp_lim : bounds;        { default display limits }

                1       log_loc_lim  : bounds;        { logical locator limits }

                1       max_loc_lim  : bounds;        { maximum locator limits }

                1       def_loc_lim  : bounds;        { default locator limits }
```

```
1062:S
1063:D      1       window_lim   : bounds;        { current window }
1064:S
1065:D      1       viewport_lim : bounds;        { current viewport }
1066:S
1067:D      1       cur_disp_lim : bounds;        { current display coordinate limits  }
1068:S
1069:D      1       cur_vir_lim  : vir_bounds;    { current virtual limits }
1070:S
1071:D      1       dgl_char_width : real;        { current character width (world)  }
1072:S
1073:D      1       dgl_char_height : real;       { current character height (world) }
1074:S
1075:D      1       char_rot_h : real;            { char rot cos vector }
1076:S
1077:D      1       char_rot_w : real;            { char rot sin vector }
1078:S
1079:D      1       disp_just    : (centered,lowerleft);   { display justification }
1080:S
1081:D      1       dxunits      : real;    { # of units in the logical display coord.     }
1082:D      1       dyunits      : real;    { system.                                      }
1083:S
1084:D      1       number_dgl_linestyles      : gshortint;
1085:D      1       number_markers             : gshortint;
1086:D      1       number_polygon_styles      : gshortint;
1087:D      1       color_table_size           : gshortint;
1088:S
1089:D      1       dgl_current_color              : gshortint;
1090:D      1       dgl_current_linestyle          : gshortint;
1091:D      1       dgl_current_linewidth          : gshortint;
1092:D      1       dgl_current_timming_mode       : gshortint;
1093:D      1       dgl_current_polygon_color      : gshortint;
1094:D      1       dgl_current_polygon_style      : gshortint;
1095:D      1       dgl_current_polygon_linestyle  : gshortint;
1096:D      1       dgl_current_polygon_angle      : real;
1097:D      1       dgl_current_polygon_density    : real;
1098:D      1       dgl_current_polygon_edge       : boolean;
1099:D      1       dgl_current_polygon_crosshatch : boolean;
1100:D      1       dgl_current_color_model        : gshortint;
1101:S
1102:D      1       maximum_polygon_vertices   : gshortint;
1103:S
1104:D      1       retroactive_polygon_support      : boolean;
1105:D      1       retroactive_color_support        : boolean;
1106:S
1107:D      1       clipping_support                 : boolean;
1108:S
1109:D      1       disp_dev_adr : integer;             { display device address }
1110:S              disp_file_name : gstring255;        { name of device file, knull if
1111:D      1                                             device address is used        }
1112:D      1       loc_dev_adr  : integer;             { locator device address }
1113:S
1114:D      1       disp_eq_loc : boolean;              { true if loc and disp are same device }
1115:S
1116:D      1       poly_table_ptr : poly_table_ptr_def;
1117:S
1118:D      1       color_table_ptr : color_table_ptr_def;
1119:S
1120:D      1       marker_size_x : integer;  { marker size in device units }
1121:D      1       marker_size_y : integer;
```



```
1122:S
1123:D      1       proc_output_esc : procedure (       opc,isize,rsize : integer;
1124:D      1                                     anyvar ilist : gint_list;
1125:D      1                                     anyvar rlist : greal_list;
1126:D      1                                        var error : integer);
1127:S
1128:D      1       proc_input_esc :  procedure (       opc,isize,rsize : integer;
1129:D      1                                     anyvar ilist : gint_list;
1130:D      1                                     anyvar rlist : greal_list;
1131:D      1                                        var error : integer);
1132:S
1133:D      1       proc_linestyle :  procedure ( index : integer );
1134:S
1135:D      1       proc_color     :  procedure ( index : integer );
1136:D      1
1137:D      1       proc_color_table  :  procedure ( index : integer ; p1,p2,p3 : real );
1138:S
1139:D      1       dgl_background_index : integer;
1140:S
1141:D      1       proc_await_locator : procedure ( var echo : integer; var button : integer;
1142:D      1                                        var x,y : real );
1143:D      1
1144:D      1       proc_sample_locator : procedure ( echo : integer; var x,y : real );
1145:S
1146:D      1       dgl_polygon_color_current : boolean; { true if polygon color set in gle }
1147:D      1    end;
1148:S
1149:D      1 const
1150:S           eight_diget_epsilon =  0.00000001;       { a number which will change the
1151:S                                                      value of a 8 diget number when
1152:D      1                                               added to it                    }
1153:S
1154:D      1   { Initialization constants follow                                          }
1155:S
1156:D      1   init_color = 1;
1157:D      1   init_linestyle = 1;
1158:D      1   init_linewidth = 1;
1159:D      1   init_char_width = 0.07;
1160:D      1   init_char_height = 0.1;
1161:D      1   init_char_width_factor = 0.035;
1162:D      1   init_char_height_factor = 0.05;
1163:D      1   init_char_rot_w = 1.0;
1164:D      1   init_char_rot_h = 0.0;
1165:S
1166:D      1   init_timming_mode = 0;       { This is diff from HP 9000 due to PWS 1.0 compatiblity }
1167:D      1   init_cpx = 0;
1168:D      1   init_cpy = 0;
1169:S
1170:D      1   init_dev_adr = 0;
1171:S
1172:D      1   init_window      = bounds [ xmin : -1.0,
1173:D      1                                xmax : 1.0,
1174:D      1                                ymin : -1.0,
1175:D      1                                ymax : 1.0];
1176:S
1177:D      1   init_viewport    = bounds [ xmin : 0.0,
1178:D      1                                xmax : 1.0,
1179:D      1                                ymin : 0.0,
1180:D      1                                ymax : 1.0];
1181:S
```

```
1182 D         1   init_aspect      = 1.0;
1183 S
1184 D         1   init_vir_lim     = vir_bounds [ xlim : 1.0,
1185 D         1                                  ylim : 1.0];
1186 S
1187 D         1   init_display_lim = bounds [ xmin : 0.0,
1188 D         1                                xmax : 8000000.0,
1189 D         1                                ymin : 0.0,
1190 D         1                                ymax : 8000000.0];
1191 S
1192 D         1   init_locator_lim = bounds [ xmin : 0.0,
1193 D         1                                xmax : 8000000.0,
1194 D         1                                ymin : 0.0,
1195 D         1                                ymax : 8000000.0];
1196 S
1197 S
1198 D         1   { definitions of some standard colors (used only on raster displays)       }
1199 D         1   { that are used in locator echoing                                         }
1200 S
1201 D         1   dominate_mode = 0;
1202 D         1   non_dominate_mode = 1;
1203 D         1   erase_mode = 2;
1204 D         1   complement_mode = 3;
1205 S
1206 D         1   solid_linestyle = 1;
1207 S
1208 D         1   io_error_number = -26;
1209 D         1   graphics_error_number = -27;
1210 S
1211 D         1   { definitions of some std device adrs                                       }
1212 S
1213 D         1   internal_display = 3;
1214 D         1   internal_locator = 2;
1215 S
1216 D         1   { definitions of some defauts used for raster linestyle generation        }
1217 S
1218 D         1   initial_pattern = -1;
1219 S
1220 D         1   { The following const are for user errors                                  }
1221 S
1222 D         1   err_sys_int = 1;        { system not initialized }
1223 D         1   err_dis_int = 2;        { display not initialized }
1224 D         1   err_loc_int = 3;        { locator not initialized }
1225 D         1   err_echo_dis_int = 4;   { echo needs display initialized }
1226 D         1   err_aspect  = 6;        { illegal aspect ratio }
1227 D         1   err_bad_parms = 7;      { illegal parameters }
1228 D         1   err_out_phys  = 8;      { parms outside physical disp lim }
1229 D         1   err_out_wind  = 9;      { parms outside window lim }
1230 D         1   err_disp_eq_loc = 10;   { loc lim given when disp and loc are the same }
1231 D         1   err_out_virt  = 11;     { parms outside virt lim }
1232 D         1   err_no_display_hardware = 12;   { missing display hardware }
1233 D         1   err_out_loc = 13;               { parms outside loc lim }
1234 D         1   err_no_ctable = 14;             { device does not support color table }
1235 D         1   err_neg_points = 18;              { polygon npts < 0 }
1236 S
1237 D         1 var
1238 D         1   { define system initialization variables                                   }
1239 S
1240 D    -1   1   system_init : boolean;              { is the system init }
1241 D    -2   1   disp_init   : boolean;              { is the display init }
```



```
1242 D    -3   1   loc_init    : boolean;              { is the locator init }
1243 S
1244 D    -8   1   graphics_error : integer;          { holds last error number }
1245 S
1246 D   -12   1   gcb : ^graphics_control_block1;     { pointer to the dynamic vars }
1247 S
1248 D   -12   1   { define holders of the current position                                   }
1249 S
1250 D   -16   1   cpy         : integer;
1251 D   -20   1   cpx         : integer;
1252 D   -20   1
1253 D   -22   1   world_int_cpx : gshortint;   { last int_move / int_line cp value }
1254 D   -24   1   world_int_cpy : gshortint;
1255 S
1256 D   -25   1   int_cp : boolean;            { true if last move or line was integer }
1257 S
1258 D   -34   1   world_real_cpx : real;       { last move / line cp value }
1259 D   -42   1   world_real_cpy : real;
1260 S
1261 D   -42   1   { define the holders of the locator echo position                         }
1262 D   -42   1   { Note that the echo is set by two routines (set_echo_pos,               }
1263 D   -42   1   { calculate_viewing) and both a world cord value and a device cord       }
1264 D   -42   1   { value is kept.  This was done since the device dependent value is      }
1265 D   -42   1   { needed as a global value during echoing, yet the world cord value     }
1266 D   -42   1   { is needed as a return value for some 'snap' echoes                     }
1267 S
1268 D   -50   1   w_loc_echo_x  : real;      { world units }
1269 D   -58   1   w_loc_echo_y  : real;
1270 S
1271 D   -62   1   d_loc_echo_x  : integer;   { device units }
1272 D   -66   1   d_loc_echo_y  : integer;
1273 S
1274 D   -66   1   { conversion factors                                                      }
1275 S
1276 D   -74   1   xwtod_scale  : real;    { world to display scale  }
1277 D   -82   1   ywtod_scale  : real;
1278 S
1279 D   -90   1   xdtow_scale  : real;    { display to world scale  }
1280 D   -98   1   ydtow_scale  : real;
1281 S
1282 D  -106   1   xwtod_offset : real;    { world to display translation  }
1283 D  -114   1   ywtod_offset : real;
1284 S
1285 D  -122   1   xdtow_offset : real;    { display to world translation  }
1286 D  -130   1   ydtow_offset : real;
1287 S
1288 D  -138   1   xltod_scale  : real;    { locator to display scale }
1289 D  -146   1   yltod_scale  : real;
1290 S
1291 D  -147   1   calc_text_xform : boolean;     { true if text xform needs to be recalc    }
1292 S
1293 D  -147   1   scalef : record                          { used for int_move / int_line }
1294 D  -147   1              x_display_delta  : gshortint;
1295 D  -147   1              x_window_delta   : gshortint;
1296 D  -147   1              x_display_offset : gshortint;
1297 D  -147   1              y_display_delta  : gshortint;
1298 D  -147   1              y_window_delta   : gshortint;
1299 D  -147   1              y_display_offset : gshortint;
1300 D  -160   1            end;
1301 S
```

```
1302:D  -160  1 type
1303:D  -160  1   ls_patterns = packed array [0..7] of gshortint;
1304:S
1305:D  -160  1 const
1306:D  -160  1   raster_patterns =
1307:D  -160  1     ls_patterns [ -1     ($FFFF) (................ ), ( 1 )
1308:D  -160  1                   -256   ($FF00) ( ........         ), ( 2 )
1309:D  -160  1                   -64    ($FFC0) ( ..........       ), ( 3 )
1310:D  -160  1                   -6     ($FFFA) ( ............ .   ), ( 4 )
1311:D  -160  1                   -10    ($FFF6) ( ............ ..  ), ( 5 )
1312:D  -160  1                   -220   ($FF24) ( ........  .  .   ), ( 6 )
1313:D  -160  1                   -32640 ($8080) (.       .        ), ( 7 )
1314:D  -160  1                   -21846 ($AAAA) ( . . . . . . . .  )];( 8 )
1315:S
1316:D  -160  1 var
1317:D  -161  1   short_flag : boolean;                { true if int_move / int_line }
1318:D  -161  1                                        { can used fast internal routines }
1319:S
1320:D  -162  1   short_defaults : boolean;            { true if  viewport bounds }
1321:D  -162  1                                        { map to max edges of raster display }
1322:S
1323:D  -162  1 { input info                                                          }
1324:S
1325:D  -166  1   current_echo_type : integer;
1326:D  -170  1   display_echo_mult : integer;
1327:S
1328:D  -174  1   cursor_size_x : integer;
1329:D  -178  1   cursor_size_y : integer;
1330:S
1331:D  -182  1   cursor_color : integer;
1332:S
1333:D  -186  1   gle_gcb : graphics_control_block_ptr;
1334:D  -190  1   gle_gcbi : graphics_input_control_block_ptr;
1335:S           gle_knob_echo_gcb : graphics_control_block_ptr;  { for echo on internal
1336:S                                                              crt even if display is
1337:D  -194  1                                                     moonunit             }
1338:D  -956  1   gcb_space               : graphics_control_block1;
1339:D -1686  1   gle_gcb_space           : graphics_control_block;
1340:D -1886  1   gle_gcbi_space          : graphics_input_control_block;
1341:D -2616  1   gle_knob_echo_gcb_space : graphics_control_block;
1342:D -3384  1   color_table_def_space   : color_table_def;
1343:D -3672  1   poly_table_def_space    : poly_table_def;
1344:D -3672  1
1345:D -3672  1 const
1346:D -3672  1   dno       = 0;
1347:D -3672  1   dyes      = 1;
1348:S
1349:D -3672  1 implement
1350:S
1351:C        1 end.  { of module DGL_VAR }
1352:S

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

Description

DI_DRV provides low-level driver support.

Usage

DI_DRV is used with the 98625 interface.

Requirements

DISCINT and the I/O library kernel (IODECLARATIONS, etc.)

```
     1:S                                                    (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1983.
     4:S          All rights are reserved.  Copying or other
     5:S          reproduction of this program except for archival
     6:S          purposes is prohibited without the prior
     7:S          written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                       RESTRICTED RIGHTS LEGEND
    11:S
    12:S          Use, duplication, or disclosure by the Government
    13:S          is subject to restrictions as set forth in
    14:S          paragraph (b) (3) (B) of the Rights in Technical
    15:S          Data and Computer Software clause in
    16:S          DAR 7-104.9(a).
    17:S
    18:S          HEWLETT-PACKARD COMPANY
    19:D        0 Fort Collins, Colorado                        *)
    20:S
    21:S
    22:D        0 $MODCAL ON$
    23:D        0 $PARTIAL_EVAL ON$
    24:D        0 $STACKCHECK ON$
    25:D        0 $RANGE OFF$
    26:D        0 $DEBUG OFF$
    27:D        0 $OVFLCHECK OFF$
```



```
    28:D        0 $PAGE$
    29:S
    30:D        0 (*****************************************************************************)
    31:D        0 (*                                                                           *)
    32:D        0 (* NOT  RELEASED    VERSION  FOR    3.0                                       *)
    33:D        0 (*                                                                           *)
    34:D        0 (*****************************************************************************)
    35:D        0 (*                                                                           *)
    36:D        0 (*                                                                           *)
    37:D        0 (*      IOLIB           DISCINT_DRIVERS (98625 Disc Interface)               *)
    38:D        0 (*                                                                           *)
    39:D        0 (*                                                                           *)
    40:D        0 (*****************************************************************************)
    41:D        0 (*                                                                           *)
    42:D        0 (*                                                                           *)
    43:D        0 (*      library       -  IOLIB                                               *)
    44:D        0 (*      name          -  DISCINT_DRIVERS                                     *)
    45:D        0 (*      module(s)     -  di_drv                                              *)
    46:D        0 (*                    -  discint                                             *)
    47:D        0 (*                                                                           *)
    48:D        0 (*      author        -                                                      *)
    49:D        0 (*      phone         -                                                      *)
    50:D        0 (*                                                                           *)
    51:D        0 (*      date          -  May   5 , 1982                                      *)
    52:D        0 (*      update        -  Aug  10 , 1983                                      *)
    53:D        0 (*      release       -  ???????????????                                     *)
    54:D        0 (*                                                                           *)
    55:D        0 (*      source        -  IOLIB:DI_DRV.TEXT                                   *)
    56:D        0 (*      object        -  IOLIB:DI_DRV.CODE                                   *)
    57:D        0 (*                                                                           *)
    58:D        0 (*****************************************************************************)
```

```
    59:D        0 $PAGE$
    60:D        0 (*************************************************************************)
    61:D        0 (*                                                                       *)
    62:D        0 (*                                                                       *)
    63:D        0 (*       This is the source code for an external procedures library      *)
    64:D        0 (*       to be used for general purpose interfacing on the HP 9826.      *)
    65:D        0 (*                                                                       *)
    66:D        0 (*       The library consists of 3 primary sets of modules -             *)
    67:D        0 (*                                                                       *)
    68:D        0 (*               1.      KERNEL modules                                  *)
    69:D        0 (*               2.      driver modules                                  *)
    70:D        0 (*               3.      IOLIB  modules                                  *)
    71:D        0 (*                                                                       *)
    72:D        0 (*       The KERNEL modules consist of the following modules -           *)
    73:D        0 (*                                                                       *)
    74:D        0 (*               1.      iodeclarations  ( contains static r/w space )   *)
    75:D        0 (*               2.      iocomasm                                        *)
    76:D        0 (*               3.      general_0       ( initialization & low level    *)
    77:D        0 (*                                         routines like ioread/iowrite) *)
    78:D        0 (*       The KERNEL modules also have an executable program segement     *)
    79:D        0 (*       that gets executed at the time it is loaded.  This program      *)
    80:D        0 (*       initializes the static read/write memory.  This program also    *)
    81:D        0 (*       allocates the temporary storage for any card that exists -      *)
    82:D        0 (*       independent of whether there is or is not a driver for it.      *)
    83:D        0 (*                                                                       *)
    84:D        0 (*       The driver modules consist of the actual assembly or PASCAL     *)
    85:D        0 (*       routines that deal with a specific interface card.  There is    *)
    86:D        0 (*       also an executable program segment for each driver module.      *)
    87:D        0 (*       This program searches the select code table in the static r/w   *)
    88:D        0 (*       initialized by the KERNEL general_0 module for all select codes *)
    89:D        0 (*       that have the right interface card ( HPIB drivers will search   *)
    90:D        0 (*       for the 98624 interface ).  This program will then set up the   *)
    91:D        0 (*       driver tables to point to the correct drivers.                  *)
    92:D        0 (*                                                                       *)
    93:D        0 (*       The rest of the IOLIB modules are high-level modules that are   *)
    94:D        0 (*       used by an end user in his/her application program.             *)
    95:D        0 (*                                                                       *)
    96:D        0 (*       The KERNEL and some set of driver modules will exist in the     *)
    97:D        0 (*       SYSTEM.INITLIB file as object code ( not EXPORT text ).  The    *)
    98:D        0 (*       export text will reside on the SYSTEM.LIBRARY file.  The rest   *)
    99:D        0 (*       of the library will reside on the SYSTEM.LIBRARY.               *)
   100:D        0 (*                                                                       *)
   101:D        0 (*************************************************************************)
```



```
   102:D        0 $PAGE$
   103:D        0 (*************************************************************************)
   104:D        0 (*                                                                       *)
   105:D        0 (*                                                                       *)
   106:D        0 (*       REFERENCES :                                                    *)
   107:D        0 (*                                                                       *)
   108:D        0 (*                                                                       *)
   109:D        0 (*       1.  9826 I/O Designers Guide            ( ---------- )         *)
   110:D        0 (*                                                                       *)
   111:D        0 (*       2.  68000 Manual                        ( Motorola )           *)
   112:D        0 (*                                                                       *)
   113:D        0 (*       3.  Pascal alpha site ERS               ( ---------- )         *)
   114:D        0 (*                                                                       *)
   115:D        0 (*       4.  Pascal I/O Library ERS              ( ---------- )         *)
   116:D        0 (*                                                                       *)
   117:D        0 (*       5.  9826 HPL EIO & IOD listings         ( ---------- )         *)
   118:D        0 (*                                                                       *)
   119:D        0 (*       6.  9826 HPL Misc. I/O Doc.             ( ---------- )         *)
   120:D        0 (*                                                                       *)
   121:D        0 (*       7.  9826 card documentation             ( Mfg. Specs. )        *)
   122:D        0 (*                                                                       *)
   123:D        0 (*       8.  Pascal I/O Library IRS              ( ---------- )         *)
   124:D        0 (*                                                                       *)
   125:D        0 (*                                                                       *)
   126:D        0 (*************************************************************************)
```

```
127:D        0 $PAGE$
128:D        0 PROGRAM discint_initialize ( INPUT , OUTPUT );
129:S
130:S
131:S                  { This module has a program segment so that there is
132:S                    an executable entry point into the module.
133:D        1           At INITLIB time this program is executed. }
```



```
134:D        1 $PAGE$
135:D        1 EXTERNAL MODULE extdi;
136:S                  {
137:S                    date    05/03/82
138:S                    update  08/10/83
139:S
140:S                    purpose This module is a declaration of the importation text for
141:S                            the external drivers.
142:S
143:S                    note    The assembly language code that is imported needs to be
144:S                            called 'extdi'.  The routines need to be called
145:S                            'extdi_@@@@@' - edi_init referenced below would be
146:S                            extdi_edi_init.  'edi' refers to external disc interface.
147:D        1           }
148:S
149:D        1 IMPORT  $SEARCH 'IOLIB:KERNEL.CODE'$
150:D        1           sysglobals ,
151:D        1           iodeclarations ;
152:S
153:D        1 EXPORT
154:S
155:D        1   PROCEDURE edi_init  ( temp : ANYPTR );
156:D        1   PROCEDURE edi_isr   ( temp : PISRIB );
157:D        1   PROCEDURE edi_rdb   ( temp : ANYPTR ;  VAR x : CHAR);
158:D        1   PROCEDURE edi_wtb   ( temp : ANYPTR ;  val  : CHAR);
159:D        1   PROCEDURE edi_rdw   ( temp : ANYPTR ;  VAR x : io_word);
160:D        1   PROCEDURE edi_wtw   ( temp : ANYPTR ;  val  : io_word);
161:D        1   PROCEDURE edi_rds   ( temp : ANYPTR ;  reg  : io_word;
162:D        2                                          VAR x : io_word);
163:D        1   PROCEDURE edi_wtc   ( temp : ANYPTR ;  reg  : io_word;
164:D        2                                          val  : io_word );
165:D        1   PROCEDURE edi_tfr   ( temp : ANYPTR ;  bcb  : ANYPTR );
166:D        1   PROCEDURE edi_send  ( temp : ANYPTR ;  val  : CHAR );
167:D        1   PROCEDURE edi_ppoll ( temp : ANYPTR ;  VAR x : CHAR );
168:D        1   PROCEDURE edi_clr   ( temp : ANYPTR ;  line : io_bit );
169:D        1   PROCEDURE edi_set   ( temp : ANYPTR ;  line : io_bit );
170:D        1   PROCEDURE edi_test  ( temp : ANYPTR ;  line : io_bit ;
171:D        2                                          VAR x : BOOLEAN );
172:D        1   PROCEDURE edi_end   ( temp : ANYPTR ;  VAR x : BOOLEAN );
173:S
174:D        1 END; { of extdi }
```

```
175:D        1 $PAGE$
176:S
177:S
178:D        1 MODULE init_discint;
179:S                {
180:S                  date    05/03/82
181:S                  update  08/10/83  vers. number
182:S
183:S                  purpose This module initializes the DISCINT drivers.
184:S
185:D        1         }
186:S
187:D        1 IMPORT    iodeclarations ;
188:S
189:S
190:D        1 EXPORT
191:D        1   VAR
192:D  -120  1     discint_drivers  : drv_table_type;
193:S
194:S
195:D  -120  1   PROCEDURE io_init_discint;
196:S
197:S
198:D  -120  1 IMPLEMENT
199:S
200:D  -120  1   IMPORT  sysglobals ,
201:D  -120  1           isr ,
202:D  -120  1           general_0 ,
203:D  -120  1           extdi ;
204:S
205:S
206:D        1   PROCEDURE io_init_discint;
207:D        2   VAR
208:D    -2  2     io_isc        : type_isc;
209:D    -6  2     dummy         : INTEGER;
210:D    -8  2     io_lvl        : io_byte;
211:C        2   BEGIN
212:S
213:C        2     io_revid := io_revid + ' DI3.0';
214:S
215:C        2     { set up the driver tables }
216:S
217:C        2     WITH discint_drivers DO BEGIN
218:C        3       discint_drivers := dummy_drivers;
219:C        3       iod_init  := edi_init;
220:C        3       iod_isr   := edi_isr;
221:C        3       iod_rdb   := edi_rdb;
222:C        3       iod_wtb   := edi_wtb;
223:C        3       iod_rdw   := edi_rdw;
224:C        3       iod_wtw   := edi_wtw;
225:C        3       iod_rds   := edi_rds;
226:C        3       iod_wtc   := edi_wtc;
227:C        3       iod_end   := edi_end;
228:C        3       iod_tfr   := edi_tfr;
229:C        3       iod_send  := edi_send;
230:C        3       iod_ppoll := edi_ppoll;
231:C        3       iod_set   := edi_set;
232:C        3       iod_clr   := edi_clr;
233:C        3       iod_test  := edi_test;
234:C        3     END; { of WITH }
```



```
235:S
236:S
237:C        2     { set up drivers for the interfaces }
238:S
239:C        2     FOR io_isc:=iominisc TO iomaxisc DO
240:C        3       WITH isc_table[io_isc] DO BEGIN
241:S
242:C        4         IF card_id = hp98625
243:C        5           THEN BEGIN
244:S
245:C        5             io_drv_ptr:=ADDR(discint_drivers);
246:S
247:C        5             WITH io_drv_ptr^, io_tmp_ptr^ DO BEGIN
248:C        6               io_lvl:=((ioread_byte(io_isc,3) DIV 16) MOD 4)+3;
249:S
250:C        6               { if the card exists then link in an ISR for it }
251:C        6               { ??? - what happens if an ISR fires during init }
252:S
253:C        6               IF myisrib.INTREGADDR <> NIL
254:C        7                 THEN BEGIN
255:C        7                   { remove any existant isr }
256:C        7                   ISRUNLINK(io_lvl,              { interrupt level       }
257:C        7                             addr(myisrib));      { ptr to the isrib      }
258:C        7                 END; { of IF }
259:S
260:C        6               PERMISRLINK(iod_isr,                              { isr          }
261:C        6                     ANYPTR(INTEGER(card_ptr)+3),                { int reg addr }
262:C        6                     192,                                        { int reg mask }
263:C        6                     192,                                        { int reg value }
264:C        6                     io_lvl,                                     { int level    }
265:C        6                     ADDR(myisrib));                             { isrib info   }
266:C        6             END; { of WITH // DO BEGIN }
267:S
268:C        5           END; { of IF card_id }
269:S
270:C        4       END; { of FOR io_isc WITH isc_table[io_isc] BEGIN }
271:S
272:S
273:C        2     { call the actual driver initialization }
274:S
275:S               { this is seperate from the set up stuff in case
276:S                 there are 2 or more cards connected - and generate
277:C        2        an isr }
278:S
279:C        2     FOR io_isc:=iominisc TO iomaxisc DO
280:C        3       WITH isc_table[io_isc] DO
281:C        4       IF card_id = hp98625
282:C        5         THEN BEGIN
283:C        5           CALL(io_drv_ptr^.iod_init , io_tmp_ptr);
284:C        5         END; { of WITH IF }
285:S
286:S
287:C        2   END; { of io_init_discint }
288:S
289:C        1 END; { of MODULE init_discint }
```

```
240:D         1 $PAGE$
241:S
242:D         1 IMPORT    init_discint ,
243:D         1           LOADER  ;                                          { 367 TM 9/22/82 }
244:S
245:S
246:C         1 BEGIN
247:C         1   io_init_discint;
248:C         1   MARKUSER;                                                  { 367 TM 9/22/82 }
249:C         1 END. { of discint_initialize }
300:S
301:S
```

No errors. No warnings.
***** Nonstandard language features enabled *****

DISCHPIB

Description

DISCHPIB is the common message-oriented HP-IB protocol interface used by the Amigo and CS80 mass storage transfer methods.

Requirements

IODECLARATIONS, HPIB or DISC_INTF

Optionally: DMA

```
                                                (*
                                *)

$modcal, debug off, range off, ovflcheck off, stackcheck off, iocheck off$
$search 'DRVASM', 'IOLIB:KERNEL'$


program init_bkgnd;

module bkgnd;

import
  sysglobals, misc, iodeclarations;

export
  type
    uep_type = {unit entry pointer}
      ^unitentry;
    bi_type = {background info}
      record
        iores               : iorsltwd;      {ioresult                    }
        set_in_use          : boolean;       {allocation flag             }
        async               : boolean;       {overlapped transfer flag    }
        eot                 : eotproc;       {end of transfer procedure   }
        fibptr              : fibp;          {file information block ptr  }
        read_operation      : boolean;       {transfer direction flag     }
        buffered_transfer   : boolean;       {amigo driver flag           }
        xfr_chain_semaphore : boolean;       {driver semaphore            }
        bx_tries            : shortint;      {number of previous tries    }
        bx_strt_rcrd        : integer;       {record address              }
        bx_bufptr           : charptr;       {R/W address pointer         }
        bx_length           : integer;       {total transfer byte count   }
        bx_tfr_length       : integer;       {intermediate tfr byte count }
        bdx_chain_semaphore : boolean;       {data transfer semaphore     }
        bdx_pre_eoi         : boolean;       {premature eoi flag          }
        bdx_nb              : integer;       {data transfer byte count    }
        bdx_proc            : io_proc;       {data transfer completion proc }
        buf_info            : buf_info_type; {as defined by the iolibrary }
      end;
    bip_type = ^bi_type;
```



```
$page$

  procedure initialize_bkgnd;
  procedure allocate_bkgnd_info  (uep: uep_type);
  procedure deallocate_bkgnd_info(uep: uep_type);
  procedure abort_bkgnd_process  (uep: uep_type);
  procedure ioresc_bkgnd         (uep: uep_type; ior: iorsltwd);
  function  unit_busy            (uep: uep_type): boolean;
  procedure unit_wait            (uep: uep_type);

implement {bkgnd}

const
  n = 3;  {number of bkgnd info sets}

type
  bia_type = array[0..n-1] of bi_type;

var
  bia_ptr: ^bia_type;
  index: shortint;


procedure initialize_bkgnd;
  var
    local_index: shortint;
  begin {initialize_bkgnd}
    if bia_ptr=nil then
      new(bia_ptr);
    for local_index := 0 to n-1 do
      bia_ptr^[local_index].set_in_use := false;
    index := 0;
  end; {initialize_bkgnd}


function bip_valid(bip: bip_type): boolean;
  var
    local_index: shortint;
  begin {bip_valid}
    bip_valid := false;
    for local_index := 0 to n-1 do
      if bip=addr(bia_ptr^[local_index]) then
        bip_valid := true;
  end; {bip_valid}
```

```
103:D    -6  1 $page$
104:S
105:D        1 procedure allocate_bkgnd_info(uep: uep_type);
106:D        2   var
107:D    -4  2     bip: bip_type;
108:C        2   begin {allocate_bkgnd_info}
109:C        2     lockup;
110:C        2     repeat
111:C        3       bip := addr(bia_ptr^[index]);
112:C        3       index := index+1;
113:C        3       if index>=n then index := 0;
114:C        3     until not bip^.set_in_use;
115:C        2     uep^.dvrtemp := integer(bip);
116:C        2     with bip^ do
117:C        3       begin
118:C        3         iores      := inoerror;
119:C        3         set_in_use := true;
120:C        3         async      := false;
121:C        3       end; {with}
122:C        2   end; {allocate_bkgnd_info}
123:S
124:S
125:D        1 procedure deallocate_bkgnd_info(uep: uep_type);
126:D        2   var
127:D    -4  2     saved_ioresult: integer;
128:C        2   begin {deallocate_bkgnd_info}
129:C        2     if bip_valid(bip_type(uep^.dvrtemp)) then
130:C        3       with bip_type(uep^.dvrtemp)^ do
131:C        4         begin
132:C        4           set_in_use := false;
133:C        4           uep^.dvrtemp := ord(iores);
134:C        4           lockdown;
135:C        4           if async then {call the eot procedure}
136:C        5             begin
137:C        5               saved_ioresult := ioresult;
138:C        5               ioresult := uep^.dvrtemp;
139:C        5               call(feot, fibptr);
140:C        5               ioresult := saved_ioresult;
141:C        5             end; {if}
142:C        4         end; {with}
143:C        2   end; {deallocate_bkgnd_info}
144:S
145:S
146:D        1 procedure abort_bkgnd_process(uep: uep_type);
147:C        2   begin {abort_bkgnd_process}
148:C        2     if escapecode<>-10 then  {prevent any eot procedure call while deallocating}
149:C        3       bip_type(uep^.dvrtemp)^.async := false;
150:C        2     deallocate_bkgnd_info(uep);
151:C        2     if escapecode<>-10 then escape(escapecode);
152:C        2   end; {abort_bkgnd_process}
153:S
154:S
155:D        1 procedure ioresc_bkgnd(uep: uep_type; ior: iorsltwd);
156:C        2   begin {ioresc_bkgnd}
157:C        2     if bip_valid(bip_type(uep^.dvrtemp)) then
158:C        3       bip_type(uep^.dvrtemp)^.iores := ior;
159:C        2     escape(-10);
160:C        2   end; {ioresc_bkgnd}
```



```
161:D    -6  1 $page$
162:S
163:D        1 function unit_busy(uep: uep_type): boolean;
164:C        2   begin {unit_busy}
165:C        2     if bip_valid(bip_type(uep^.dvrtemp)) then
166:C        3       unit_busy := true
167:C        3     else
168:C        3       begin
169:C        3         unit_busy := false;
170:C        3         ioresult := uep^.dvrtemp;
171:C        3       end; {else}
172:C        2   end; {unit_busy}
173:S
174:S
175:D        1 procedure unit_wait(uep: uep_type);
176:C        2   begin {unit_wait}
177:C        2     while unit_busy(uep) do {nothing};
178:C        2   end; {unit_wait}
179:S
180:S
181:C        1 end; {bkgnd}
```

```
182:D        1 $page$
183:D        1 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'$
184:S
185:S
186:D        1 module discHPIB;
187:S
188:D        1 import
189:D        1   sysglobals, iodeclarations, drvasm, bkgnd;
190:S
191:D        1 export
192:D        1   function  Simon_no_DMA       (uep: uep_type): boolean;
193:D        1   function  Simon_DMA          (uep: uep_type): boolean;
194:D        1   procedure HPIBcheck_sc       (uep: uep_type);
195:D        1   procedure HPIBwait_for_ppol  (uep: uep_type);
196:D        1   procedure HPIBshort_msge_out (uep: uep_type; sec: byte; bp: charptr; nb: shortint);
197:D        1   procedure HPIBamigo_clear    (uep: uep_type);
198:D        1   procedure HPIBshort_msge_in  (uep: uep_type; sec: byte; bp: charptr; nb: shortint);
199:D        1   function  HPIBamigo_identify (uep: uep_type): shortint;
200:D        1   procedure HPIBget_amigo_ident(uep: uep_type; var ident: shortint);
201:D        1   procedure HPIBupon_ppol_resp (uep: uep_type; proc: io_proc);
202:D        1   procedure HPIBupon_dxfr_comp (uep: uep_type; sec: byte; bp: charptr; nb: integer; proc: io_proc);
203:D        1
204:D        1 implement {discHPIB}
205:S
206:S
207:D        1 const
208:D        1   standard_tc = 5000;   {standard byte timeout value milliseconds}
209:D        1   short_tc = 25;        {short byte timeout value milliseconds}
210:D        1   SDC = 4;              {selective device clear}
211:D        1   LAG_base = 32;        {listen address group base}
212:D        1   TAG_base = 64;        {talk address group base}
213:D        1   SCG_base = 96;        {secondary command group base}
214:S
215:S
216:D        1 procedure delay_timer(microsec_value: integer); external;
217:S
218:S
219:D        1 procedure confirm_timeout_and_reset_card(uep: uep_type);
220:C        2   begin {confirm_timeout_and_reset_card}
221:C        2     if (escapecode<>ioescapecode) or (ioe_isc<>uep^.sc) then escape(escapecode);
222:C        2     if ioe_result<>ioe_timeout then ioresc_bkgnd(uep, znodevice);
223:C        2     with isc_table[uep^.sc] do
224:C        3       call(io_drv_ptr^.iod_init, io_tmp_ptr);
225:C        2     ioresc_bkgnd(uep, ztimeout);
226:C        2   end; {confirm_timeout_and_reset_card}
227:S
```



```
228:D        1 $page$
229:S
230:D        1 function Simon_no_DMA(uep: uep_type): boolean;
231:C        2   begin {Simon_no_DMA}
232:C        2     with isc_table[uep^.sc] do
233:C        3       Simon_no_DMA := (card_id=hp98625) and not dma_here;
234:C        2   end; {Simon_no_DMA}
235:S
236:S
237:D        1 function Simon_DMA(uep: uep_type): boolean;
238:C        2   begin {Simon_DMA}
239:C        2     with isc_table[uep^.sc] do
240:C        3       Simon_DMA := (card_id=hp98625) and dma_here;
241:C        2   end; {Simon_DMA}
242:S
243:S
244:D        1 procedure HPIBcheck_sc(uep: uep_type);
245:C        2   begin {HPIBcheck_sc}
246:C        2     with isc_table[uep^.sc] do
247:C        3       begin
248:C        3         if card_type<>hpib_card then
249:C        4           ioresc_bkgnd(uep, znodevice);
250:C        3         with io_tmp_ptr^ do
251:C        4           while (in_bufptr<>nil) or (out_bufptr<>nil) do {nothing};
252:C        3       end; {with}
253:C        2   end; {HPIBcheck_sc}
254:S
255:S
256:D        1 procedure HPIBwait_for_ppol(uep: uep_type);
257:D        2   var
258:D    -2  2     pprb: packed array[0..7] of boolean;        {parallel poll response byte}
259:C        2   begin {HPIBwait_for_ppol}
260:C        2     try
261:C        3       with isc_table[uep^.sc], io_drv_ptr^ do
262:C        4         repeat
263:C        5           call(iod_ppoll, io_tmp_ptr, charptr(addr(pprb))^);
264:C        5         until pprb[uep^.ba];
265:C        3     recover
266:C        3       confirm_timeout_and_reset_card(uep);
267:C        2   end; {HPIBwait_for_ppol}
```

```
266:D        1 $page$
269:S
270:D        1 procedure address_for_msge_out(var isc_te: isc_table_type; ba, sec: byte; tc: integer);
271:D        2   var
272:D    -1  2     dummy_char: char;
273:C        2   begin {address_for_msge_out}
274:C        2     with isc_te, io_drv_ptr^ do
275:C        3       begin
276:C        3         call(iod_send, io_tmp_ptr, '?');
277:C        3         io_tmp_ptr^.timeout := tc;
278:C        3         call(iod_send, io_tmp_ptr, chr(TAG_base+io_tmp_ptr^.addressed));
279:C        3         call(iod_send, io_tmp_ptr, chr(LAG_base+ba));
280:C        3         if card_id<>hp98625 then  {delay to avoid Chinook bug}
281:C        4           delay_timer(85 {microseconds});
282:C        3         call(iod_send, io_tmp_ptr, chr(SCG_base+sec));
283:C        3         call(iod_ppoll, io_tmp_ptr, dummy_char);  {enforce timeout}
284:C        3       end; {with}
285:C        2   end; {address_for_msge_out}
286:S
287:D        1 procedure HPIBshort_msge_out(uep: uep_type; sec: byte; bp: charptr; nb: shortint);
288:D        2   var
289:D    -4  2     isc_te_ptr: ^isc_table_type;
290:C        2   begin {HPIBshort_msge_out}
291:C        2     try
292:C        3       isc_te_ptr := addr(isc_table[uep^.sc]);
293:C        3       address_for_msge_out(isc_te_ptr^, uep^.ba, sec, standard_tc);
294:C        3       with isc_te_ptr^, io_drv_ptr^ do
295:C        4         begin
296:C        4           while nb>1 do
297:C        5             begin
298:C        5               call(iod_wtb,  io_tmp_ptr, bp^);
299:C        5               bp := addr(bp^, 1);
300:C        5               nb := nb-1;
301:C        5             end; {while}
302:C        4           call(iod_set,  io_tmp_ptr, ord(eoi_line));
303:C        4           call(iod_wtb,  io_tmp_ptr, bp^);
304:C        4           call(iod_send, io_tmp_ptr, '?');
305:C        4         end; {with}
306:C        3     recover
307:C        3       confirm_timeout_and_reset_card(uep);
308:C        2   end; {HPIBshort_msge_out}
309:S
310:D        1 procedure HPIBamigo_clear(uep: uep_type);
311:D        2   var
312:D    -4  2     isc_te_ptr: ^isc_table_type;
313:C        2   begin {HPIBamigo_clear}
314:C        2     try
315:C        3       isc_te_ptr := addr(isc_table[uep^.sc]);
316:C        3       address_for_msge_out(isc_te_ptr^, uep^.ba, 16, standard_tc);
317:C        3       with isc_te_ptr^, io_drv_ptr^ do
318:C        4         begin
319:C        4           call(iod_set,  io_tmp_ptr, ord(eoi_line));
320:C        4           call(iod_wtb,  io_tmp_ptr, chr(0));  {disable parity check}
321:C        4           call(iod_send, io_tmp_ptr, chr(SDC));
322:C        4           call(iod_send, io_tmp_ptr, '?');
323:C        4         end; {with}
324:C        3     recover
325:C        3       confirm_timeout_and_reset_card(uep);
326:C        2   end; {HPIBamigo_clear}
```



```
327:D        1 $page$
328:S
329:D        1 procedure address_for_msge_in(var isc_te: isc_table_type; ba, sec: byte; tc: integer);
330:D        2   var
331:D    -1  2     dummy_char: char;
332:C        2   begin {address_for_msge_in}
333:C        2     with isc_te, io_drv_ptr^ do
334:C        3       begin
335:C        3         call(iod_send, io_tmp_ptr, '?');
336:C        3         io_tmp_ptr^.timeout := tc;
337:C        3         call(iod_send, io_tmp_ptr, chr(LAG_base+io_tmp_ptr^.addressed));
338:C        3         call(iod_send, io_tmp_ptr, chr(TAG_base+ba));
339:C        3         if card_id<>hp98625 then  {delay to avoid Chinook bug}
340:C        4           delay_timer(85 {microseconds});
341:C        3         call(iod_send, io_tmp_ptr, chr(SCG_base+sec));
342:C        3         call(iod_ppoll, io_tmp_ptr, dummy_char);  {enforce timeout}
343:C        3         if card_id<>hp98625 then  {delay to avoid Chinook bug}
344:C        4           delay_timer(85 {microseconds});
345:C        3       end; {with}
346:C        2   end; {address_for_msge_in}
347:S
348:S
349:D        1 procedure premature_eoi(uep: uep_type);
350:C        2   begin {premature_eoi}
351:C        2     with isc_table[uep^.sc] do
352:C        3       call(io_drv_ptr^.iod_send, io_tmp_ptr, '_');
353:C        2     ioresc_bkgnd(uep, zbadhardware);  {all "expected" premature eoi's have to be trapped}
354:C        2   end; {premature_eoi}
355:S
356:S
357:D        1 procedure HPIBshort_msge_in(uep: uep_type; sec: byte; bp: charptr; nb: shortint);
358:D        2   var
359:D    -4  2     isc_te_ptr: ^isc_table_type;
360:D    -5  2     eoi_set: boolean;
361:C        2   begin {HPIBshort_msge_in}
362:C        2     try
363:C        3       isc_te_ptr := addr(isc_table[uep^.sc]);
364:C        3       address_for_msge_in(isc_te_ptr^, uep^.ba, sec, standard_tc);
365:C        3       with isc_te_ptr^, io_drv_ptr^ do
366:C        4         begin
367:C        4           while nb>1 do
368:C        5             begin
369:C        5               call(iod_rdb,  io_tmp_ptr, bp^);
370:C        5               call(iod_end,  io_tmp_ptr, eoi_set);
371:C        5               if eoi_set then premature_eoi(uep);
372:C        5               bp := addr(bp^, 1);
373:C        5               nb := nb-1;
374:C        5             end; {while}
375:C        4           call(iod_rdb,  io_tmp_ptr, bp^);
376:C        4           call(iod_send, io_tmp_ptr, '_');
377:C        4         end; {with}
378:C        3     recover
379:C        3       confirm_timeout_and_reset_card(uep);
380:C        2   end; {HPIBshort_msge_in}
```

```
   361:D        1 $page$
   362:S
   363:D        1 function HPIBamigo_identify(uep: uep_type): shortint;
   364:D        2   var
   365:D    -4  2     isc_te_ptr: ^isc_table_type;
   366:D    -4  2     ident: {the two identify bytes}
   367:D    -4  2       record case integer of
   368:D    -4  2         0: (word: shortint);
   369:D    -4  2         1: (upper_char, lower_char: char);
   370:D    -6  2       end;
   371:D    -7  2     eoi_set: boolean;
   372:C        2   begin {HPIBamigo_identify}
   373:C        2     try
   374:C        3       isc_te_ptr := addr(isc_table[uep^.sc]);
   375:C        3       address_for_msge_in(isc_te_ptr^, {"ba"} 31, {"sec"} uep^.ba, short_tc);
   376:C        3       with isc_te_ptr^, io_drv_ptr^ do
   377:C        4         begin
   378:C        4           call(iod_rdb,  io_tmp_ptr, ident.upper_char);
   379:C        4           call(iod_end,  io_tmp_ptr, eoi_set);
   380:C        4           if eoi_set then premature_eoi(uep);
   381:C        4           call(iod_rdb,  io_tmp_ptr, ident.lower_char);
   382:C        4           call(iod_send, io_tmp_ptr, chr(TAG_base+io_tmp_ptr^.addressed));
   383:C        4         end; {with}
   384:C        3     recover
   385:C        3       confirm_timeout_and_reset_card(uep);
   386:C        2     HPIBamigo_identify := ident.word;
   387:C        2   end; {HPIBamigo_identify}
   388:S
   389:S
   390:D        1 procedure HPIBget_amigo_ident(uep: uep_type; var ident: shortint);
   391:C        2   begin {HPIBget_amigo_ident}
   392:C        2     ident := HPIBamigo_identify(uep);
   393:C        2   end; {HPIBget_amigo_ident}
   394:S
   395:S
   396:D        1 procedure HPIBupon_ppol_resp(uep: uep_type; proc: io_proc);
   397:S            {
   398:S              NOTE: when SIM0N drivers become available, this routine needs to
   399:S              be modified to utilize the "interrupt on parallel poll response"
   400:S              capability of SIM0N.  However, until then, this will have to do.
   401:D        2   }
   402:D        2   var
   403:D    -2  2     pprb: packed array[0..7] of boolean;        {parallel poll response byte}
   404:C        2   begin {HPIBupon_ppol_resp}
   405:C        2     try
   406:C        3       with isc_table[uep^.sc], io_drv_ptr^ do
   407:C        4         repeat
   408:C        5           call(iod_ppoll, io_tmp_ptr, charptr(addr(pprb))^);
   409:C        5         until pprb[uep^.ba];
   410:C        3     recover
   411:C        3       confirm_timeout_and_reset_card(uep);
   412:C        2     call(proc, uep);
   413:C        2   end; {HPIBupon_ppol_resp}
```



```
   414:D        1 $page$
   415:S
   416:D        1 procedure enter_bdx_chain(uep: uep_type);  forward;
   417:D        1 procedure initiate_transfer(uep: uep_type);  forward;
   418:D        1 procedure upon_transfer_complete(uep: anyptr); forward;
   419:S
   420:D        1 procedure HPIBupon_dxfr_comp(uep: uep_type; sec: byte; bp: charptr; nb: integer; proc: io_proc);
   421:D        2   var
   422:D    -4  2     isc_te_ptr: ^isc_table_type;
   423:D    -6  2     t_dir: dir_of_tfr;
   424:C        2   begin {HPIBupon_dxfr_comp}
   425:C        2     try
   426:C        3       isc_te_ptr := addr(isc_table[uep^.sc]);
   427:C        3       with bip_type(uep^.dvrtemp)^ do
   428:C        4         begin
   429:S
   430:C        4           if read_operation then
   431:C        5             begin
   432:C        5               address_for_msge_in (isc_te_ptr^, uep^.ba, sec, standard_tc);
   433:C        5               t_dir := to_memory;
   434:C        5             end {then}
   435:C        5           else
   436:C        5             begin
   437:C        5               address_for_msge_out(isc_te_ptr^, uep^.ba, sec, standard_tc);
   438:C        5               t_dir := from_memory;
   439:C        5             end; {else}
   440:S
   441:C        4           bdx_nb := nb;
   442:C        4           bdx_proc := proc;
   443:S
   444:C        4           with isc_te_ptr^, buf_info do
   445:C        5             begin
   446:C        5               drv_tmp_ptr         :=  io_tmp_ptr;
   447:C        5               active_isc          :=  no_isc;
   448:C        5               {  act_tfr is set by the driver  }
   449:C        5               if dma_here
   450:C        6                 then usr_tfr      :=  overlap_FASTEST  {DMA, or BURST FHS if both channels are busy}
   451:C        6                 else usr_tfr      :=  serial_FHS;      {unlike BURST FHS, won't lock out interrupts}
   452:C        5               b_w_mode            :=  false;
   453:C        5               {  end_mode is setup in initiate_transfer  }
   454:C        5               direction           :=  t_dir;
   455:C        5               term_char           :=  -1;
   456:C        5               {  term_count is setup in initiate_transfer  }
   457:C        5               buf_ptr             :=  anyptr(bp);
   458:C        5               buf_size            :=  nb;
   459:C        5               buf_empty           :=  anyptr(bp);
   460:C        5               buf_fill            :=  anyptr(bp);
   461:C        5               eot_proc.real_proc  :=  upon_transfer_complete;
   462:C        5               eot_parm            :=  uep;
   463:C        5               dma_priority        :=  card_id=hp98625;
   464:C        5             end; {with}
   465:S
   466:C        4           bdx_chain_semaphore := false;
   467:C        4           enter_bdx_chain(uep);
   468:S
   469:C        4         end; {with}
   470:C        3     recover
   471:C        3       confirm_timeout_and_reset_card(uep);
   472:C        2   end; {HPIBupon_dxfr_comp}
```

```
   493:D        1 $page$
   494:S
   495:D        1 procedure enter_bdx_chain(uep: uep_type);
   496:C        2   begin {enter_bdx_chain}
   497:C        2     with bip_type(uep^.dvrtemp)^ do
   498:C        3       if not test_and_toggle(bdx_chair_semaphore) then
   499:C        4         repeat
   500:C        5           initiate_transfer(uep);
   501:C        5         until test_and_toggle(bdx_chain_semaphore);
   502:C        2   end; {enter_bdx_chain}
   503:S
   504:D        1 procedure initiate_transfer(uep: uep_type);
   505:D        2   var
   506:D    -4  2     maximum_term_count: integer;
   507:C        2   begin {initiate_transfer}
   508:C        2     with bip_type(uep^.dvrtemp)^, isc_table[uep^.sc], buf_info do
   509:C        3       begin
   510:C        3         if (usr_tfr=serial_FHS) or (card_id=hp98625)
   511:C        4           then maximum_term_count := maxint  {"no" limitation}
   512:C        4           else maximum_term_count := 65536;  {DMA hardware/9914 driver limitation}
   513:C        3         if bdx_nb<=maximum_term_count
   514:C        4           then term_count := bdx_nb
   515:C        4           else term_count := maximum_term_count;
   516:C        3         bdx_nb := bdx_nb-term_count;
   517:C        3         end_mode := (direction=to_memory) or (bdx_nb=0);
   518:C        3         call(io_drv_ptr^.iod_tfr, io_tmp_ptr, addr(buf_info));
   519:C        3       end; {with}
   520:C        2   end; {initiate_transfer}
   521:S
   522:D        1 procedure upon_transfer_complete(uep: anyptr);
   523:D        2   var
   524:D    -1  2     unaddressing_char: char;
   525:C        2   begin {upon_transfer_complete}
   526:C        2     with bip_type(uep_type(uep)^.dvrtemp)^, isc_table[uep_type(uep)^.sc], io_drv_ptr^, buf_info do
   527:C        3       try
   528:C        4         if direction=to_memory then  {check for premature transfer termination}
   529:C        5           if bdx_nb=0
   530:C        6             then bdx_pre_eoi := term_count<>0
   531:C        6             else call(iod_end, io_tmp_ptr, bdx_pre_eoi)
   532:C        6         else
   533:C        5           bdx_pre_eoi := false;
   534:C        4         if (bdx_nb>0) and not bdx_pre_eoi then {re-initiate the transfer}
   535:C        5           enter_bdx_chain(uep)
   536:C        5         else {unaddress the bus and call the specified end-of-transfer procedure}
   537:C        5           begin
   538:C        5             if direction=to_memory
   539:C        6               then unaddressing_char := '_'     {untalk}
   540:C        6               else unaddressing_char := '?';    {unlisten}
   541:C        5             call(iod_send, io_tmp_ptr, unaddressing_char);
   542:C        5             call(bdx_proc, uep);
   543:C        5           end; {else}
   544:C        4       recover
   545:C        4         confirm_timeout_and_reset_card(uep);
   546:C        2   end; {upon_transfer_complete}
   547:S
   548:C        1 end; {discHPIB}
   549:S
```



```
   550:D        1 $page$
   551:S
   552:D        1 import
   553:D        1   loader, bkgnd;
   554:S
   555:C        1 begin {init_bkgnd}
   556:C        1   initialize_bkgnd;  {allocate temp space}
   557:C        1   markuser;          {make temp space and modules permanent}
   558:C        1 end. {init_bkgnd}
   559:S
   560:S
   561:S

No errors. No warnings.
          ***** Nonstandard language features enabled *****
```

Description

DMA_DRV provides low-level driver support.

Usage

DMA_DRV is used with the 98620A/B DMA card.

Requirements

The I/O library kernel (IODECLARATIONS, etc.)

```
    1:S                                                 (*
    2:S
    3:S           (c) Copyright Hewlett-Packard Company, 1983.
    4:S          All rights are reserved.  Copying or other
    5:S          reproduction of this program except for archival
    6:S          purposes is prohibited without the prior
    7:S          written consent of Hewlett-Packard Company.
    8:S
    9:S
   10:S                       RESTRICTED RIGHTS LEGEND
   11:S
   12:S          Use, duplication, or disclosure by the Government
   13:S          is subject to restrictions as set forth in
   14:S          paragraph (b) (3) (B) of the Rights in Technical
   15:S          Data and Computer Software clause in
   16:S          DAR 7-104.9(a).
   17:S
   18:S          HEWLETT-PACKARD COMPANY
   19:D        0 Fort Collins, Colorado                        *)
   20:S
   21:S
   22:D        0 $MODCAL ON$
   23:D        0 $PARTIAL_EVAL ON$
   24:D        0 $STACKCHECK ON$
   25:D        0 $RANGE OFF$
   26:D        0 $DEBUG OFF$
   27:D        0 $OVFLCHECK OFF$
```



```
   28:D        0 $PAGE$
   29:S
   30:D        0 {*************************************************************************}
   31:D        0 {*                                                                       *}
   32:D        0 {*      NOT RELEASED    VERSION         3.0                             *}
   33:D        0 {*                                                                       *}
   34:D        0 {*************************************************************************}
   35:D        0 {*                                                                       *}
   36:D        0 {*                                                                       *}
   37:D        0 {*      IOLIB           DMA_DRIVERS                                     *}
   38:D        0 {*                                                                       *}
   39:D        0 {*                                                                       *}
   40:D        0 {*************************************************************************}
   41:D        0 {*                                                                       *}
   42:D        0 {*                                                                       *}
   43:D        0 {*      library       -  IOLIB                                          *}
   44:D        0 {*      name          -  DMA_DRIVERS                                    *}
   45:D        0 {*      module(s)     -  init_dma                                       *}
   46:D        0 {*                                                                       *}
   47:D        0 {*      author        -                                                 *}
   48:D        0 {*      phone         -                                                 *}
   49:D        0 {*                                                                       *}
   50:D        0 {*      date          -  June  1 , 1981                                 *}
   51:D        0 {*      update        -  Aug  10 , 1983  - rev. number                  *}
   52:D        0 {*      release       -  ?????????????                                  *}
   53:D        0 {*                                                                       *}
   54:D        0 {*      source        -  IOLIB:DMA_DRV.TEXT                             *}
   55:D        0 {*      object        -  IOLIB:DMA_DRV.CODE                             *}
   56:D        0 {*                                                                       *}
   57:D        0 {*************************************************************************}
```

```
    58:D        0 $PAGE$
    59:D        0 (*************************************************************************)
    60:D        0 (*                                                                       *)
    61:D        0 (*      BUG FIX HISTORY          - after release 1.0                     *)
    62:D        0 (*                                                                       *)
    63:D        0 (*                                                                       *)
    64:D        0 (*      BUG #   BY  / ON         LOC              DESCRIPTION           *)
    65:D        0 (*      -----   ----------       ---------------  ----------------------*)
    66:D        0 (*                                                                       *)
    67:D        0 (*      367     -----------      dma_initialize   Allow eXecute of driver*)
    68:D        0 (*              09/22/82                          and have it install    *)
    69:D        0 (*                                                itself in the system.  *)
    70:D        0 (*                                                                       *)
    71:D        0 (*                                                                       *)
    72:D        0 (*************************************************************************)
```



```
    73:D        0 $PAGE$
    74:D        0 (*************************************************************************)
    75:D        0 (*                                                                       *)
    76:D        0 (*                                                                       *)
    77:D        0 (*      This is the source code for an external procedures library       *)
    78:D        0 (*      to be used for general purpose interfacing on the HP 9826.       *)
    79:D        0 (*                                                                       *)
    80:D        0 (*      The library consists of 3 primary sets of modules -              *)
    81:D        0 (*                                                                       *)
    82:D        0 (*              1.      KERNEL modules                                   *)
    83:D        0 (*              2.      driver modules                                   *)
    84:D        0 (*              3.      IOLIB  modules                                   *)
    85:D        0 (*                                                                       *)
    86:D        0 (*      The KERNEL modules consist of the following modules -            *)
    87:D        0 (*                                                                       *)
    88:D        0 (*              1.      iodeclarations  ( contains static r/w space )    *)
    89:D        0 (*              2.      iocomasm                                         *)
    90:D        0 (*              3.      general_0       ( initialization & low level     *)
    91:D        0 (*                                        routines like ioread/iowrite) *)
    92:D        0 (*      The KERNEL modules also have an executable program segement      *)
    93:D        0 (*      that gets executed at the time it is loaded.  This program       *)
    94:D        0 (*      initializes the static read/write memory.  This program also     *)
    95:D        0 (*      allocates the temporary storage for any card that exists -       *)
    96:D        0 (*      independent of whether there is or is not a driver for it.       *)
    97:D        0 (*                                                                       *)
    98:D        0 (*      The driver modules consist of the actual assembly or PASCAL      *)
    99:D        0 (*      routines that deal with a specific interface card.  There is     *)
   100:D        0 (*      also an executable program segment for each driver module.       *)
   101:D        0 (*      This program searches the select code table in the static r/w    *)
   102:D        0 (*      initialized by the KERNEL gereral_0 module for all select codes  *)
   103:D        0 (*      that have the right interface card ( HPIB drivers will search    *)
   104:D        0 (*      for the 98624 interface ).  This program will then set up the    *)
   105:D        0 (*      driver tables to point to the correct drivers.                   *)
   106:D        0 (*                                                                       *)
   107:D        0 (*      The rest of the IOLIB modules are high-level modules that are    *)
   108:D        0 (*      used by an end user in his/her application program.              *)
   109:D        0 (*                                                                       *)
   110:D        0 (*      The KERNEL and some set of driver modules will exist in the      *)
   111:D        0 (*      SYSTEM.INITLIB file as object code ( not EXPORT text ).  The     *)
   112:D        0 (*      export text will reside on the SYSTEM.LIBRARY file.  The rest    *)
   113:D        0 (*      of the library will reside on the SYSTEM.LIBRARY.                *)
   114:D        0 (*                                                                       *)
   115:D        0 (*************************************************************************)
```

```
116:D    0 $PAGE$
117:D    0 (*************************************************************************)
118:D    0 (*                                                                       *)
119:D    0 (*      REFERENCES :                                                     *)
120:D    0 (*                                                                       *)
121:D    0 (*                                                                       *)
122:D    0 (*      1.  9826 I/O Designers Guide          ( ---------- )           *)
123:D    0 (*      2.  68000 Manual                      ( Motorola )             *)
124:D    0 (*      3.  Pascal alpha site ERS             ( ---------- )           *)
125:D    0 (*      4.  Pascal I/O Library ERS            ( ---------- )           *)
126:D    0 (*      5.  9826 HPL EIO & IOD listings       ( ---------- )           *)
127:D    0 (*      6.  9826 HPL Misc. I/O Doc.           ( ---------- )           *)
128:D    0 (*      7.  9826 card documentation           ( Mfg. Specs. )          *)
129:D    0 (*      8.  Pascal I/O Library ISR            ( ---------- )           *)
130:D    0 (*                                                                       *)
131:D    0 (*                                                                       *)
132:D    0 (*************************************************************************)
```



```
133:D    0 $PAGE$
134:D    0 PROGRAM dma_initialize ( INPUT , OUTPUT );
```

```
143:D        1 $PAGE$
144:D        1 (*****************************************************************************)
145:D        1 (*                                                                           *)
146:D        1 (*                                                                           *)
147:D        1 (*      DMA DRIVERS                                                          *)
148:D        1 (*                                                                           *)
149:D        1 (*                                                                           *)
150:D        1 (*****************************************************************************)
```



```
151:D        1 $PAGE$
152:D        1 MODULE init_dma ;
153:S
154:S                  {
155:S                    date    09/23/81
156:S                    update  10/04/82
157:S
158:S                    purpose This module is a series of DMA procedures to be linked into
159:S                            the system isr poll routines. The module also contains the
160:S                            initialization code.
161:D        1           }
162:S
163:D        1 IMPORT  $SEARCH 'IOLIB:KERNEL.CODE'$
164:D        1           sysglobals, iodeclarations ;
165:S
166:D        1 EXPORT
167:S
168:S
169:D        1   PROCEDURE dma_isr_0   ( an_isrib   : PISRIB );
170:D        1   PROCEDURE dma_isr_1   ( an_isrib   : PISRIB );
171:D        1   PROCEDURE io_init_dma;
172:S
173:S
174:D        1 IMPLEMENT
175:D        1
176:S
177:D        1   IMPORT  isr,
178:D        1           general_0 ;
179:S
180:S
181:S
182:D        1   PROCEDURE dma_isr_0   ( an_isrib   : PISRIB );
183:D    -2  2   VAR io_work_word : io_word;
184:C        2   BEGIN
185:C        2     IF dma_ch_0.dummy_pr = NIL
186:C        3       THEN BEGIN
187:C        3         { no procedure for dma isr - disable channel }
188:C        3         io_work_word:=ioread_word(3,0);
189:C        3       END
190:C        3       ELSE BEGIN
191:C        3         { call the procedure with a temp ptr }
192:C        3         CALL(dma_ch_0.real_proc,isc_table[dma_isc_0].io_tmp_ptr);
193:C        3       END; { of IF }
194:C        2   END; { of dma_isr_0 }
195:S
196:S
197:S
198:S
199:D        1   PROCEDURE dma_isr_1   ( an_isrib   : PISRIB );
200:D    -2  2   VAR io_work_word : io_word;
201:C        2   BEGIN
202:C        2     IF dma_ch_1.dummy_pr = NIL
203:C        3       THEN BEGIN
204:C        3         { no procedure for dma isr - disable channel }
205:C        3         io_work_word:=ioread_word(3,8);
206:C        3       END
207:C        3       ELSE BEGIN
208:C        3         { call the procedure with a temp ptr }
209:C        3         CALL(dma_ch_1.real_proc,isc_table[dma_isc_1].io_tmp_ptr);
210:C        3       END; { of IF }
```

```
21i:C         2   END; { of dma_isr_1 }
21j:S
21k:S
214:D         1   PROCEDURE io_init_dma;
215:D     -4  2   VAR dummy       : INTEGER;
216:C         2   BEGIN
217:S
21f:C         2     io_revid := io_revid + ' D3.0';      { DMA revision added 2/5/82 TM }
21g:S
220:C         2     { initialize the DMA variables }
221:S
222:C         2     dma_ch_0.dummy_pr := NIL;
223:C         2     dma_ch_0.dummy_sl := NIL;
224:C         2     dma_isc_0         := no_isc;
225:C         2     dma_ch_1.dummy_pr := NIL;
226:C         2     dma_ch_1.dummy_sl := NIL;
227:C         2     dma_isc_1         := no_isc;
228:S
229:S
230:S
231:C         2     { see if dma really exists }
232:S
233:C         2     TRY
234:C         3       dummy:=ioread_word(3,0);
235:C         3       dma_here:=TRUE;
236:C         3       isc_table[3].card_id:=hp98620;
237:C         3     RECOVER
238:C         3       BEGIN
239:C         3         IF escapecode=-12
240:C         4           THEN BEGIN
241:C         4             dma_here := FALSE;
242:C         4           END
243:C         4           ELSE BEGIN
244:C         4             { some other problem }
245:C         4             ESCAPE(ESCAPECODE);
246:C         4           END; { of IF }
247:C         3       END; { of RECOVER BEGIN }
248:S
249:S
250:C         2     { if the card has isrs then unlink the ISR }
251:S
252:C         2     IF dma_isrib0.INTREGADDR <> NIL THEN
253:C         3         ISRUNLINK(3,                  { interrupt level       }
254:C         3                   ADDR(dma_isrib0));  { ptr to the isrib      }
255:S
256:C         2     IF dma_isrib1.INTREGADDR <> NIL THEN
257:C         3         ISRUNLINK(3,                  { interrupt level       }
258:C         3                   ADDR(dma_isrib1));  { ptr to the isrib      }
259:S
260:S
261:C         2     { if the card exists then link in an ISR for it }
262:S
263:C         2     IF dma_here
264:C         3       THEN WITH isc_table[3] DO BEGIN
265:C         4         { dma if installed is always int. level 3 }
266:C         4         PERMISRLINK(dma_isr_0,                  { channel 0 isr  }
267:C         4                     ANYPTR(INTEGER(card_ptr)+7),{ channel 0 addr }
268:C         4                     2,                          { int reg mask   }
269:C         4                     2,                          { int reg value  }
270:C         4                     3,                          { int level      }
```



```
271:C         4                     ADDR(dma_isrib0));          { isrib 0 info   }
272:C         4         PERMISRLINK(dma_isr_1,                  { channel 1 isr  }
273:C         4                     ANYPTR(INTEGER(card_ptr)+15),{ channel 1 addr }
274:C         4                     2,                          { int reg mask   }
275:C         4                     2,                          { int reg value  }
276:C         4                     3,                          { int level      }
277:C         4                     ADDR(dma_isrib1));          { isrib 1 info   }
278:C         4       END; { of IF }
279:S
280:C         2   END; { of io_init_dma }
281:S
282:C         1 END; { of MODULE init_dma }
```

```
283:D        1 $PAGE$
284:S
285:D        1 IMPORT    init_dma ,
286:D        1           LOADER ;                                    { 367 TM 9/22/82 }
287:S
288:S
289:C        1 BEGIN
290:C        1   io_init_dma;
291:C        1   MARKUSER;                                           { 367 TM 9/22/82 }
292:C        1 END. { of dma_initialize }

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

EDRIVER

Description

EDRIVER contains a utility library for driving the EPROM programming card.

Requirements

SYSGLOBALS, IODECLARATIONS, and ASM.

Pascal [Rev 3.0M 6/ 4/84] EDRIVER.TEXT   26-Dec-84 22:51:53 Page 1   *** File name: EDRIVER ***

```
                                                      (*

              (c) Copyright Hewlett-Packard Company, 1984.
             All rights are reserved.  Copying or other
             reproduction of this program except for archival
             purposes is prohibited without the prior
             written consent of Hewlett-Packard Company.


                              RESTRICTED RIGHTS LEGEND

             Use, duplication, or disclosure by the Government
             is subject to restrictions as set forth in
             paragraph (b) (3) (B) of the Rights in Technical
             Data and Computer Software clause in
             DAR 7-104.9(a).

             HEWLETT-PACKARD COMPANY
           0 Fort Collins, Colorado                           *)

           0 $SYSPROG$
           0 $RANGE OFF$  $DEBUG OFF$
           0 MODULE EDRIVER;
           1 IMPORT SYSGLOBALS,IODECLARATIONS,ASM;
           1 EXPORT
           1   TYPE
           1     EPINFOREC = RECORD
           1                   BFAST   : BOOLEAN;
           1                   EPSTART : INTEGER;
           1                   EPEND   : INTEGER;
           1                 END;

           1     EOPTYPE = (EREAD,EWRITE,ECWRITE,EBLANK,ECHECK);
           1     EPERROR = (ENOERROR,ENOTPROG,ENOEPROM,ECFAIL,EPFAIL,ENOTBLANK,EBADARGS);

           1     FUNCTION EPROG(        SCODE    : INTEGER;
           2                            OP       : EOPTYPE;
           2                     ANYVAR BUFFER   : WINDOW;
           2                            BUFSIZE  : INTEGER;
           2                        VAR POSIT    : INTEGER):EPERROR;

           1     FUNCTION EGETINFO(    SCODE : INTEGER;
           2                       VAR INFO  : EPINFOREC):EPERROR;

           1     FUNCTION EINIT(SCODE : INTEGER):EPERROR;

           1     FUNCTION EBRATE(SCODE : INTEGER;
           2                     BRATE : BOOLEAN):EPERROR;

           1 IMPLEMENT
           1   TYPE
           1     EPOPTYPE = (EPINIT,EPREAD,EPWRITE,EPCWRITE,EPBLANK,EPCHECK,EPBRATE,EPINFO);
           1     STATREG= PACKED RECORD
           1                 B4  : BYTE;  { PUT HERE TO MAKE THE COMPILER HAPPY }
           1                 PAD : 0..15; { 7,6,5,4 }
           1                 FAST: BOOLEAN;
           1                 ACCS: BOOLEAN;
           1                 ENAB: BOOLEAN;
           1                 BUSY: BOOLEAN;
           1               END;
```

Pascal [Rev 3.0M 6/ 4/84] EDRIVER.TEXT   26-Dec-84 22:51:53 Page 2   *** File name: EDRIVER ***

```
           1     EPTYPE = PACKED RECORD
           1                 B0 : BYTE;
           1                 ID : BYTE;
           1                 B2 : BYTE;
           1                 B3 : BYTE;
           1                 CASE BOOLEAN OF
           1                 TRUE:(CONTROL:STATREG);
           1                 FALSE:(STATUS:STATREG);
           1               END;
           1     EPTR = ^EPTYPE;
           1     INFOPTR = ^EPINFOREC;


           1   PROCEDURE EPPROG(        SCODE    : INTEGER;
           2                            OP       : EPOPTYPE;
           2                     ANYVAR BUFFER   : WINDOW;
           2                            BUFSIZE  : INTEGER;
           2                        VAR POSIT    : INTEGER;
           2                        VAR ECODE    : EPERROR);
           2   CONST
           2     LO_ROM = HEX('20000');  { 128k }
           2     HI_ROM = HEX('200000'); { 2048k }
           2     STEPSIZE = HEX('20000');{ 128k }
           2   TYPE
           2     TWOBYTES = PACKED ARRAY[0..1] OF CHAR;
           2     BREC = RECORD
           2              CASE INTEGER OF
           2              0:  (INT:INTEGER);
           2              1:  (BPTR:^BYTE);
           2              2:  (WPTR:^TWOBYTES);
           2              3:  (CPTR:^CHAR);
           2              4:  (WINPTR:WINDOWP);
           2            END;

           2   VAR
       -4  2     CARD         : EPTR;
       -8  2     PBYTE        : BREC;
      -12  2     DUMMY        : INTEGER;
      -13  2     DONE         : BOOLEAN;
      -18  2     ENDSCAN      : INTEGER;
      -22  2     I            : INTEGER;
      -24  2     WBYTES       : TWOBYTES;
      -28  2     INFO         : INFOPTR;

           2     PROCEDURE BADIO(ECCODE:EPERROR);
           3     BEGIN
           3       ECODE := ECCODE; ESCAPE(0);
           3     END;

           2     PROCEDURE SETUP;
           3     BEGIN
           3       WITH INFO^ DO
           4       BEGIN
           4         IF EPSTART=0 THEN BADIO(ENOEPROM);
           4         ENDSCAN := POSIT + BUFSIZE;
           4         IF (POSIT<EPSTART) OR (ENDSCAN>EPEND) THEN BADIO(EBADARGS);
           4         PBYTE.INT := POSIT;
           4         DONE := FALSE;
           4       END;
           3     END;
```

```
121:S
122:D        2     PROCEDURE GETINFO(ANYVAR UINFO : EPINFOREC);
123:C        3     BEGIN
124:C        3       UINFO := INFO^;
125:C        3     END;
126:S
127:C        2   BEGIN { EPPROG }
128:C        2     ECODE := ENOERROR;
129:C        2     TRY
130:C        3       { VALIDATE THE SELECT CODE }
131:C        3       IF (SCODE<IOMINISC) OR (SCODE>IOMAXISC) THEN BADIO(ENOTPROG);
132:C        3       WITH ISC_TABLE[SCODE] DO
133:C        4       BEGIN
134:C        4         CARD := CARD_PTR;
135:C        4         IF (CARD=NIL) OR (CARD_TYPE=0) THEN BADIO(ENOTPROG);
136:C        4         IF ((CARD_TYPE=1) AND (CARD_ID=0)) OR
137:C        5            ((CARD_TYPE=9) AND (CARD_ID=27)) THEN
138:C        5         BEGIN
139:C        5           IF CARD^.ID<>27 THEN BADIO(ENOTPROG);
140:C        5         END
141:C        5         ELSE BADIO(ENOTPROG);
142:C        4         INFO := ADDR(IO_TMP_PTR^.DRV_MISC);
143:C        4       END;
144:C        3       CASE OP OF
145:C        4       EPINIT:   { INITIALIZE THE CONTROL RECORD }
146:C        4         BEGIN   { VALIDATE THE CARD ID }
147:C        4           WITH ISC_TABLE[SCODE] , INFO^ DO
148:C        5           BEGIN
149:C        5             IF (CARD_TYPE=1) AND (CARD_ID=0) THEN
150:C        6             BEGIN  { FIX CARD TYPE AND ID }
151:C        6               CARD_TYPE := 9; CARD_ID := 27;
152:C        6             END;
153:C        5             WITH CARD^ DO
154:C        6             BEGIN
155:C        6               ID := 0;  { RESET THE CARD }
156:C        6               { INITIALIZE THE CONTROL RECORD }
157:C        6               BFAST := STATUS.FAST;     { SAVE PROGRAMMING SPEED }
158:C        6               EPSTART :=0 ;             {ADDRESS OF ATTACHED EPROM CARD}
159:C        6               EPEND :=0 ;               {ADDRESS OF LAST EPROM BYTE + 1}
160:C        6               { FINDOUT WHICH EPROM CARD IS ATTACHED TO THE PROGRAMMER CARD }
161:C        6               PBYTE.INT := LO_ROM;
162:C        6               DONE := FALSE;
163:C        6               CACHE_OFF;                          { 3.0 BUG FIX -- 3/13/84}
164:C        6               REPEAT
165:C        7                 TRY
166:C        8                   DUMMY := PBYTE.BPTR^; { ACCESS THE EPROM SPACE }
167:C        8                   IF STATUS.ACCS THEN
168:C        9                   BEGIN  { FOUND THE EPROM CARD }
169:C        9                     DONE := TRUE;
170:C        9                     EPSTART := PBYTE.INT;
171:C        9                     EPEND :=  EPSTART + STEPSIZE;
172:C        9                     PBYTE.INT := EPEND;
173:C        9                     { CHECK FOR xx64 PARTS OR xx128 PARTS }
174:C        9                     IF EPEND<HI_ROM THEN        { DON'T PASS END OF ROM }
175:C       10                     BEGIN
176:C       10                       CONTROL.ACCS := FALSE;    { CLEAR THE ACCESS BIT }
177:C       10                       DUMMY := PBYTE.BPTR^;     { ACCESS THE EPROM SPACE }
178:C       10                       IF STATUS.ACCS THEN EPEND :=  EPEND + STEPSIZE;
179:C       10                     END;
180:C        9                   END
```



```
181:C        9                   ELSE
182:C        9                     PBYTE.INT := PBYTE.INT + STEPSIZE;
183:C        8                 RECOVER { IGNORE BUS ERRORS }
184:C        8                   IF ESCAPECODE<>-12 THEN ESCAPE(ESCAPECODE)
185:C        9                   ELSE
186:C        9                     PBYTE.INT := PBYTE.INT + STEPSIZE;
187:C        7                 DONE:=DONE OR (PBYTE.INT>=HI_ROM);
188:C        7               UNTIL DONE;
189:C        6               CACHE_ON;                           { 3.0 BUG FIX 3/13/84 }
190:C        6               IF EPSTART=0 THEN BADIO(ENOEPROM);
191:C        6             END;
192:C        5           END;
193:C        4         END;
194:S
195:C        4       EPREAD:   { MOVE DATA FROM EPROM SPACE TO BUFFER }
196:C        4         IF BUFSIZE>0 THEN
197:C        5         BEGIN
198:C        5           SETUP;
199:C        5           MOVELEFT(PBYTE.CPTR^,BUFFER[0],BUFSIZE);
200:C        5           POSIT := ENDSCAN;
201:C        5         END;
202:S
203:C        4       EPWRITE,EPCWRITE: { MOVE DATA FROM BUFFER TO EPROM }
204:C        4         IF BUFSIZE>0 THEN
205:C        5         BEGIN
206:C        5           SETUP;
207:C        5           WITH CARD^, INFO^ DO
208:C        6           BEGIN
209:C        6             TRY
210:C        7               CONTROL.FAST := BFAST;    { SET BURN SPEED }
211:C        7               CONTROL.ENAB := TRUE;     { ENABLE WRITE }
212:C        7               I := 0;
213:C        7               IF ODD(PBYTE.INT) THEN
214:C        8               BEGIN     { BURN FIRST BYTE }
215:C        8                 IF PBYTE.CPTR^<>BUFFER[I] THEN
216:C        9                 BEGIN   { PATTERN NOT SAME SO BURN IT }
217:C        9                   PBYTE.CPTR^ := BUFFER[I];
218:C        9                   WHILE CONTROL.BUSY DO;  { WAIT FOR BURN TO FINISH }
219:C        9                   IF OP = EPCWRITE THEN   { CHECK PATTERN AFTER BURN }
220:C       10                     IF PBYTE.CPTR^<>BUFFER[I] THEN BADIO(EPFAIL);
221:C        9                 END;
222:C        8                 I := I + 1;
223:C        8                 PBYTE.INT := PBYTE.INT + 1;
224:C        8               END;
225:C        7               ENDSCAN := BUFSIZE-1;
226:C        7               WHILE I<BUFSIZE DO
227:C        8               BEGIN     { BURN 2 BYTES AT A TIME EXECPT FOR LAST ODD BYTE }
228:C        8                 IF I=ENDSCAN THEN
229:C        9                 BEGIN   { BURN LAST BYTE }
230:C        9                   IF PBYTE.CPTR^<>BUFFER[I] THEN
231:C       10                   BEGIN { PATTERN NOT SAME SO BURN IT }
232:C       10                     PBYTE.CPTR^ := BUFFER[I];
233:C       10                     WHILE CONTROL.BUSY DO;  { WAIT FOR BURN TO FINISH }
234:C       10                     IF OP = EPCWRITE THEN   { CHECK PATTERN AFTER BURN }
235:C       11                       IF PBYTE.CPTR^<>BUFFER[I] THEN BADIO(EPFAIL);
236:C       10                   END;
237:C        9                   I := I + 1;
238:C        9                   PBYTE.INT := PBYTE.INT + 1;
239:C        9                 END
240:C        9                 ELSE
```

```
      9                    BEGIN   { BURN TWO BYTES }
      9                      WBYTES[0] := BUFFER[I];
      9                      WBYTES[1] := BUFFER[I+1];
      9                      IF PBYTE.WPTR^<>WBYTES THEN
     10                      BEGIN { PATTERN NOT SAME SO BURN IT }
     10                        PBYTE.WPTR^ := WBYTES;  { BURN BOTH BYTES }
     10                        WHILE CONTROL.BUSY DO;  { WAIT FOR BURN TO FINISH }
     10                        IF OP=EPCWRITE THEN      { CHECK PATTERN AFTER BURN }
     11                          IF PBYTE.WPTR^<>WBYTES THEN BADIO(EPFAIL);
     10                      END;
      9                      I := I + 2;
      9                      PBYTE.INT := PBYTE.INT + 2;
      9                    END;       { BURN TWO BYTES }
      8                  END;         { WHILE I<BUFSIZE}
      7                  CONTROL.ENAB:=FALSE;     { TURN OFF WRITE ENABLE }
      7                  POSIT := PBYTE.INT;      { RECORD THE POSITION }
      7                RECOVER
      7                  BEGIN
      7                    CONTROL.ENAB:=FALSE;     { TURN OFF WRITE ENABLE }
      7                    POSIT := PBYTE.INT;      { RECORD THE POSITION }
      7                    IF ESCAPECODE<>0 THEN ESCAPE(ESCAPECODE);
      7                  END;
      6              END;          { WITH CARD^, INFO^ }
      5            END;

      4          EPBLANK:  { CHECK THE EPROM SPACE FOR HEX FF }
      4            IF BUFSIZE>0 THEN
      5            BEGIN
      5              SETUP;
      5              REPEAT
      6                IF PBYTE.WPTR^[0]<>CHR(255) THEN DONE := TRUE
      7                ELSE PBYTE.INT := PBYTE.INT + 1;
      6                DONE := DONE OR (PBYTE.INT>=ENDSCAN);
      6              UNTIL DONE;
      5              POSIT := PBYTE.INT;
      5              IF POSIT<ENDSCAN THEN BADIO(ENOTBLANK);
      5            END;

      4          EPCHECK:  { CHECK DATA FOR POSSIBLE BURN }
      4            IF BUFSIZE>0 THEN
      5            BEGIN
      5              SETUP;
      5              I := 0;
      5              REPEAT
      6                IF IOR(ORD(PBYTE.WINPTR^[I]),ORD(BUFFER[I]))=ORD(PBYTE.WINPTR^[I])
      7                THEN I := I + 1
      7                ELSE DONE := TRUE;
      6                DONE := DONE OR (I>=BUFSIZE);
      6              UNTIL DONE;
      5              POSIT := PBYTE.INT + I;
      5              IF I<BUFSIZE THEN BADIO(ECFAIL);
      5            END;

      4          EPBRATE: INFO^.BFAST := POSIT<>0; { SET BURN RATE }

      4          EPINFO: GETINFO(BUFFER);   { GET BURN RATE AND ADDRESS INFO }

      4          END; { CASE }
      3        RECOVER
      3        BEGIN
```



```
      3          IF ESCAPECODE<>0 THEN ESCAPE(ESCAPECODE);
      3        END;
      2      END; { EPPROG }


      1      FUNCTION EPROG(       SCODE    : INTEGER;
      2                            OP       : EOPTYPE;
      2                     ANYVAR BUFFER   : WINDOW;
      2                            BUFSIZE  : INTEGER;
      2                        VAR POSIT    : INTEGER):EPERROR;
      2      VAR
  -2  2        EOP : EPOPTYPE;
  -4  2        ECODE : EPERROR;
      2      BEGIN
      2        CASE OP OF
      3          EREAD  : EOP:=EPREAD;
      3          EWRITE : EOP:=EPWRITE;
      3          ECWRITE: EOP:=EPCWRITE;
      3          EBLANK : EOP:=EPBLANK;
      3          ECHECK : EOP:=EPCHECK;
      3        END;
      2        EPPROG(SCODE,EOP,BUFFER,BUFSIZE,POSIT,ECODE); EPROG:=ECODE;
      2      END;

      1      FUNCTION EGETINFO(    SCODE : INTEGER;
      2                        VAR INFO : EPINFOREC):EPERROR;
      2      VAR
  -4  2        I : INTEGER;
  -6  2        ECODE : EPERROR;
      2      BEGIN
      2        EPPROG(SCODE,EPINFO,INFO,0,I,ECODE); EGETINFO:=ECODE;
      2      END;

      1      FUNCTION EBRATE(SCODE : INTEGER;BRATE : BOOLEAN):EPERROR;
      2      VAR
  -4  2        I : INTEGER;
  -6  2        ECODE : EPERROR;
      2      BEGIN
      2        I := ORD(BRATE);
      2        EPPROG(SCODE,EPBRATE,I,0,I,ECODE); EBRATE:=ECODE;
      2      END;

      1      FUNCTION EINIT(SCODE : INTEGER):EPERROR;
      2      VAR
  -4  2        I : INTEGER;
  -6  2        ECODE : EPERROR;
      2      BEGIN
      2        EPPROG(SCODE,EPINIT,I,0,I,ECODE); EINIT:=ECODE;
      2      END;

      1 END.
```

No errors. No warrings.
***** Nonstandard language features enabled *****

EPROMS

Description

EPROMS contains the TM (transfer method) for EPROMs.

Requirements

SYSGLOBALS and ASM.

```
 1:D        0 $SYSPROG$
 2:D        0 $DEBUG OFF$      $RANGE OFF$
 3:D        0 PROGRAM INST_EPROM;
 4:D        1 MODULE EPROMS;
 5:D        1 IMPORT
 6:D        1   SYSGLOBALS,ASM;
 7:D        1 EXPORT
 8:D        1   PROCEDURE EPROM_TM(FP:FIBP; REQUEST:AMREQUESTTYPE;
 9:D        1             ANYVAR BUFFER:WINDOW; BUFSIZE,POSITION: INTEGER);
10:D        1 IMPLEMENT
11:S
12:D        1   PROCEDURE EPROM_TM(FP:FIBP; REQUEST:AMREQUESTTYPE;
13:D        2             ANYVAR BUFFER:WINDOW; BUFSIZE,POSITION: INTEGER);
14:D        2   CONST
15:D        2     LO_ROM = HEX('20000');
16:D        2     HI_ROM = HEX('200000');
17:D        2     STEPSIZE = HEX('4000');
18:D        2     ROMHEADER= HEX('F0FF');
19:D        2     HEADERSIZE = 18;
20:D        2     K128 = 131072;
21:D        2     K256 = 262144;
22:D        2     K16  = 16384;
23:D        2   TYPE
24:D        2     HRECORD = PACKED RECORD
25:D        2                 HEADER : 0..65535;
26:D        2                 IDCHAR : BYTE;
27:D        2                 FLAG   : BYTE;
28:D        2               END;
29:S
30:D        2     HPTR = ^HRECORD;
31:D        2     SCANREC = RECORD
32:D        2                 CASE INTEGER OF
33:D        2                 0 :(INT:INTEGER);
34:D        2                 1 :(PTR:HPTR);
35:D        2                 2 :(WPTR:WINDOWP);
36:D        2                 3 :(CPTR:^CHAR);
37:D        2               END;
38:D        2   VAR
39:D    -4  2     SCANPTR : SCANREC;
40:D    -8  2     COUNTER : INTEGER;
41:D    -9  2     DONE    : BOOLEAN;
42:D   -14  2     SIZE    : INTEGER;
43:S
44:C        2   BEGIN
45:C        2     IORESULT := ORD(INOERROR);
46:C        2     IF (UNITABLE^[FP^.FUNIT].BYTEOFFSET=0) AND
47:C        3        (REQUEST<>CLEARUNIT) THEN IORESULT:=ORD(ZNODEVICE)
48:C        3       ELSE
49:C        3       CASE REQUEST OF
50:C        4       READBYTES:     { MOVE DATA FROM EPROM TO BUFFER }
51:C        4         IF BUFSIZE>0 THEN
52:C        5         WITH FP^ DO
53:C        6         BEGIN
54:C        6           IF (POSITION+BUFSIZE)>FPEOF THEN IORESULT := ORD(ZNOSUCHBLK)
55:C        7           ELSE
56:C        7           BEGIN
57:C        7             COUNTER := HEADERSIZE + FILEID + POSITION;
58:C        7             COUNTER := COUNTER+(2*(COUNTER DIV K16))+
59:C        7                        UNITABLE^[FUNIT].BYTEOFFSET;
60:C        7             SCANPTR.INT := COUNTER;
```



```
61:C        7             IF BUFSIZE=1 THEN BUFFER[0] := SCANPTR.CPTR^
62:C        8             ELSE
63:C        8             BEGIN { MOVEING MORE THAN ONE BYTE OF DATA }
64:C        8               COUNTER := 0;
65:C        8               REPEAT
66:C        9                 SIZE := (SCANPTR.INT + K16) DIV K16 * K16 - SCANPTR.INT;
67:C        9                 IF SIZE>BUFSIZE THEN SIZE := BUFSIZE;
68:C        9                 MOVELEFT(SCANPTR.CPTR^,BUFFER[COUNTER],SIZE);
69:C        9                 BUFSIZE := BUFSIZE - SIZE;
70:C        9                 COUNTER := COUNTER + SIZE;
71:C        9                 SCANPTR.INT := SCANPTR.INT + SIZE + 2;
72:C        9               UNTIL BUFSIZE=0;
73:C        8             END;
74:C        7           END;
75:C        6         END;      { READBYTES }
76:C        4       WRITEBYTES: IORESULT := ORD(ZPROTECTED);
77:C        4       FLUSH:;
78:C        4       CLEARUNIT:
79:C        4         BEGIN     { FIND THE nTH DISC HEADER }
80:C        4           SCANPTR.INT := LO_ROM;
81:C        4           UNITABLE^[FP^.FUNIT].BYTEOFFSET := 0; { CLEAR THE OFFSET }
82:C        4           COUNTER := UNITABLE^[FP^.FUNIT].DV;
83:C        4           DONE := FALSE;
84:C        4           REPEAT  { FIND THE EPROM DISC HEADER }
85:C        5             TRY
86:C        6               IF SCANPTR.PTR^.HEADER=ROMHEADER THEN
87:C        7               BEGIN
88:C        7                 IF (SCANPTR.PTR^.FLAG=HEX('08')) OR
89:C        8                    (SCANPTR.PTR^.FLAG=HEX('18')) THEN
90:C        8                 BEGIN     { FOUND DISC HEADER }
91:C        8                   COUNTER := COUNTER - 1; { COUNT IT }
92:C        8                   DONE := COUNTER<0;      { IS THIS THE ONE }
93:C        8                 END;
94:C        7               END;
95:C        6               IF NOT DONE THEN SCANPTR.INT := SCANPTR.INT + STEPSIZE;
96:C        6             RECOVER     { IGNORE BUS ERRORS }
97:C        6               IF ESCAPECODE<>-12 THEN ESCAPE(ESCAPECODE)
98:C        7               ELSE SCANPTR.INT := SCANPTR.INT + STEPSIZE;
99:S
100:C       5             DONE := DONE OR (SCANPTR.INT>=HI_ROM);
101:C       5           UNTIL DONE;
102:C       4           IF SCANPTR.INT>=HI_ROM THEN IORESULT := ORD(ZNODEVICE)
103:C       5           ELSE
104:C       5           WITH UNITABLE^[FP^.FUNIT] DO
105:C       6           BEGIN
106:C       6             BYTEOFFSET := SCANPTR.INT;
107:C       6             UMAXBYTES := (SCANPTR.INT+K256) DIV K256 * K256 - SCANPTR.INT;
108:C       6           END;
109:C       4         END;      { CLEARUNIT }
110:C       4       OTHERWISE
111:C       4         IORESULT := ORD(ZBADMODE);
112:C       4       END; { CASE REQUEST }
113:C       2   END;
114:C       1 END;    { MODULE EPROM_MODULE }
115:S
116:D       1 { EPROMS INSTALLATION PROGRAM }
117:D       1 IMPORT LOADER;
118:C       1 BEGIN
119:C       1   MARKUSER;
120:C       1 END.
```

```
No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

ETU

Description

ETU contains an interactive program for controlling the EPROM programming card.

Requirements

SYSGLOBALS, MISC, ASM, FS, CI, LOADER, SYSDEVS, LDR, and EDRIVER.

```
Pascal [Rev 3.0M 6/ 4/84] ETU.TEXT                26-Dec-84 23:04:01 Page 1    *** File name: ETU ***

                                                      (*

           (c) Copyright Hewlett-Packard Company, 1983.
          All rights are reserved.  Copying or other
          reproduction of this program except for archival
          purposes is prohibited without the prior
          written consent of Hewlett-Packard Company.


                         RESTRICTED RIGHTS LEGEND

          Use, duplication, or disclosure by the Government
          is subject to restrictions as set forth in
          paragraph (b) (3) (B) of the Rights in Technical
          Data and Computer Software clause in
          DAR 7-104.9(a).

          HEWLETT-PACKARD COMPANY
        0 Fort Collins, Colorado                          *)

        0 $COPYRIGHT 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'$
        0 $SYSPROG$ $DEBUG OFF$ $RANGE OFF$

        0 PROGRAM ETU(KEYBOARD,INPUT,OUTPUT);
        1 $SEARCH 'LIBR:EDRIVER'$
        1 IMPORT SYSGLOBALS,MISC,FS,CI,EDRIVER,ASM,SYSDEVS;

        1 VAR
        1   KEYBOARD        : TEXT;

        1 (**********************************************************************)
        1 PROCEDURE COMMANDLEVEL;
        2 CONST
        2   SH_EXC      = CHR(27);
        2   K16         = 16384;
        2   MINSC       = 8;
        2   MAXSC       = 31;

        2 TYPE
        2   PROMPTTYPE = STRING80;
        2   BUFTYPE    = PACKED ARRAY[0..MAXINT] OF CHAR;
        2   BIGPTR     = ^BUFTYPE;
        2   PASSTYPE   = (CHECK,BURN,VERIFY);
        2 TYPE
        2   EPROMPTR   = ^EPROMREC;
        2   EPROMREC   = RECORD
        2                  NEXT      : EPROMPTR;
        2                  BASEADDR  : INTEGER;
        2                  EPSIZE    : INTEGER;
        2                  EPINC     : INTEGER;
        2                  PRESENT   : ARRAY[0..7] OF BOOLEAN;
        2                  ADDRESS   : ARRAY[0..7] OF INTEGER;
        2                END;
        2   DIRSTATUS  = (DNEEDED,DWANTED,DONTCARE);
        2   CONTROL    = RECORD
        2                  CFIB      : FIB;
        2                  PATH      : INTEGER;
        2                  DIROPEN   : BOOLEAN;
        2                  FILEOPEN  : BOOLEAN;
        2                  USEUNIT   : BOOLEAN;
```

```
Pascal [Rev 3.0M 6/ 4/84] ETU.TEXT                26-Dec-84 23:04:01 Page 2    *** File name: ETU ***

        2                  MOUNTED   : BOOLEAN;
        2                  CPVOL     : VID;
        2                  CVOL      : VID;
        2                  CFILE     : FID;
        2                  DSTATUS   : DIRSTATUS;
        2                END;

        2 VAR
   -4   2   SCODE : INTEGER;
   -5   2   OP    : CHAR;
   -6   2   FASTBURN        : BOOLEAN;
  -10   2   TEMP            : INTEGER;
  -20   2   EPINFO          : EPINFOREC;
  -22   2   ERROR           : EPERROR;

  -23   2   HEAPINUSE       : BOOLEAN;

 -854   2   ININFO          : CONTROL;

 -858   2   SAVEIO          : INTEGER;
 -862   2   SAVEESC         : INTEGER;
 -866   2   LHEAP           : ANYPTR;
 -870   2   EPROMLIST       : EPROMPTR;
 -874   2   EPROMDATA       : EPROMPTR;
 -878   2   LEFTTOXFER      : INTEGER;
 -882   2   OUTPOSITION     : INTEGER;
 -886   2   OUTSTARTA       : INTEGER;
 -890   2   PASS            : INTEGER;

 -890   2 (**********************************************************************)
        2 PROCEDURE FIXLOCK;
        3 BEGIN
        3   IF LOCKLEVEL<>0 THEN
        4   BEGIN LOCKLEVEL := 1; LOCKDOWN; END;
        3 END;     { FIXLOCK }

 -890   2 (**********************************************************************)
        2 PROCEDURE PRINTIOERRMSG;
        3 VAR
  -82   3   MSG    : STRING[80];
        3 BEGIN
        3   IF IORESULT<>ORD(INOERROR) THEN
        4   BEGIN
        4     GETIOERRMSG(MSG,IORESULT);
        4     WRITELN('Error: ',MSG,CTEOL);
        4     IF STREAMING THEN ESCAPE(-1);
        4   END;
        3 END;     { PRINTIOERRMSG }

 -890   2 (**********************************************************************)
  -82   2 PROCEDURE SHOWPROMPT(P : PROMPTTYPE);
        3 BEGIN WRITE(HOMECHAR,P,CTEOL); END;

 -890   2 (**********************************************************************)
        2 PROCEDURE GOODIO;
        3 BEGIN IF IORESULT<>ORD(INOERROR) THEN ESCAPE(0); END;

 -890   2 (**********************************************************************)
        2 PROCEDURE BADIO(IOCODE : IORSLTWD);
        3 BEGIN IORESULT := ORD(IOCODE); ESCAPE(0); END;
```

```
   121:S
   122:D  -890  2 (*****************************************************************************)
   123:D   -82  2 PROCEDURE BADMESSAGE(P : PROMPTTYPE);
   124:C       3 BEGIN
   125:C       3   WRITELN(P,CTEOL);
   126:C       3   IF STREAMING THEN ESCAPE(-1) ELSE BADIO(INOERROR);
   127:C       3 END;    { BADMESSAGE }
   128:S
   129:D  -890  2 (*****************************************************************************)
   130:D       2 PROCEDURE BADCOMMAND(C:CHAR);
   131:C       3 BEGIN
   132:C       3   WRITELN('BAD COMMAND ''',C,'''',CTEOL);
   133:C       3   IF STREAMING THEN ESCAPE(-1) ELSE BADIO(INOERROR);
   134:C       3 END;    { BADCOMMAND }
   135:S
   136:D  -890  2 (*****************************************************************************)
   137:D       2 PROCEDURE READCHECK;
   138:C       3 BEGIN
   139:C       3   IF IORESULT<>ORD(INOERROR) THEN
   140:C       4   BEGIN
   141:C       4     SAVEIO := IORESULT; WRITELN; IORESULT := SAVEIO;
   142:C       4     ESCAPE(0);
   143:C       4   END;
   144:C       3 END;    { READCHECK }
   145:S
   146:D  -890  2 (*****************************************************************************)
   147:D       2 PROCEDURE READNUMBER(VAR INT : INTEGER);
   148:D       3 VAR
   149:D    -4 3   I        : INTEGER;
   150:D    -8 3   TI       : INTEGER;
   151:D   -30 3   INSTRING : STRING[20];
   152:C       3 BEGIN
   153:C       3   READLN(INSTRING);  GOODIO;
   154:C       3   INSTRING:=STRLTRIM(INSTRING);
   155:C       3   IF STRLEN(INSTRING)>0 THEN
   156:C       4   TRY
   157:C       5     IF INSTRING[1]=SH_EXC THEN ESCAPE(0);
   158:C       5     ZAPSPACES(INSTRING);
   159:C       5     IF STRLEN(INSTRING)>0 THEN
   160:C       6     BEGIN
   161:C       6       TI  := 0;
   162:C       6       FOR I:=1 TO STRLEN(INSTRING) DO
   163:C       7         IF (INSTRING[I]<'0') OR (INSTRING[I]>'9') THEN BADIO(IBADVALUE)
   164:C       8         ELSE TI := TI * 10 + (ORD(INSTRING[I]) - ORD('0'));
   165:C       6       INT := TI;
   166:C       6     END;
   167:C       5   RECOVER
   168:C       5     IF ESCAPECODE=-4 THEN BADIO(IBADVALUE)
   169:C       6                      ELSE ESCAPE(ESCAPECODE);
   170:C       3 END;    { READNUMBER }
   171:S
   172:D  -890  2 (*****************************************************************************)
   173:D       2 FUNCTION UNITNUMBER(VAR FVID : VID):BOOLEAN;
   174:D       3 LABEL 1;
   175:D       3 VAR
   176:D    -8 3   SL,I : INTEGER;
   177:C       3 BEGIN
   178:C       3   UNITNUMBER := FALSE;
   179:C       3   SL := STRLEN(FVID);
   180:C       3   IF SL<2 THEN GOTO 1;
```



```
   181:C       3   IF FVID[1]<>'#' THEN GOTO 1;
   182:C       3   FOR I:=2 TO SL DO IF (FVID[I]<'0') OR (FVID[I]>'9') THEN GOTO 1;
   183:C       3   UNITNUMBER := TRUE;
   184:C       3 1:END;
   185:D  -890  2       { UNITNUMBER }
   186:D  -890  2 (*****************************************************************************)
   187:D       2 PROCEDURE UPCCHAR(VAR CH : CHAR);
   188:C       3 BEGIN IF ('a'<=CH) AND (CH<='z') THEN CH:=CHR(ORD(CH)-32); END;
   189:S
   190:D  -890  2 (*****************************************************************************)
   191:D       2 PROCEDURE PROMPTREAD(P:PROMPTTYPE; VAR ANSWER:CHAR; LIST:PROMPTTYPE;
   192:D  -164  3                         DEFAULT:CHAR);
   193:D  -164  3 LABEL 1;
   194:D  -164  3 VAR
   195:D  -168  3   I    : INTEGER;
   196:C       3 BEGIN
   197:C       3   IF (DEFAULT<>SH_EXC) AND STREAMING THEN ANSWER:=DEFAULT
   198:C       4   ELSE
   199:C       4   BEGIN
   200:C       4     WRITE(P,CTEOL);
   201:C       4     REPEAT
   202:C       5       READ(KEYBOARD,ANSWER); READCHECK; UPCCHAR(ANSWER);
   203:C       5       IF ANSWER=SH_EXC THEN  BEGIN WRITELN; BADIO(INOERROR); END;
   204:C       5       FOR I:=1 TO STRLEN(LIST) DO  { IS CHARACTER IN THE LIST ? }
   205:C       6         IF ANSWER=LIST[I] THEN GOTO 1;
   206:C       5       IF STREAMING THEN BADCOMMAND(ANSWER);
   207:C       5     UNTIL FALSE;
   208:C       4    1:WRITELN(ANSWER);
   209:C       4   END;
   210:C       3 END;    { PROMPTREAD }
   211:S
   212:D  -890  2 (*****************************************************************************)
   213:D   -82  2 PROCEDURE PROMPTYORN(P : PROMPTTYPE; VAR ANSWER :CHAR);
   214:C       3 BEGIN
   215:C       3   PROMPTREAD(P+' ? (Y/N) ',ANSWER,'YN','Y');
   216:C       3 END;    { PROMPTYORN }
   217:S
   218:D  -890  2 (*****************************************************************************)
   219:D   -82  2 PROCEDURE MOUNTVOLUME(SD : PROMPTTYPE ;VAR FINFO : CONTROL);
   220:D   -82  3 VAR
   221:D   -83  3   ANSWER        : CHAR;
   222:D   -88  3   UNIT          : INTEGER;
   223:D  -106  3   TEMPNAME      : VID;
   224:S
   225:C       3 BEGIN
   226:C       3   WITH FINFO DO
   227:C       4   BEGIN
   228:C       4     IF STREAMING THEN
   229:C       5     BEGIN
   230:C       5       WRITELN('VOLUME ',CPVOL,' NOT ONLINE WHILE STREAMING',CTEOL);
   231:C       5       ESCAPE(-1);
   232:C       5     END;
   233:S
   234:C       4     TEMPNAME := CPVOL;
   235:C       4     UNIT     := FINDVOLUME(TEMPNAME,FALSE); { CHECK FOR BAD UNIT # }
   236:C       4     IORESULT := ORD(INOERROR);
   237:S
   238:C       4     REPEAT
   239:C       5       { CONSTRUCT THE PROMPT }
   240:C       5       WRITE('Please mount',SD);
```

```
241:C        5       IF STRLEN(CVOL)>0 THEN WRITE(' volume ',CVOL);
242:C        5       IF ((STRLEN(SD)>0) OR (STRLEN(CVOL)>0)) AND USEUNIT THEN WRITE(' in');
243:C        5       IF USEUNIT THEN WRITE(' unit ',CPVOL);
244:C        5       WRITELN(CTEOL);
245:C        5       PROMPTREAD('''C'' continues, <sh_exc> aborts ',ANSWER,'C','C');
246:S
247:C        5       IF USEUNIT THEN TEMPNAME := CPVOL ELSE TEMPNAME := CVOL;
248:C        5       CFIB.FUNIT := FINDVOLUME(TEMPNAME,TRUE);
249:S
250:C        5       IF CFIB.FUNIT>0 THEN
251:C        6       BEGIN
252:C        6         IF IORESULT=ORD(INODIRECTORY) THEN
253:C        7         BEGIN
254:C        7           IF DSTATUS<>DONTCARE THEN WRITELN('NO DIRECTORY ON ',CPVOL);
255:C        7           SETSTRLEN(TEMPNAME,0);
256:C        7           CASE DSTATUS OF
257:C        8             DNEEDED: CFIB.FUNIT := 0;
258:C        8             DWANTED: BEGIN
259:C        8                        PROMPTYORN('Use current media',ANSWER);
260:C        8                        IF ANSWER='N' THEN CFIB.FUNIT := 0
261:C        9                                      ELSE DSTATUS    := DONTCARE;
262:C        8                      END;
263:C        8             OTHERWISE
264:C        8           END;   { CASE DSTATUS }
265:C        7         END
266:C        7         ELSE
267:C        7         BEGIN
268:C        7           IF IORESULT<>ORD(INOERROR) THEN
269:C        8           BEGIN
270:C        8             PRINTIOERRMSG; CFIB.FUNIT := 0;
271:C        8           END
272:C        8           ELSE
273:C        8           BEGIN { FOUND A DIRECTORY }
274:C        8             IF CVOL='' THEN CVOL := TEMPNAME
275:C        9             ELSE
276:C        9             IF CVOL<>TEMPNAME THEN CFIB.FUNIT := 0;
277:C        8           END;
278:C        7         END;
279:C        6       END;
280:C        5     UNTIL CFIB.FUNIT>0;
281:C        4     CFIB.FVID := CVOL;
282:C        4     MOUNTED   := TRUE;
283:C        4   END;
284:C        3 END;    { MOUNT VOLUME }
285:S
286:D  -890   2 (*****************************************************************************)
287:D        2 PROCEDURE SPACEWAIT;
288:D        3 VAR
289:D    -1  3   ANSWER        : CHAR;
290:C        3 BEGIN
291:C        3   PROMPTREAD('<space> continues, <sh_exc> aborts ',ANSWER,' ',' ');
292:C        3 END;    { SPACEWAIT }
293:S
294:D  -890   2 (*****************************************************************************)
295:D   -82   2   PROCEDURE PROMPTFORCHAR(PL : PROMPTTYPE; VAR CH : CHAR);
296:C        3   BEGIN
297:C        3     FGOTOXY(OUTPUT,0,0); WRITE(PL,' ? ',CTEOL);
298:C        3     READ(KEYBOARD,CH);   READCHECK;
299:C        3     UPCCHAR(CH); WRITELN(CH);
300:C        3     FGOTOXY(OUTPUT,0,1); WRITELN(CTEOL);
```



```
301:C        3   END;    { PROMPTFORCHAR }
302:S
303:D  -890   2 $IOCHECK OFF$
304:D  -890   2 (*****************************************************************************)
305:D        2 PROCEDURE SETUPFIBFORFILE(FILENAME      : FID;
306:D        3                           VAR LFIB      : FIB;
307:D  -122  3                           VAR VNAME     : VID);
308:D  -122  3 VAR
309:D  -124  3   LKIND : FILEKIND;
310:D  -128  3   SEGS  : INTEGER;
311:S
312:C        3 BEGIN
313:C        3   SEGS     := 0;
314:C        3   IORESULT := ORD(INOERROR);
315:C        3   WITH LFIB DO
316:C        4     IF SCANTITLE(FILENAME,FVID,FTITLE,SEGS,LKIND) THEN
317:C        5       BEGIN
318:C        5         VNAME      := FVID;
319:C        5         FUNIT      := FINDVOLUME(FVID,TRUE);
320:C        5         FKIND      := LKIND;      FEFT := EFTTABLE^[LKIND];
321:C        5         FOPTSTRING := NIL;        FBUFFERED  := TRUE;
322:C        5         FPOS       := SEGS * 512; FREPTCNT   := 0;
323:C        5         FANONYMOUS := FALSE;      FMODIFIED  := FALSE;
324:C        5         FBUFCHANGED:= FALSE;      FSTARTADDRESS := 0;
325:C        5         FLASTPOS   := -1;         PATHID     := -1;
326:C        5         FNOSRMTEMP := TRUE;       FLOCKED    := TRUE;
327:C        5         FEOF       := FALSE;      FEOLN      := FALSE;
328:C        5       END
329:C        5       ELSE BADIO(IBADTITLE);
330:C        3 END;    { SETUPFIBFORFILE }
331:S
332:D  -890   2 (*****************************************************************************)
333:D        2 PROCEDURE CLOSEDIR;
334:C        3 BEGIN
335:C        3   WITH ININFO, CFIB DO
336:C        4   BEGIN
337:C        4     IF DIROPEN THEN
338:C        5     BEGIN
339:C        5       LOCKUP;       { LOCK KEYBOARD FOR THIS OPERATION }
340:C        5       PATHID := PATH;   { RESTORE PATHID }
341:C        5       CALL(UNITABLE^[FUNIT].DAM,CFIB,FUNIT,CLOSEDIRECTORY);
342:C        5       DIROPEN := FALSE;
343:C        5       LOCKDOWN;
344:C        5     END;
345:C        4   END;
346:C        3 END;    { CLOSEDIR }
347:S
348:D  -890   2 (*****************************************************************************)
349:D        2 PROCEDURE OPENDIR(FILENAME      : FID;
350:D        3                   VAR SOURCEFILE    : FID;
351:D        3                       PROMPT        : PROMPTTYPE;
352:D        3                   VAR FINFO         : CONTROL;
353:D  -204  3                   VAR DIRCATENTRY   : CATENTRY);
354:D  -204  3 VAR
355:D  -208  3   UNIT     : INTEGER;
356:S
357:C        3 BEGIN   { OPENDIR }
358:C        3   IORESULT := ORD(INOERROR);
359:C        3   WITH FINFO, CFIB DO
360:C        4   TRY
```

```
   361:C        5     SETUPFIBFORFILE(FILENAME,CFIB,CPVOL);
   362:C        5     USEUNIT := UNITNUMBER(CPVOL);       DSTATUS := DNEEDED;
   363:C        5     IF USEUNIT THEN CVOL := '' ELSE CVOL := CPVOL;
   364:C        5     IF (FUNIT=0) OR UNITNUMBER(FVID) THEN MOUNTVOLUME(PROMPT,FINFO)
   365:C        6                                      ELSE MOUNTED := TRUE;
   366:C        5     WITH UNITABLE^[FUNIT] DO
   367:C        6     BEGIN
   368:C        6       LOCKUP;           { LOCK KEYBOARD }
   369:C        6       FWINDOW    := ADDR(DIRCATENTRY);
   370:C        6       CALL(DAM,CFIB,FUNIT,OPENDIRECTORY);
   371:C        6       DIROPEN    := (IORESULT=ORD(INOERROR));
   372:C        6       IF DIROPEN THEN
   373:C        7       BEGIN
   374:C        7         PATH       := PATHID;
   375:C        7         SOURCEFILE := FTITLE;
   376:C        7         CVOL       := DIRCATENTRY.CNAME;
   377:C        7       END;
   378:C        6       LOCKDOWN;         { UNLOCK KEYBOARD }
   379:C        6       IF NOT DIROPEN THEN ESCAPE(0);    { OPENDIRECTORY FAILED }
   380:C        6     END
   381:C        6   RECOVER
   382:C        5     IF ESCAPECODE<>0 THEN ESCAPE(ESCAPECODE);
   383:C        3 END;    { OPENDIR }
   384:S
   385:D  -890  2 (*****************************************************************************)
   386:D        2 PROCEDURE INMOUNT(SWAP : BOOLEAN);
   387:C        3 BEGIN
   388:C        3   IF NOT ININFO.MOUNTED THEN
   389:C        4   WITH ININFO, CFIB DO
   390:C        5   BEGIN
   391:C        5     MOUNTVOLUME(' SOURCE',ININFO);
   392:C        5     UNITABLE^[FUNIT].UMEDIAVALID := TRUE;
   393:C        5   END;
   394:C        3 END;    { INMOUNT }
   395:S
   396:D  -890  2 (*****************************************************************************)
   397:D        2 PROCEDURE CLOSEINFILE;
   398:C        3 BEGIN
   399:C        3   WITH ININFO ,CFIB DO
   400:C        4   BEGIN
   401:C        4     IF FILEOPEN THEN
   402:C        5     BEGIN
   403:C        5       LOCKUP;
   404:C        5       FMODIFIED := FALSE;
   405:C        5       CALL(UNITABLE^[FUNIT].DAM,CFIB,FUNIT,CLOSEFILE);
   406:C        5       FILEOPEN := FALSE;
   407:C        5       LOCKDOWN;
   408:C        5     END;
   409:C        4   END;
   410:C        3 END;    { CLOSEINFILE }
   411:S
   412:S
   413:D  -890  2 (*****************************************************************************)
   414:D        2 PROCEDURE CLOSEALL;
   415:C        3 BEGIN CLOSEINFILE; CLOSEDIR; END;
   416:S
   417:D  -890  2 (*****************************************************************************)
   418:D        2 PROCEDURE ANYTOMEM(       FFIB   : FIBP;
   419:D        3                     ANYVAR BUFFER : BIGPTR;
   420:D        3                            MAXBUF : INTEGER);
```



```
   421:D        3 VAR
   422:D    -4  3   BUFREC    :  ^STRING255;
   423:D    -8  3   BUFPTR    :  ^CHAR;
   424:D   -12  3   LEFTINBUF :  INTEGER;
   425:S
   426:C        3 BEGIN   { ANYTOMEM }
   427:C        3   BUFPTR    := ADDR(BUFFER^);
   428:C        3   BUFPTR^   := CHR(0);  { DATA COMMING }
   429:C        3   BUFREC    := ADDR(BUFPTR^,1);
   430:C        3   SETSTRLEN(BUFREC^,0); { ZERO LENGTH RECORD }
   431:C        3   BUFPTR    := ADDR(BUFREC^,1);
   432:C        3   LEFTINBUF := MAXBUF;
   433:S
   434:C        3   WITH FFIB^, UNITABLE^[FUNIT] DO
   435:C        4   BEGIN
   436:C        4     CALL(AM,FFIB,READTOEOL,BUFREC^,255,FPOS);
   437:C        4     GOODIO;
   438:C        4     REPEAT
   439:C        5       GOODIO; { CHECK IORESULT FROM LAST READTOEOL }
   440:C        5       LEFTTOXFER := LEFTTOXFER - STRLEN(BUFREC^);
   441:C        5       IF LEFTTOXFER<0 THEN
   442:C        6       BEGIN   { CLIP THIS LINE AND FAKE END OF FILE }
   443:C        6         SETSTRLEN(BUFREC^,STRLEN(BUFREC^)+LEFTTOXFER);
   444:C        6         LEFTTOXFER := 0;
   445:C        6       END;
   446:C        5       BUFPTR := ADDR(BUFPTR^,STRLEN(BUFREC^));
   447:S
   448:C        5       LEFTINBUF := LEFTINBUF - STRLEN(BUFREC^) - 2;
   449:C        5       IF STRLEN(BUFREC^) = 255 THEN BUFPTR := ADDR(BUFPTR^,-1)
   450:C        6       ELSE
   451:C        6       BEGIN
   452:C        6         IF STRLEN(BUFREC^)=0 THEN
   453:C        7         BEGIN { DISCARD THE LENGTH BYTE }
   454:C        7           BUFPTR := ADDR(BUFREC^,-1); LEFTINBUF := LEFTINBUF + 1;
   455:C        7         END;
   456:S
   457:C        6           { CHECK END OF LINE/FILE }
   458:C        6         CALL(AM,FFIB,READBYTES,BUFPTR^,1,FPOS);
   459:C        6         IF FEOLN THEN
   460:C        7         BEGIN  { END OF LINE }
   461:C        7           BUFPTR^ := CHR(1);  FEOLN := FALSE;
   462:C        7           IF IORESULT = ORD(IEOF) THEN BUFPTR := ADDR(BUFPTR^,1);
   463:C        7         END;
   464:C        6         IF (IORESULT=ORD(IEOF)) OR (LEFTTOXFER=0) THEN
   465:C        7         BEGIN  { END OF FILE }
   466:C        7           BUFPTR^  := CHR(2);
   467:C        7           IORESULT := ORD(INOERROR);
   468:C        7           FEOF     := TRUE;
   469:C        7         END;
   470:C        6         GOODIO;       { CHECK IORESULT FROM READBYTES }
   471:C        6       END;
   472:C        5       IF NOT ((LEFTINBUF < 259) OR FEOF) THEN
   473:C        6       BEGIN { SETUP FOR TO READ THE NEXT LINE }
   474:C        6         BUFPTR    := ADDR(BUFPTR^,1);
   475:C        6         BUFPTR^   := CHR(0);  { DATA RECORD }
   476:C        6         BUFREC    := ADDR(BUFPTR^,1);
   477:C        6         SETSTRLEN(BUFREC^,0); { ZERO LENGTH RECORD }
   478:C        6         BUFPTR    := ADDR(BUFREC^,1);
   479:C        6         CALL(AM,FFIB,READTOEOL,BUFREC^,255,FPOS);
   480:C        6       END;
```

```
481:C        5     UNTIL (LEFTINBUF < 259) OR FEOF;
482:C        4     BUFPTR := ADDR(BUFPTR^,1);    BUFPTR^ := CHR(3); { END BUFFER }
483:C        4   END;
484:C        3 END;    { ANYTOMEM }
485:S
486:D   -890  2 (*****************************************************************************)
487:D        2 FUNCTION CHECKSPACE(START,SIZE:INTEGER; VAR OKSIZE:INTEGER):BOOLEAN;
488:S
489:D        3 VAR
490:D    -4  3   POINT : INTEGER;
491:D    -8  3   PART  : INTEGER;
492:S
493:C        3 BEGIN { CHECK SPACE }
494:C        3   OKSIZE := -1; CHECKSPACE := FALSE;
495:C        3   WITH EPROMDATA^ DO
496:C        4   BEGIN
497:C        4     { FIND THE PART CONTAINING THE START ADDRESS }
498:C        4     PART := 0;
499:C        4     WHILE PART<8 DO
500:C        5     BEGIN
501:C        5       IF (START>=ADDRESS[PART]) AND
502:C        6          (START<(ADDRESS[PART]+EPINC)) THEN
503:C        6       BEGIN
504:C        6         IF PRESENT[PART] THEN
505:C        7         BEGIN   { FOUND START NOW FIND END POINT }
506:C        7           POINT := START + SIZE - 1;    { END POINT }
507:C        7           WHILE PART<8 DO
508:C        8           BEGIN
509:C        8             IF PRESENT[PART] THEN
510:C        9             BEGIN
511:C        9               IF (POINT<(ADDRESS[PART]+EPINC)) THEN
512:C       10               BEGIN     { IT ALL FITS }
513:C       10                 OKSIZE := SIZE; CHECKSPACE:=TRUE; PART := 8; { FORCE EXIT }
514:C       10               END
515:C       10               ELSE     { FIGURE SPACE SO FAR }
516:C       10               BEGIN
517:C       10                 OKSIZE := (ADDRESS[PART]+EPINC)-START;
518:C       10                 PART := PART + 1;
519:C       10               END;
520:C        9             END { IF PRESENT }
521:C        9             ELSE PART := 8;     { FORCE EXIT }
522:C        8           END;  { WHILE }
523:C        7         END;    { IF PRESENT }
524:C        6         PART := 8;       { FORCE EXIT }
525:C        6       END       { IF ADDRESS IN RANGE }
526:C        6       ELSE PART := PART + 1;
527:C        5     END;        { WHILE }
528:C        4   END;  { WITH EPROMDATA }
529:C        3 END;    { CHECKSPACE }
530:S
531:D   -890  2 (*****************************************************************************)
532:D        2 PROCEDURE PASSFAILED(FAILCODE:PASSTYPE);
533:D        3 VAR
534:D    -4  3   PART : INTEGER;
535:D    -5  3   UL   : CHAR;
536:D   -10  3   I    : INTEGER;
537:S
538:C        3 BEGIN
539:C        3   WITH EPROMDATA^ DO
540:C        4   BEGIN
```



```
541:C        4     FOR I := 0 TO 7 DO
542:C        5       IF (OUTPOSITION>=ADDRESS[I]) AND
543:C        6          (OUTPOSITION<(ADDRESS[I]+EPINC)) THEN PART := I;
544:S
545:C        4     IF ODD(OUTPOSITION) THEN UL := 'L' ELSE UL := 'U';
546:C        4     WRITE(FAILCODE,' FAIL');
547:C        4     WRITELN(' AT ABSOLUTE ADDRESS ',OUTPOSITION:1);
548:C        4     WRITELN(' BYTE POSITION ',OUTPOSITION-OUTSTARTA:1,' FROM START LOCATION');
549:C        4     WRITELN(' EPROM SOCKET ',PART:1,UL,
550:C        4             ' BYTE ',(OUTPOSITION-ADDRESS[PART]) DIV 2:1);
551:C        4   END;
552:C        3   IF STREAMING THEN ESCAPE(-1) ELSE BADIO(INOERROR);
553:C        3 END;    { PASSFAILED }
554:S
555:D   -890  2 (*****************************************************************************)
556:D        2 PROCEDURE PRINTBR;
557:C        3 BEGIN
558:C        3   FGOTOXY(OUTPUT,0,4); WRITE('Burn rate   ');
559:C        3   IF FASTBURN THEN WRITE('FAST') ELSE WRITE('SLOW');
560:C        3   WRITELN(CTEOL);
561:C        3 END;
562:S
563:D   -890  2 (*****************************************************************************)
564:D        2 PROCEDURE BURNIT(ANYVAR BUFFER   : WINDOW;
565:D        3                  SIZE     : INTEGER);
566:D        3 VAR
567:D    -4  3   OLDPOSIT : INTEGER;
568:D   -12  3   OLDX,OLDY: INTEGER;
569:C        3 BEGIN
570:C        3   IF PASS=1 THEN
571:C        4   BEGIN       { CHECK IF CAN BURN }
572:C        4     ERROR:=EPROG(SCODE,ECHECK,BUFFER,SIZE,OUTPOSITION);
573:C        4     IF ERROR<>ENOERROR THEN PASSFAILED(CHECK);
574:C        4   END
575:C        4   ELSE
576:C        4   BEGIN       { TRY TO BURN IT }
577:C        4     OLDPOSIT := OUTPOSITION;    { SAVE POSITION FOR RETRY }
578:C        4     ERROR:=EPROG(SCODE,ECWRITE,BUFFER,SIZE,OUTPOSITION);
579:C        4     IF ERROR<>ENOERROR THEN
580:C        5       IF NOT FASTBURN THEN
581:C        6       BEGIN
582:C        6         IF ERROR=ECFAIL THEN PASSFAILED(VERIFY)
583:C        7                         ELSE PASSFAILED(BURN);
584:C        6       END
585:C        6       ELSE
586:C        6       BEGIN
587:C        6         FASTBURN := FALSE;      { SET BURN RATE TO SLOW }
588:C        6         FGETXY(OUTPUT,OLDX,OLDY);       { UPDATE THE DISPLAY }
589:C        6         PRINTBR;
590:C        6         FGOTOXY(OUTPUT,OLDX,OLDY);
591:C        6         ERROR := EBRATE(SCODE,FASTBURN);
592:C        6         OUTPOSITION := OLDPOSIT;{ RESET POSITION AND RETRY }
593:C        6         ERROR:=EPROG(SCODE,ECWRITE,BUFFER,SIZE,OUTPOSITION);
594:C        6         IF ERROR<>ENOERROR THEN
595:C        7           IF ERROR=ECFAIL THEN PASSFAILED(VERIFY)
596:C        8                           ELSE PASSFAILED(BURN);
597:C        6       END;
598:C        4   END;
599:C        3   IORESULT:=ORD(INOERROR);      { CLEAR I/O RESULT }
600:C        3 END;    { BURNIT }
```

```
   601:S
   602:D  -890  2 (*****************************************************************************)
   603:D        2 PROCEDURE MEMTOEPROM(ANYVAR BUFFER : BIGPTR);
   604:D        3 VAR
   605:D    -4  3   BYTES : INTEGER;
   606:D    -8  3   BUFPTR: ^CHAR;
   607:S
   608:C        3 BEGIN
   609:C        3   BUFPTR := ADDR(BUFFER^);
   610:C        3   BEGIN
   611:C        3     BYTES := 0;
   612:C        3     REPEAT
   613:C        4       BUFPTR := ADDR(BUFPTR^,BYTES);
   614:C        4       BYTES  := ORD(BUFPTR^);
   615:C        4       BUFPTR := ADDR(BUFPTR^,1);
   616:C        4       CASE BYTES OF
   617:C        5       0: BEGIN          { DATA BYTES }
   618:C        5            BYTES := ORD(BUFPTR^);       { RECORD LENGTH }
   619:C        5            BUFPTR:= ADDR(BUFPTR^,1);
   620:C        5            BURNIT(BUFPTR^,BYTES);
   621:C        5          END;
   622:C        5       1: BEGIN          { END RECORD }
   623:C        5            BYTES := 0;
   624:C        5          END;
   625:C        5       2: BEGIN          { END FILE }
   626:C        5            BYTES := -1;
   627:C        5          END;
   628:C        5       3: BYTES := -1;   { END BUFFER }
   629:C        5       OTHERWISE IORESULT := ORD(IBADREQUEST);
   630:C        5       END;
   631:C        4       GOODIO;
   632:C        4     UNTIL BYTES<0;
   633:C        3   END;
   634:C        3 END;    { MEMTOEPROM }
   635:S
   636:D  -890  2 (*****************************************************************************)
   637:D        2   FUNCTION CHECKCARD(SC:INTEGER):BOOLEAN;
   638:D        3   VAR
   639:D    -2  3     ECODE : EPERROR;
   640:C        3   BEGIN
   641:C        3     ECODE:=EINIT(SC);
   642:C        3     CHECKCARD:=(ECODE=ENOERROR) OR (ECODE=ENOEPROM);
   643:C        3   END;  { CHECKCARD }
   644:S
   645:D  -890  2 (*****************************************************************************)
   646:D        2   PROCEDURE CHECKSCODE;
   647:C        3   BEGIN
   648:C        3     IF NOT CHECKCARD(SCODE)
   649:C        4       THEN BADMESSAGE('*** NO PROGRAMMER CARD IN SYSTEM ***');
   650:C        3   END;
   651:S
   652:D  -890  2 (*****************************************************************************)
   653:D        2   PROCEDURE CHECKEPROM;
   654:D        3   VAR
   655:D    -4  3     I       : INTEGER;
   656:D    -5  3     DONE    : BOOLEAN;
   657:S
   658:C        3   BEGIN
   659:C        3     CHECKSCODE;
   660:C        3     ERROR:=EGETINFO(SCODE,EPINFO);
```



```
   661:C        3     IF EPINFO.EPSTART=0 THEN
   662:C        4       BADMESSAGE('NO EPROM CARD ATTACHED TO PROGRAMMER CARD');
   663:S
   664:C        3     EPROMDATA := EPROMLIST;
   665:C        3     DONE := FALSE;
   666:C        3     REPEAT
   667:C        4       IF EPROMDATA=NIL THEN
   668:C        5       BEGIN
   669:C        5         NEW(EPROMDATA);
   670:C        5         WITH EPROMDATA^ , EPINFO DO
   671:C        6         BEGIN
   672:C        6           BASEADDR  := EPSTART;
   673:C        6           EPSIZE    := EPEND-EPSTART;
   674:C        6           EPINC     := EPSIZE DIV 8;
   675:C        6           ADDRESS[0]:= BASEADDR;
   676:C        6           PRESENT[0]:= TRUE;
   677:C        6           FOR I:=1 TO 7 DO
   678:C        7           BEGIN
   679:C        7             ADDRESS[I]:= ADDRESS[I-1]+EPINC;
   680:C        7             PRESENT[I]:= TRUE;
   681:C        7           END;
   682:C        6           NEXT := EPROMLIST;
   683:C        6           EPROMLIST := EPROMDATA;
   684:C        6           DONE := TRUE;
   685:C        6         END;
   686:C        5       END
   687:C        5       ELSE
   688:C        5       BEGIN
   689:C        5         IF EPROMDATA^.BASEADDR=EPINFO.EPSTART THEN DONE := TRUE
   690:C        6         ELSE EPROMDATA := EPROMDATA^.NEXT;
   691:C        5       END;
   692:C        4     UNTIL DONE;
   693:C        3   END; { CHECKEPROM }
   694:S
   695:D  -890  2 (*****************************************************************************)
   696:D        2   PROCEDURE FINDCARD(SCODES:BOOLEAN);
   697:D        3   VAR
   698:D    -4  3     SC : INTEGER;
   699:C        3   BEGIN
   700:C        3     FOR SC:=MINSC TO MAXSC DO
   701:C        4     IF CHECKCARD(SC) THEN
   702:C        5     BEGIN
   703:C        5       IF SCODES THEN WRITE(SC:3);
   704:C        5       SCODE := SC;
   705:C        5     END;
   706:C        3   END; { FINDCARD }
   707:S
   708:D  -890  2 (*****************************************************************************)
   709:D        2   PROCEDURE DOCONFIGURE(DOPROMPT:BOOLEAN);
   710:D        3   VAR
   711:D    -1  3     DONE : BOOLEAN;
   712:D    -6  3     OLDSCODE : INTEGER;
   713:D    -7  3     OP       : CHAR;
   714:D   -12  3     I        : INTEGER;
   715:S
   716:D   -12  3     {-------------------------------------------------------------------------}
   717:D        3     PROCEDURE PRINTSC;
   718:C        4     BEGIN
   719:C        4       FGOTOXY(OUTPUT,0,3);
   720:C        4       WRITELN('Active programmer card at select code ',SCODE:1,CTEOL);
```

```
721:C         4     END;
722:S
723:S
724:D   -12   3     {-------------------------------------------------------------------------}
725:D         3     PROCEDURE PRINTEPINFO;
726:C         4     BEGIN
727:C         4       FGOTOXY(OUTPUT,0,5);
728:C         4       TRY
729:C         5         CHECKEPROM;
730:C         5         TEMP := EPINFO.EPEND-EPINFO.EPSTART;
731:C         5         WRITELN('EPROM at address ',EPINFO.EPSTART:1,' for ',
732:C         5                 TEMP:1,' bytes',CTEOL);
733:C         5         WRITELN('EPROM type XX',TEMP DIV 2048:1,CTEOL);
734:C         5       RECOVER
735:C         5         IF ESCAPECODE<>0 THEN ESCAPE(ESCAPECODE);
736:C         4     END;
737:S
738:D   -12   3     {-------------------------------------------------------------------------}
739:D         3     PROCEDURE PRINTSOCKETS;
740:D         4     VAR
741:D    -4   4       I : INTEGER;
742:C         4     BEGIN
743:C         4       FGOTOXY(OUTPUT,0,7);
744:C         4       WRITELN('Socket status (UL means EPROM pair present)');
745:C         4       WITH EPROMDATA^ DO
746:C         5       FOR I:= 0 TO 3 DO
747:C         6       BEGIN
748:C         6         WRITE(I:1);
749:C         6         IF PRESENT[I] THEN WRITE('UL    ') ELSE WRITE(' empty');
750:C         6         WRITE(I+4:4);
751:C         6         IF PRESENT[I+4] THEN WRITE('UL    ') ELSE WRITE(' empty');
752:C         6         WRITELN;
753:C         6       END;
754:C         4     END;
755:S
756:D   -12   3     {-------------------------------------------------------------------------}
757:C         3   BEGIN { DOCONFIGURE }
758:C         3     FGOTOXY(OUTPUT,0,2); WRITE(CTEOS);
759:C         3     CHECKSCODE;
760:C         3     OLDSCODE:=SCODE;
761:C         3     WRITE('Programmer card(s) at ');
762:C         3     FINDCARD(TRUE);
763:C         3     SCODE := OLDSCODE;
764:C         3     PRINTSC;
765:C         3     PRINTBR;
766:C         3     PRINTEPINFO;
767:C         3     IF EPINFO.EPSTART>0 THEN PRINTSOCKETS;
768:C         3     IF DOPROMPT THEN
769:C         4     REPEAT
770:C         5     TRY
771:C         6       PROMPTFORCHAR('CONFIGURE: Selectcode Burnrate Emptysockets Qt',OP);
772:C         6       FGOTOXY(OUTPUT,0,13); WRITE(CTEOS);
773:C         6       IF OP=SH_EXC THEN OP:='Q';
774:S
775:C         6       IF OP='S' THEN
776:C         7       BEGIN    { NEW SELECT CODE }
777:C         7         OLDSCODE := SCODE;
778:C         7         WRITE('New select code (',SCODE:1,') ? '); READNUMBER(SCODE);
779:C         7         IF (SCODE<MINSC) OR (SCODE>MAXSC) THEN
780:C         8         BEGIN SCODE := OLDSCODE; BADMESSAGE('SELECT CODE OUT OF RANGE'); END;
```



```
781:C         7         IF NOT CHECKCARD(SCODE) THEN
782:C         8         BEGIN
783:C         8           SCODE := OLDSCODE; BADMESSAGE('SELECT CODE NOT A PROGRAMMER CARD');
784:C         8         END;
785:C         7         PRINTSC;
786:C         7         PRINTBR;
787:C         7         PRINTEPINFO;
788:C         7         IF EPINFO.EPSTART>0 THEN PRINTSOCKETS ELSE WRITE(CTEOS);
789:C         7       END
790:C         7       ELSE
791:C         7       IF OP='B' THEN
792:C         8       BEGIN
793:C         8         FASTBURN := NOT FASTBURN; PRINTBR;
794:C         8       END
795:C         8       ELSE
796:C         8       IF OP='E' THEN
797:C         9       BEGIN
798:C         9         IF EPINFO.EPSTART>0 THEN
799:C        10         BEGIN
800:C        10           I := -1;
801:C        10           WRITE('SOCKET (PAIR) NUMBER ? ',CTEOL); READNUMBER(I);
802:C        10           IF I>=0 THEN
803:C        11             IF (I<0) OR (I>7) THEN BADMESSAGE('SOCKET NUMBER OUT OF RANGE')
804:C        12             ELSE
805:C        12             BEGIN
806:C        12               EPROMDATA^.PRESENT[I] := NOT EPROMDATA^.PRESENT[I];
807:C        12               PRINTSOCKETS;
808:C        12             END;
809:C        10         END
810:C        10         ELSE BEEP;
811:C         9       END
812:C         9       ELSE
813:C         9       IF (OP<>'Q') THEN
814:C        10         IF STREAMING THEN BADCOMMAND(OP)
815:C        11                      ELSE BEEP;
816:S
817:C         6     RECOVER
818:C         6     BEGIN
819:C         6       LOCKUP;
820:C         6       SAVEIO       := IORESULT;
821:C         6       SAVEESC      := ESCAPECODE;
822:C         6       IORESULT     := SAVEIO;
823:C         6       IF (SAVEESC<>0) AND (SAVEESC<>-10) THEN IORESULT := ORD(INOERROR);
824:C         6       LOCKDOWN;
825:C         6       PRINTIOERRMSG;
826:C         6       FIXLOCK;
827:C         6       IF SAVEESC<>0 THEN ESCAPE(SAVEESC) ELSE OP := ' ';
828:C         6     END;
829:S
830:C         5     UNTIL OP='Q';
831:C         3   END; { DOCONFIGURE }
832:S
833:D  -890   2 (*************************************************************************)
834:D         2   PROCEDURE DOTRANSFER;
835:D         3   TYPE
836:D         3     HEADREC = PACKED ARRAY[0..17] OF BYTE;
837:S
838:D         3     HEADP = RECORD
839:D         3               CASE BOOLEAN OF
840:D         3               TRUE : (BINT:INTEGER);
```

```
841:D        3              FALSE: (BPTR:^HEADREC);
842:D        3            END;
843:D        3  VAR
844:D  -122  3    FILENAME1     : FID;
845:D  -244  3    SOURCEFILE    : FID;
846:S
847:D  -245  3    FILEMOVED     : BOOLEAN;
848:D  -246  3    DONE          : BOOLEAN;
849:D  -247  3    FORMAT        : BOOLEAN;
850:S
851:D  -252  3    I             : INTEGER;
852:D  -256  3    J             : INTEGER;
853:D  -260  3    EPART         : INTEGER;
854:D  -264  3    INSTATE       : INTEGER;
855:D  -268  3    OUTSTATE      : INTEGER;
856:D  -272  3    BUF           : BIGPTR;
857:D  -276  3    POSITION      : INTEGER;
858:D  -280  3    MOVESIZE      : INTEGER;
859:D  -284  3    MSIZE         : INTEGER;
860:D  -288  3    BUFSIZE       : INTEGER;
861:D  -292  3    OUTSIZE       : INTEGER;
862:D  -296  3    SAVEIO        : INTEGER;
863:D  -300  3    SAVEESC       : INTEGER;
864:D  -304  3    DUMWINDOW     : WINDOWP;
865:D  -322  3    EDHEADER      : HEADREC;
866:D  -578  3    MSGLINE       : STRING[255];
867:D  -658  3    DIRCATENTRY   : CATENTRY;
868:D  -662  3    BLANKCHK      : HEADP;
869:D  -663  3    ANSWER        : CHAR;
870:S
871:C        3  BEGIN   { DOTRANSFER }
872:C        3    DOCONFIGURE(FALSE);
873:C        3    FGOTOXY(OUTPUT,0,13);
874:C        3    WRITELN('TRANSFER OPERATION',CTEOS);
875:C        3    CHECKEPROM;
876:S
877:C        3    WRITE('Source (',DKVID,':) ? ');
878:C        3    READLN(FILENAME1);     GOODIO;
879:C        3    FILENAME1 := STRLTRIM(STRRTRIM(FILENAME1));
880:C        3    IF STRLEN(FILENAME1)=0 THEN FILENAME1:=DKVID+':';
881:C        3    ZAPSPACES(FILENAME1);
882:C        3    IF STRLEN(FILENAME1)>0 THEN
883:C        4    WITH EPROMDATA^ DO
884:C        5    BEGIN { HAVE A SOURCE NAME }
885:C        5      WITH ININFO DO
886:C        6        BEGIN DIROPEN := FALSE; FILEOPEN := FALSE; MOUNTED := FALSE; END;
887:C        5      MARK(LHEAP);  HEAPINUSE := TRUE;
888:C        5      NEWWORDS(DUMWINDOW,1);  { DUMMY WINDOW FOR FILE TRANSFER }
889:C        5      TRY
890:C        6        WITH ININFO, CFIB DO
891:C        7        BEGIN
892:C        7          { OPEN THE SOURCE }
893:C        7          SETUPFIBFORFILE(FILENAME1,CFIB,CPVOL);
894:C        7          IF STRLEN(FTITLE)=0 THEN
895:C        8          BEGIN { VOLUME -> EPROM }
896:C        8            USEUNIT := UNITNUMBER(CPVOL);     DSTATUS := DWANTED;
897:C        8            IF USEUNIT THEN CVOL := '' ELSE CVOL := CPVOL;
898:C        8            MOUNTED := (FUNIT>0) AND NOT(UNITNUMBER(FVID));
899:C        8            IF MOUNTED THEN CVOL := FVID ELSE INMOUNT(TRUE);
900:S
```



```
901:C        8            LOCKUP;   { LOCK THE KEYBOARD THEN OPEN THE VOLUME }
902:C        8            FBUFFERED := FALSE;
903:C        8            FKIND     := UNTYPEDFILE;     FEFT := EFTTABLE^[FKIND];
904:C        8            CALL(UNITABLE^[FUNIT].DAM,CFIB,FUNIT,OPENVOLUME);
905:C        8            FILEOPEN  := (IORESULT=ORD(INOERROR));
906:C        8            LOCKDOWN; { UNLOCK THE KEYBOARD }
907:C        8            GOODIO;
908:C        8                      { FINISH THE SETUPT }
909:C        8            OUTSIZE   := FPEOF;
910:C        8            SOURCEFILE := '';
911:C        8            FTID      := '';
912:C        8            FORMAT    := FALSE;
913:C        8            FGOTOXY(OUTPUT,0,14);
914:C        8            WRITELN('TRANSFERRING VOLUME ',FVID,':');
915:C        8          END { VOLUME -> EPROM }
916:C        8          ELSE
917:C        8          BEGIN { FILE -> EPROM }
918:C        8            OPENDIR(FILENAME1,SOURCEFILE,' SOURCE',ININFO,DIRCATENTRY);
919:C        8            IF NOT DIROPEN THEN ESCAPE(0);
920:C        8            IF STRLEN(SOURCEFILE)=0 THEN
921:C        9              BADMESSAGE('CAN''T TRANSFER A DIRECTORY');
922:C        8            FTITLE := SOURCEFILE;
923:C        8            FINITB(CFIB,DUMWINDOW,-3);
924:C        8            PATHID := PATH;
925:C        8            LOCKUP;
926:C        8            CALL(UNITABLE^[FUNIT].DAM,CFIB,FUNIT,OPENFILE);
927:C        8            FILEOPEN := IORESULT=ORD(INOERROR);
928:C        8            LOCKDOWN;
929:C        8            GOODIO;
930:C        8            FORMAT := (FKIND=ASCIIFILE) OR (FKIND=TEXTFILE);
931:C        8            OUTSIZE := FLEOF;
932:C        8            FGOTOXY(OUTPUT,0,14);
933:C        8            WRITELN('TRANSFERING FILE ',CVOL,':',SOURCEFILE,CTEOL);
934:C        8          END; { FILE -> EPROM }
935:C        7        END;   { WITH ININFO, CFIB -- OPEN THE SOURCE }
936:S
937:C        6        { ALLOCATE BUFFER SPACE }
938:C        6        BUFSIZE := (MEMAVAIL DIV 256) * 256 - 30 * 512; {SAME SOME FOR SLOP}
939:C        6        IF BUFSIZE<512 THEN ESCAPE(-2);          { NOT ENOUGH ROOM }
940:C        6        NEWWORDS(BUF,BUFSIZE DIV 2);             { ALLOCATE BUFFER SPACE }
941:S
942:C        6        { GET START ADDRESS ON EPROM }
943:C        6        MSGLINE := '';
944:C        6        WITH EPINFO DO
945:C        7        IF SOURCEFILE='' THEN
946:C        8        BEGIN     { VOLUME TRANSFER }
947:C        8          { SET DEFAULT START BLOCK }
948:C        8          TEMP := 0;
949:C        8          OUTSTARTA := EPSTART ;
950:C        8          DONE := FALSE;
951:C        8          REPEAT
952:C        9            IF PRESENT[TEMP] THEN
953:C       10            BEGIN       { SOCKET PRESENT, CHECK CONTENTS }
954:C       10              BLANKCHK.BINT := OUTSTARTA;
955:C       10              IF (BLANKCHK.BPTR^[0]=255) AND
956:C       11                 (BLANKCHK.BPTR^[1]=255) THEN DONE := TRUE
957:C       11            ELSE
958:C       11            BEGIN     { INCREMENT TO NEXT BLOCK }
959:C       11              OUTSTARTA := OUTSTARTA+K16;
960:C       11              IF TEMP<7 THEN
```

```
961:C  11                  IF OUTSTARTA>=ADDRESS[TEMP+1] THEN TEMP := TEMP+1;
962:C  11                END;
963:C  10              END
964:C  10              ELSE
965:C  10              BEGIN       { SKIP EMPTY SOCKET PAIR }
966:C  10                IF TEMP<7 THEN TEMP := TEMP+1;
967:C  10                OUTSTARTA := OUTSTARTA+EPINC;
968:C  10              END;
969:C   9            UNTIL DONE OR (OUTSTARTA>EPEND);
970:S
971:C   8            IF OUTSTARTA>EPEND THEN
972:C   9            BEGIN
973:C   9              WRITELN('*** NO BLANK BLOCK ON THIS EPROM CARD ***');
974:C   9              OUTSTARTA:=EPSTART;
975:C   9            END;
976:C   8            OUTSTARTA := (OUTSTARTA-EPSTART) DIV K16;
977:S
978:C   8            WRITE('Start at EPROM block offset (',OUTSTARTA:1,') ? ');
979:C   8            READNUMBER(OUTSTARTA); OUTSTARTA:=OUTSTARTA*K16;
980:C   8            STRWRITE(MSGLINE,1,I,'BLOCK OFFSET NOT IN RANGE 0..',
981:C   8                                 (EPSIZE DIV K16)-1:1);
982:C   8          END
983:C   8          ELSE
984:C   8          BEGIN     { FILE TRANSFER }
985:C   8            OUTSTARTA := 0;          { DEFAULT VALUE }
986:C   8            WRITE('Start at EPROM byte offset (',OUTSTARTA:1,') ? ');
987:C   8            READNUMBER(OUTSTARTA);
988:C   8            STRWRITE(MSGLINE,1,I,'BYTE OFFSET NOT IN RANGE 0..',EPSIZE-1:1);
989:C   8          END;
990:S
991:C   6          IF OUTSTARTA>(EPSIZE-1) THEN BADMESSAGE(MSGLINE);
992:C   6          OUTSTARTA := OUTSTARTA + EPINFO.EPSTART;
993:S
994:C   6          { CHECK TO SEE IF DATA WILL FIT IN AVAILABLE EPROM SPACE }
995:C   6          J := OUTSIZE;    { VOL / FILE SIZE }
996:C   6          IF SOURCEFILE='' THEN
997:C   7          BEGIN
998:C   7            J := J + 18; { ADD HEADER }
999:C   7            J := J + (J DIV K16)*2; { ADD 16K HEADER GAPS }
1000:C  7          END;
1001:C  6          IF NOT CHECKSPACE(OUTSTARTA,J,I) THEN
1002:C  7          BEGIN
1003:C  7            IF I<=0 THEN
1004:C  8            BEGIN WRITELN('NO EPROM AT START ADDRESS'); BADIO(INOERROR); END;
1005:S
1006:C  7            WRITELN('DATA EXCEEDS EPROM SPACE BY ',J-I:1,' BYTES',CTEOL);
1007:C  7            IF STREAMING THEN ESCAPE(-1);
1008:C  7            PROMPTREAD('Abort transfer or Truncate file (A/T) ? ',
1009:C  7                       ANSWER,'AT',SH_EXC);
1010:C  7            IF ANSWER='A' THEN ESCAPE(0);
1011:C  7            IF SOURCEFILE='' THEN
1012:C  8            BEGIN
1013:C  8              OUTSIZE := (I-18); OUTSIZE := OUTSIZE - (OUTSIZE DIV K16)*2;
1014:C  8            END
1015:C  8            ELSE OUTSIZE := I;
1016:C  7          END;
1017:S
1018:C  6          INSTATE  := 1;
1019:C  6          PASS     := 1;
1020:S
```



```
1021:C  6          BEGIN       { TRY THE TRANSFER }
1022:C  6            FILEMOVED := FALSE;
1023:C  6            WITH ININFO, CFIB, EPROMDATA^ DO
1024:C  7            REPEAT    { MOVE THE FILE }
1025:C  8              DONE := FALSE;
1026:C  8              REPEAT
1027:C  9                CASE INSTATE OF
1028:C 10                1: BEGIN      { INITIALIZE SOURCE PARAMETERS }
1029:C 10                     FGOTOXY(OUTPUT,0,16);
1030:C 10                     WRITELN('now on pass ',PASS:1,CTEOS);
1031:C 10                     OUTPOSITION := OUTSTARTA;
1032:C 10                     LEFTTOXFER  := OUTSIZE;
1033:C 10                     POSITION    := 0;
1034:C 10                     FEOF        := FALSE;   FEOLN    := FALSE;
1035:C 10                     FLASTPOS    := -1;      FPOS     := 0;
1036:C 10                     INSTATE     := 2;
1037:C 10                     OUTSTATE    := 1;
1038:C 10                   END;
1039:C 10                2: BEGIN      { READ THE FILE/VOLUME }
1040:C 10                     WRITE('reading ....',CTEOL,CHR(13));
1041:C 10                     IF FORMAT THEN
1042:C 11                     BEGIN    { FORMATED TRANSFER }
1043:C 11                       ANYTOMEM(ADDR(CFIB),BUF,BUFSIZE);
1044:C 11                       DONE := TRUE;
1045:C 11                       IF FEOF THEN LEFTTOXFER := 0;
1046:C 11                       GOODIO;
1047:C 11                     END
1048:C 11                     ELSE
1049:C 11                     BEGIN    { UNFORMATED TRANSFER }
1050:C 11                       IF BUFSIZE>LEFTTOXFER THEN MOVESIZE := LEFTTOXFER
1051:C 12                                             ELSE MOVESIZE := BUFSIZE;
1052:C 11                       CALL(UNITABLE^[FUNIT].TM,ADDR(CFIB),READBYTES,
1053:C 11                                                BUF^,MOVESIZE,POSITION);
1054:C 11                       GOODIO;
1055:C 11                       LEFTTOXFER := LEFTTOXFER - MOVESIZE;
1056:C 11                       DONE := TRUE;
1057:C 11                     END;
1058:C 10                  END;
1059:C 10              END;  { CASE INSTATE }
1060:C  9            UNTIL DONE;
1061:C  8            WRITE(CTEOL);
1062:S
1063:C  8            DONE := FALSE;
1064:C  8            IF NOT FILEMOVED THEN
1065:C  9            REPEAT
1066:C 10              IF PASS=1 THEN WRITE('checking ...',CTEOL,CHR(13))
1067:C 11                        ELSE WRITE('writing ...',CTEOL,CHR(13));
1068:C 10              CASE OUTSTATE OF
1069:C 11              1: BEGIN
1070:C 11                   { SET BURN RATE }
1071:C 11                   ERROR:=EBRATE(SCODE,FASTBURN);
1072:C 11                   IF SOURCEFILE='' THEN
1073:C 12                   BEGIN       { VOLUME TRANSFER }
1074:C 12                     { PUT EDISC VOLUME HEADER }
1075:C 12                     FOR I := 0 TO 17 DO EDHEADER[I]:=0;
1076:C 12                     EDHEADER[0]:=HEX('F0');
1077:C 12                     EDHEADER[1]:=HEX('FF');
1078:C 12                     EDHEADER[2]:=ORD(' ');
1079:C 12                     EDHEADER[3]:=HEX('18');
1080:C 12                     EDHEADER[12]:=HEX('01');
```

```
1081:C       12                      EDHEADER[13]:=HEX('02');
1082:C       12                      BURNIT(EDHEADER,18);
1083:C       12                    END;
1084:C       11                    OUTSTATE := 2;
1085:C       11                  END;
1086:S
1087:C       11                2: BEGIN  { WRITE DATA }
1088:C       11                     IF FORMAT THEN MEMTOEPROM(BUF)
1089:C       12                     ELSE
1090:C       12                     BEGIN
1091:C       12                       IF SOURCEFILE='' THEN
1092:C       13                       BEGIN      { TRANSFERING A VOLUME }
1093:C       13                         I := 0;
1094:C       13                         REPEAT
1095:C       14                           { WATCH FOR 16K BYTE BOUNDARIES }
1096:C       14                           MSIZE:=(((OUTPOSITION+K16) DIV K16)*K16)-OUTPOSITION;
1097:C       14                           IF MSIZE>MOVESIZE THEN MSIZE:=MOVESIZE;
1098:C       14                           BURNIT(BUF^[I],MSIZE);
1099:C       14                           I := I + MSIZE;
1100:C       14                           MOVESIZE := MOVESIZE - MSIZE;
1101:C       14                           IF ((OUTPOSITION MOD K16)=0) AND
1102:C       15                              ((MOVESIZE>0) OR (LEFTTOXFER>0)) THEN
1103:C       15                           BEGIN        { PUT ZEROES IN BOUNDARY BYTES }
1104:C       15                             J:=0; BURNIT(J,2);
1105:C       15                           END;
1106:C       14                         UNTIL MOVESIZE=0;
1107:C       13                       END
1108:C       13                       ELSE
1109:C       13                       BEGIN      { TRANSFERING A FILE }
1110:C       13                         BURNIT(BUF^,MOVESIZE);
1111:C       13                       END;
1112:C       12                     END;
1113:C       11                     DONE:=TRUE;
1114:C       11                     IF LEFTTOXFER=0 THEN
1115:C       12                     BEGIN
1116:C       12                       IF PASS=2 THEN FILEMOVED:=TRUE
1117:C       13                       ELSE
1118:C       13                       BEGIN  PASS:=2; INSTATE := 1; END;
1119:C       12                     END;
1120:C       11                   END;
1121:C       11                 END;      { CASE OUTSTATE }
1122:C       10               UNTIL DONE;
1123:C        8               WRITE(CTEOL);
1124:C        8             UNTIL FILEMOVED;
1125:C        6             WRITELN('TRANSFER COMPLETED');
1126:C        6             IF FORMAT THEN I := OUTPOSITION-OUTSTARTA
1127:C        7                        ELSE I := OUTSIZE;
1128:C        6             WRITELN(I:1,' data bytes programmed and verified');
1129:C        6           END;
1130:C        6         RELEASE(LHEAP);       HEAPINUSE := FALSE;
1131:C        6         CLOSEALL;
1132:C        6       RECOVER
1133:C        6       BEGIN
1134:C        6         LOCKUP;
1135:C        6         RELEASE(LHEAP);       HEAPINUSE := FALSE;
1136:C        6         SAVEIO    := IORESULT;
1137:C        6         SAVEESC   := ESCAPECODE;
1138:C        6         CLOSEALL;
1139:C        6         IORESULT  := SAVEIO;
1140:C        6         LOCKDOWN;
```



```
1141:C        6         PRINTIOERRMSG;
1142:C        6         IF SAVEESC<>0 THEN ESCAPE(SAVEESC);
1143:C        6       END;
1144:C        5     END;  { HAVE SOURCE NAME }
1145:C        3   END;    { DOTRANSFER }
1146:D  -890  2
1147:D  -890  2 (*****************************************************************************)
1148:D   -82  2   PROCEDURE PUTMENU(MSTRING:STRING80);
1149:C        3   BEGIN
1150:C        3     FGOTOXY(OUTPUT,0,2);
1151:C        3     WRITE(MSTRING,' ? ',CTEOL);
1152:C        3   END;
1153:S
1154:D  -890  2 (*****************************************************************************)
1155:D        2   PROCEDURE DOBLANKCHECK;
1156:D        3   TYPE
1157:D        3     TWOBYTES = PACKED ARRAY[0..1] OF BYTE;
1158:D        3   VAR
1159:D    -4  3     OLDSTART    : INTEGER;
1160:D    -8  3     START       : INTEGER;
1161:D   -12  3     ENDSCAN     : INTEGER;
1162:D   -16  3     NBYTES      : INTEGER;
1163:D   -17  3     BLANKS      : BOOLEAN;
1164:D   -22  3     LINES       : INTEGER;
1165:D   -28  3     I           : INTEGER;
1166:D   -34  3     X,Y         : INTEGER;
1167:D   -34  3     BREC        : RECORD
1168:D   -34  3                     CASE BOOLEAN OF
1169:D   -34  3                     TRUE:(BPTR : ^TWOBYTES);
1170:D   -34  3                     FALSE:(BINT : INTEGER);
1171:D   -38  3                   END;
1172:C        3   BEGIN
1173:C        3     DOCONFIGURE(FALSE);
1174:C        3     FGOTOXY(OUTPUT,0,13);
1175:C        3     WRITELN('BLANK CHECK',CTEOS);
1176:C        3     CHECKEPROM;
1177:C        3     OLDSTART := 0;
1178:C        3     BLANKS := FALSE;
1179:C        3     FGETXY(OUTPUT,X,Y); LINES := 0;
1180:C        3     WITH EPROMDATA^ DO
1181:C        4     FOR I:=0 TO 7 DO    { DO ONE SOCKET PAIR AT A TIME }
1182:C        5     IF NOT PRESENT[I] THEN
1183:C        6     BEGIN      { CLOSE OFF REPORT OF PREVIOUS PAIR }
1184:C        6       IF BLANKS THEN
1185:C        7         WRITELN(BREC.BINT-1-BASEADDR:1,' (',BREC.BINT-OLDSTART:1,')');
1186:C        6       BLANKS:=FALSE
1187:C        6     END
1188:C        6     ELSE
1189:C        6     BEGIN       { SOCKET PAIR PRESENT SO CHECK IT OUT }
1190:C        6       BREC.BINT := ADDRESS[I] ;
1191:C        6       ENDSCAN := BREC.BINT + EPINC;
1192:C        6       REPEAT
1193:C        7         IF BLANKS THEN
1194:C        8         BEGIN
1195:C        8           IF BREC.BPTR^[0]<>255 THEN
1196:C        9           BEGIN
1197:C        9             WRITELN(BREC.BINT-1-BASEADDR:1,' (',BREC.BINT-OLDSTART:1,')');
1198:C        9             BLANKS:=FALSE;
1199:C        9           END;
1200:C        8         END
```

```
1201:C         8          ELSE
1202:C         8          BEGIN
1203:C         8            IF BREC.BPTR^[0]=255 THEN
1204:C         9            BEGIN
1205:C         9              IF LINES>5 THEN
1206:C        10              BEGIN
1207:C        10                LINES := 0; SPACEWAIT;
1208:C        10                FGOTOXY(OUTPUT,X,Y); WRITE(CTEOS);
1209:C        10              END;
1210:C         9              LINES := LINES + 1;
1211:C         9              OLDSTART := BREC.BINT; WRITE(OLDSTART-BASEADDR,' - ');
1212:C         9              BLANKS:=TRUE;
1213:C         9            END;
1214:C         8          END;
1215:C         7          BREC.BINT := BREC.BINT + 1;
1216:C         7        UNTIL BREC.BINT=ENDSCAN;
1217:S
1218:C         6        IF BLANKS AND (I=7) THEN
1219:C         7          WRITELN(BREC.BINT-1-BASEADDR:1,' (',BREC.BINT-OLDSTART:1,')');
1220:S
1221:C         6      END; { FOR I := ... }
1222:C         3      IF OLDSTART=0 THEN WRITELN('NO BLANK SPACE FOUND');
1223:C         3    END;  { DOBLANKCHECK }
1224:S
1225:D  -890   2  (*********************************************************************)
1226:C         2  BEGIN  { COMMANDLEVEL }
1227:C         2    FIXLOCK;
1228:C         2    WITH ININFO DO BEGIN DIROPEN:=FALSE; FILEOPEN:=FALSE; END;
1229:C         2    HEAPINUSE := FALSE;
1230:C         2    IORESULT  := ORD(INOERROR);
1231:C         2    SCODE := 0;      TEMP := 0;
1232:C         2    FASTBURN := FALSE;     { DEFAULT BURN RATE }
1233:C         2    EPROMLIST := NIL;      { NO EPROM CARD INFO YET }
1234:C         2    FINDCARD(FALSE);       { FIND A PROGRAMMER CARD }
1235:C         2    TRY
1236:C         3      DOCONFIGURE(FALSE); { DISPLAY DEFAULT CONFIGURATION }
1237:C         3    RECOVER
1238:C         3      IF ESCAPECODE<>0 THEN ESCAPE(ESCAPECODE);
1239:S
1240:C         2    FGOTOXY(OUTPUT,0,13);
1241:C         2    WRITELN('Copyright 1983 Hewlett-Packard Company.');
1242:C         2    WRITELN('       All rights are reserved.');
1243:S
1244:C         2    REPEAT
1245:C         3      TRY
1246:C         4        PROMPTFORCHAR('ETU: Transfer Configure Blankcheck Quit',OP);
1247:C         4        IF OP=SH_EXC THEN OP:='Q';
1248:S
1249:C         4        IF OP='T' THEN DOTRANSFER
1250:C         5        ELSE
1251:C         5        IF OP='C' THEN DOCONFIGURE(TRUE)
1252:C         6        ELSE
1253:C         6        IF OP='B' THEN DOBLANKCHECK
1254:C         7        ELSE
1255:C         7        IF OP='Q' THEN BEGIN END
1256:C         8        ELSE
1257:C         8        IF STREAMING THEN BADCOMMAND(OP)
1258:C         9                     ELSE BEEP;
1259:C         4      RECOVER
1260:C         4      BEGIN
```



```
1261:C         4        LOCKUP;
1262:C         4        IF HEAPINUSE THEN RELEASE(LHEAP);
1263:C         4        HEAPINUSE    := FALSE;
1264:C         4        SAVEIO       := IORESULT;
1265:C         4        SAVEESC      := ESCAPECODE;
1266:C         4        CLOSEALL;
1267:C         4        IORESULT     := SAVEIO;
1268:C         4        IF (SAVEESC<>0) AND (SAVEESC<>-10) THEN IORESULT := ORD(INOERROR);
1269:C         4        LOCKDOWN;
1270:C         4        PRINTIOERRMSG;
1271:C         4        FIXLOCK;
1272:C         4        IF SAVEESC<>0 THEN ESCAPE(SAVEESC) ELSE OP := ' ';
1273:C         4      END;
1274:C         3    UNTIL OP='Q';
1275:C         2  END {COMMANDLEVEL} ;
1276:S
1277:D         1  (*************************************************************************)
1278:C         1  BEGIN
1279:C         1    WRITELN(CLEARSCR);
1280:C         1    FGOTOXY(OUTPUT,0,1);
1281:C         1    WRITELN('EPROM TRANSFER UTILITY (4-Jun-84)');
1282:C         1    COMMANDLEVEL;
1283:C         1  END.
1284:S
```

No errors. No warnings.
***** Nonstandard language features enabled *****

F9885

Description

F9885 is the transfer method (TM) for the 9885 flexible disc drives.

Requirements

IODECLARATIONS, GPIODVR.

Notes

Although the 9885 drive uses the 98622 GPIO interface and the 98620 DMA card, it does not require the IOLIBRARY GPIO and DMA modules.

```
    1:S                                                 (*
    2:S
    3:S           (c) Copyright Hewlett-Packard Company, 1983.
    4:S           All rights are reserved.  Copying or other
    5:S           reproduction of this program except for archival
    6:S           purposes is prohibited without the prior
    7:S           written consent of Hewlett-Packard Company.
    8:S
    9:S
   10:S                        RESTRICTED RIGHTS LEGEND
   11:S
   12:S           Use, duplication, or disclosure by the Government
   13:S           is subject to restrictions as set forth in
   14:S           paragraph (b) (3) (B) of the Rights in Technical
   15:S           Data and Computer Software clause in
   16:S           DAR 7-104.9(a).
   17:S
   18:S           HEWLETT-PACKARD COMPANY
   19:D         0 Fort Collins, Colorado                       *)
   20:S
   21:S
   22:D         0 $modcal$
   23:D         0 $debug off, range off, ovflcheck off$
   24:D         0 $stackcheck off, iocheck off$
   25:S
   26:D         0 $search 'GPIODVR', 'IOLIB:COMASM', 'IOLIB:KERNEL'$
   27:S
```



```
   28:D         0 $page$
   29:D         0 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'$
   30:S
   31:D         0 program F9885init;
   32:S
   33:S
   34:D         1 module F9885dvr;
   35:S
   36:D         1 import
   37:D         1   sysglobals, mini, gp, iodeclarations, iocomasm, misc;
   38:S
   39:D         1 export
   40:D         1   procedure F9885io (fp: fibp; request: amrequesttype; anyvar buffer: window;
   41:D         2                      length, position: integer);
   42:D         1 implement
   43:S
   44:S
   45:D         1 procedure F9885io;
   46:S
   47:D         2   type
   48:D         2     errors = (noerror, nopower, dooropen, nodisc, badcommand, norecord,
   49:D         2               notrack, badcheckword, dataoverrun, badverify);
   50:S
   51:D         2     primarycommands = (readblock, verifyblock, writeblock, settracksector);
   52:S
   53:D         2     fd = {floppy disc command & status structure}
   54:D         2       packed record case integer of
   55:D         2         -1: (w: shortint);
   56:D         2          0: (case primary: primarycommands of
   57:D         2                readblock, verifyblock, writeblock:
   58:D         2                   (drv: 0..3; nrecords: 0..4095);
   59:D         2                settracksector:
   60:D         2                   (driv: 0..3; track: 0..127; sector: 0..31));
   61:D         2              1: (pad: 0..15; errcode: errors; p2, transfercomplete,
   62:D         2                  seekcomplete, notready, writeprotected, dooropened: boolean;
   63:D         2                  drve: 0..3);
   64:D         2       end;
   65:S
   66:D         2     gpio_enable_type = packed array[0..1] of gpio_r3_type;
   67:S
   68:D         2   const
   69:D         2     maxtries = 10;
   70:D         2     password = -20857;
   71:S
   72:D         2     gpio_enable = {gpio enable bytes for the 2 DMA channels}
   73:D         2       gpio_enable_type
   74:D         2         [ gpio_r3_type
   75:D         2             [ wenab: false, w3pad:0, wword:true, wdmac1:false, wdmac0:true ],
   76:D         2           gpio_r3_type
   77:D         2             [ wenab: false, w3pad:0, wword:true, wdmac1:true, wdmac0:false ] ];
   78:D         2   var
   79:D    -4   2     uep: ^unitentry;
   80:D    -8   2     gptr: ^gpiotype;
   81:D   -12   2     tptr: pio_tmp_ptr;
   82:D   -16   2     bufptr: charptr;
```

```
    83:D   -16  2 $page$
    84:S
    85:D        2   procedure clear_unit;
    86:C        3     begin
    87:C        3       with gptr^ do
    88:C        4         if sti1 or sti0 then
    89:C        5           ioresc(znodevice);
    90:C        3       gpioclear(gptr^);          {also tests psts while waiting for ready}
    91:C        3     end;
    92:S
    93:S
    94:D        2   procedure clear_and_escape(escape_value: shortint; iores_value: integer);
    95:C        3     begin {clear_and_escape}
    96:C        3       try
    97:C        4         gpioclear(gptr^);
    98:C        4       recover
    99:C        4         {do nothing};
   100:C        3       ioresult := iores_value;
   101:C        3       escape(escape_value);
   102:C        3     end; {clear_and_escape}
   103:S
   104:S
   105:D        2   procedure transfer(record_addr, total_words: integer);
   106:D        3     var
   107:D        3       gpiodma_proc: procedure(var gpio: gpiotype;
   108:D        3                               command: shortint; enable_byte: gpio_r3_type;
   109:D        3                               var dma_channel: dmachanneltype;
   110:D     -8 3                               buffer: charptr; length: integer);
   111:D    -12 3       status, opcode: fd;
   112:D    -18 3       chan, tries, sectors: shortint;
   113:D    -22 3       words: integer;
   114:D    -22 3     const
   115:D    -22 3       request_status = fd
   116:D    -22 3         [ primary: settracksector, driv: 0, track: 127, sector: 31 ];
   117:C        3     begin {transfer}
   118:S
   119:C        3       if not dma_here then ioresc(zbaddma);
   120:S
   121:C        3       gptr^.r3 := 0;                    {setup gpio card}
   122:C        3       gptr^.r7 := 0;
   123:S
   124:C        3       gpiowordout(gptr^, password);     {issue password}
   125:C        3       opcode := request_status;
   126:C        3       opcode.driv := uep^.du;
   127:C        3       gpiowordout(gptr^, opcode.w);     {issue request status command}
   128:C        3       gpiowordout(gptr^, 0);            {clear output regs & request data word}
   129:C        3       status.w := gpiowordin(gptr^);    {input status word}
   130:S
   131:C        3       if (status.drve<>uep^.du) or (status.pad<>0) then
   132:C        4         clear_and_escape(-10, ord(zcatchall));
   133:S
   134:C        3       if status.dooropened then
   135:C        4         begin
   136:C        4           uep^.umediavalid := false;
   137:C        4           if uep^.ureportchange then
   138:C        5             ioresc(zmediumchanged);
   139:C        4         end; {if}
```



```
   140:C        3 $page$
   141:S
   142:C        3       tries := 0;
   143:C        3       while total_words>0 do
   144:C        4         begin
   145:C        4           try
   146:C        5             gpiowordout(gptr^, password);
   147:C        5             opcode.primary := settracksector;
   148:C        5             opcode.driv    := uep^.du;
   149:C        5             opcode.track   := record_addr div 30;
   150:C        5             opcode.sector  := record_addr mod 30;
   151:C        5             gpiowordout(gptr^, opcode.w);
   152:S
   153:C        5             repeat
   154:C        6               chan := dma_request(tptr);
   155:C        6             until chan>=0;
   156:C        5             if (chan<>0) and (chan<>1) then ioresc(zcatchall);
   157:S
   158:C        5             if total_words<=65536
   159:C        6               then words := total_words
   160:C        6               else words := 65536;
   161:C        5             sectors := (words+127) div 128;
   162:S
   163:C        5             gpiowordout(gptr^, password);
   164:C        5             opcode.drv := uep^.du;
   165:C        5             opcode.nrecords:= sectors;
   166:C        5             case request of
   167:C        6               readbytes, startread:
   168:C        6                 begin
   169:C        6                   opcode.primary := readblock;
   170:C        6                   gpiodma_proc := gpiodmain;
   171:C        6                 end;
   172:C        6               writebytes, startwrite:
   173:C        6                 begin
   174:C        6                   opcode.primary := writeblock;
   175:C        6                   gpiodma_proc := gpiodmaout;
   176:C        6                 end;
   177:C        6             end; {case}
   178:C        5             call(gpiodma_proc, gptr^, opcode.w, gpio_enable[chan], dma_port[chan], bufptr, words);
   179:C        5
   180:C        5             ioresc(inoerror);  {invoke proper cleanup}
   181:C        5           recover
   182:C        5             begin
   183:C        5               gptr^.r3 := 0;          {disable the gpio card}
   184:C        5               dma_release(tptr);      {release the dma resource}
   185:C        5               if (escapecode=-10) and
   186:C        6                  ( (ioresult=ord(inoerror)) or (ioresult= ord(zcatchall)) )
   187:C        6                 then ioresult := ord(inoerror)
   188:C        6                 else clear_and_escape(escapecode, ioresult);
   189:C        5             end; {recover}
   190:S
   191:C        4           with gptr^ do
   192:C        5             begin
   193:C        5               r7 := 1;                          {set the end of transfer bit}
   194:C        5               Wdata := 0;                       {clear bidirectional buffer for reading status}
   195:C        5               setpctl := 0;                     {request the status word}
   196:C        5               status.w := gpiowordin(gptr^);    {save the status word}
   197:C        5               r7 := 0;                          {clear the end of transfer bit}
   198:C        5             end; {with}
```

```
19:C         4 $page$
20:S
20:C         4           if (status.drve<>uep^.du) or (status.pad<>0) then
20:C         5             clear_and_escape(-10, ord(zcatchall));
20:S
20:C         4           with status do
20:C         5             case errcode of
20:C         6               noerror:
20:C         6                 begin
20:C         6                   if notready or (not seekcomplete) or (not transfercomplete) then
20:C         7                     clear_and_escape(-10, ord(zcatchall));
20:C         6                   tries       := 0;
21:C         6                   record_addr := record_addr+sectors;
21:C         6                   total_words := total_words-words;
21:C         6                   bufptr      := addr(bufptr^,words*2)
21:C         6                 end;
21:S
21:C         6               nopower:
21:C         6                 ioresc(znodevice);
21:S
21:C         6               dooropen, nodisc:
22:C         6                 ioresc(znomedium);
22:S
22:C         6               badcommand:
22:C         6                 if writeprotected and ( (request=writebytes) or (request=startwrite) )
22:C         7                   then ioresc(zprotected)
22:C         7                   else clear_and_escape(-10, ord(zcatchall));
22:S
22:C         6               notrack:
22:C         6                 ioresc(znoblock);
22:S
23:C         6               norecord, badcheckword:
23:C         6                 begin
23:C         6                   tries := tries+1;
23:C         6                   if tries>=maxtries then
23:C         7                     begin
23:C         7                       if errcode=norecord then ioresc(znoblock);
23:C         7                       if errcode=badcheckword then ioresc(zbadblock);
23:C         7                       ioresc(zcatchall);
23:C         7                     end; {if}
23:C         6                 end;
23:S
24:C         6               dataoverrun:
24:C         6                 ioresc(zbadhardware);
24:S
24:C         6               otherwise
24:C         6                 clear_and_escape(-10, ord(zcatchall));
24:C         6             end; {case}
24:S
24:C         4         end; {while}
24:S
25:C         3     end; {transfer}
```



```
251:D   -16  2 $page$
252:S
253:C        2   begin (F9885io)
254:C        2     uep := addr(unitable^[fp^.funit]);
255:C        2     if uep^.offline then ioresult := ord(znodevice)
256:C        3     else
257:C        3       begin
258:C        3         lockup;
259:C        3         try
260:C        4           with isc_table[uep^.sc] do
261:C        5             begin
262:C        5               if card_id<>hp98622 then ioresc(znodevice);
263:C        5               gptr := card_ptr;
264:C        5               tptr := io_tmp_ptr;
265:C        5             end; {with}
266:S
267:C        4           case request of
268:C        5             clearunit:
269:C        5               clear_unit;
270:S
271:C        5             unitstatus:
272:C        5               fp^.fbusy := false;
273:S
274:C        5             flush:
275:C        5               {do nothing};
276:S
277:C        5             readbytes, writebytes, startread, startwrite:
278:C        5               begin
279:C        5                 if uep^.ureportchange and not uep^.umediavalid then
280:C        6                   ioresc(zmediumchanged);
281:C        5                 bufptr := addr(buffer);
282:C        5                 if (position mod 256<>0) or odd(integer(bufptr)) then
283:C        6                   ioresc(zbadmode);
284:C        5                 if (position<0) or (length<0) or (position+length>fp^.fpeof) then
285:C        6                   ioresc(ieof);
286:C        5                 transfer((position+fp^.fileid+uep^.byteoffset) div 256, (length+1) div 2);
287:C        5               end;
288:S
289:C        5             otherwise
290:C        5               ioresc(ibadrequest);
291:C        5           end; {cases}
292:S
293:C        4           ioresc(inoerror);  {set ioresult & perform lockdown}
294:C        4         recover
295:C        4           begin
296:C        4             lockdown;
297:C        4             if escapecode<>-10 then escape(escapecode);
298:C        4             if (request=startread) or (request=startwrite) then call(fp^.feot, fp);
299:C        4           end; {recover}
300:C        3       end; {else}
301:C        2   end; {f9885io}
302:S
303:C        1 end; {f9885dvr}
304:S
305:S
306:D        1 { program F9885init }
307:S
308:D        1 import
309:D        1   loader;
310:S
```

```
   311:C         1 begin {F9885init}
   312:C         1   markuser;
   313:C         1 end. {F9885init}
   314:S
   315:S

No errors. No warnings.
              ***** Nonstandard language features enabled *****
```

Description

GCRT contains high-level code for the high-resolution 9837A CRT, including the TM, the debugger window handler, and the dump graphics procedure.

Requirements

SYSGLOBALS, ASM, MISC, and SYSDEVS.

Notes

This module is linked with GASSM (in the assembly listings) to produce the INITLIB module CRTB.

```
    1:S                                                 (*
    2:S
    3:S           (c) Copyright Hewlett-Packard Company, 1984.
    4:S           All rights are reserved.  Copying or other
    5:S           reproduction of this program except for archival
    6:S           purposes is prohibited without the prior
    7:S           written consent of Hewlett-Packard Company.
    8:S
    9:S                       RESTRICTED RIGHTS LEGEND
   10:S
   11:S           Use, duplication, or disclosure by the Government
   12:S           is subject to restrictions as set forth in
   13:S           paragraph (b) (3) (B) of the Rights in Technical
   14:S           Data and Computer Software clause in
   15:S           DAR 7-104.9(a).
   16:S
   17:S           HEWLETT-PACKARD COMPANY
   18:S         0 Fort Collins, Colorado                        *)
   19:S
   20:S
   21:S         0 $modcal$
   22:D         0 $heap_dispose off$
   23:D         0 $iocheck off$
   24:D         0 $range off$ $ovflcheck off$
   25:D         0 $stackcheck off$
   26:D         0 $search  'INITLOAD','ASM','INIT','SYSDEVS'$
   27:D
   28:S         0 program initcrtb;
   29:D
   30:S
   31:S         1 module crtb;
   32:D         1 import sysglobals, asm, misc, sysdevs;
   33:D         1 export
   34:D
   35:S         1 function gatorcrttype: boolean;
   36:D
   37:S         1 implement
   38:D         1
   39:D         1 const
   40:D         1
   41:D
   42:S         1   environc=environ[miscinfo:crtfrec[
   43:D         1                                       nobreak:false,
   44:D         1                                       stupid :false,
   45:D         1                                       slowterm:false,
   46:D         1                                       hasxycrt:true,
   47:D         1                                       haslccrt:FALSE,  {INDICATES BITMAP}
   48:D         1                                       hasclock:true,
   49:D         1                                       canupscroll:true,
   50:D         1                                       candownscroll:true],
   51:D         1                       crttype:0,
   52:D         1                       crtctrl:crtcrec[
   53:D         1                                       rlf:chr(31),
   54:D         1                                       ndfs:chr(28),
   55:D         1                                       eraseeol:chr(9),
   56:D         1                                       eraseeos:chr(11),
   57:D         1                                       home:chr(1),
   58:D         1                                       escape:chr(0),
   59:D         1                                       backspace:chr(8),
   60:D
```



```
   61:D         1                                       fillcount:10,
   62:D         1                                       clearscreen:chr(0),
   63:D         1                                       clearline:chr(0),
   64:D         1                                       prefixed:b9[9 of false]],
   65:D         1                       crtinfo:crtirec[
   66:D         1                                       width :128,height:47,
   67:D         1                                       crtmemaddr:0,
   68:D         1                                       crtcontroladdr:0,
   69:D         1                                       keybufferaddr: 0,
   70:D         1                                       progstateinfoaddr: 0,
   71:D         1                                       keybuffersize: 119,
   72:D         1                                       crtcon: crtconsttype [ 0, 0, 0, 0, 0, 0, 0, 0,
   73:D         1                                                              0,0, 0, 0,0],
   74:D         1                                       right(FS):chr(28),
   75:D         1                                       left(BS):chr(8),
   76:D         1                                       down(LF):chr(10),    up(US):chr(31),
   77:D         1                                       badch(?):chr(63),
   78:D         1                                       chardel(BS):chr(8),stop(DC3) :chr(19),
   79:D         1                                       break(DLE):chr(16),
   80:D         1                                       flush(ACK):chr(6),  eof(ETX):chr(3),
   81:D         1                                       altmode(ESC):chr(27),
   82:D         1                                       linedel(DEL):chr(127),
   83:D         1                                       backspace(BS):chr(8),
   84:D         1                                       etx:chr(3),prefix:chr(0),
   85:D         1                                       prefixed:bi4[14 of false],
   86:D         1                                       cursormask : 0,     spare : 0]];
   87:S
   88:S
   89:S
   90:S
   91:D         1 type
   92:S
   93:D         1 scrtype = packed array[0..maxint] of crtword;
   94:D         1 scrptr=^scrtype;
   95:S
   96:S
   97:D         1 crtregtype = 0..15;
   98:D         1 crtcmdwrd = packed record case integer of
   99:D         1                  0: (topbyte, botbyte: byte);
  100:D         1                  1: (longword: shortint);
  101:D         1                  2: (p1,p2, textfield, softfield: boolean);
  102:D         1                end;
  103:S
  104:S
  105:D         1 var
  106:S
  107:D    -4   1 cursoraddr: integer;
  108:D    -6   1 screenwidth: shortint;
  109:D    -8   1 screenheight:shortint;
  110:D   -10   1 maxx: shortint;
  111:D   -12   1 maxy: shortint;
  112:D   -14   1 screensize:shortint;
  113:D   -16   1 defaulthighlight: shortint;
  114:D   -18   1 highlight: shortint;
  115:S
  116:S
  117:D   -18   1 procedure  cchar(c,x,y:shortint);external;
  118:D   -18   1 procedure  changecursor; external;
  119:D   -18   1 procedure  cscrollup;external;
  120:D   -18   1 procedure  cscrolldown;external;
```

```
   121:D   -18  1 procedure  cclear(x,y,n:shortint);external;
   122:D   -18  1 procedure  cupdatecursor(x,y:shortint);external;
   123:D   -18  1 procedure  cbuildtable;external;
   124:D   -18  1 procedure  cshiftleft; external;
   125:D   -18  1 procedure  cshiftright; external;
   126:D        1 procedure  cexchange( savearea: windowp; ymin, ymax, xmin, width: shortint);
   127:D   -18  2           external;
   128:D   -18  1 procedure  cscrollwindow( ymin, ymax, xmin, width: shortint); external;
   129:D   -18  1 procedure  cscrollwinddn( ymin, ymax, xmin, width: shortint); external;
   130:D   -18  1 procedure  cdbscrolll( ymin, ymax, xmin, width: shortint); external;
   131:D   -18  1 procedure  cdbscrollr( ymin, ymax, xmin, width: shortint); external;
   132:D   -18  1 procedure  cdbhighl( c, x, y: shortint); external;
   133:D   -18  1
   134:D        1 procedure dumpg ;
   135:S
   136:S
   137:D        2 label 1;
   138:S
   139:D        2 const
   140:D        2   gwidthb = 128;
   141:D        2   gmaxheight = 512;
   142:D        2   gbuffersize = gwidthb + 7;
   143:S
   144:D        2 type
   145:D        2   gbyte = 0..255;
   146:D        2   row_def = packed array [0..(1024*768)-1] of gbyte;
   147:S
   148:D        2 var
   149:D    -4  2   row : ^row_def;
   150:S
   151:D  -140  2   gbuffer : packed array [1..gbuffersize] of char;
   152:D  -148  2   i,j : integer;
   153:D  -152  2   index : integer;
   154:D  -156  2   bit_mask : integer;
   155:D  -160  2   result : integer;
   156:S
   157:C        2 begin
   158:S
   159:C        2   row := anyptr(frameaddr);
   160:S
   161:C        2   write(gfiles[4]^,#27'*rA');   { initiate graphics sequence }
   162:S
   163:C        2   gbuffer[1] := chr(27); { escape sequence for graphics }
   164:C        2   gbuffer[2] := '*';
   165:C        2   gbuffer[3] := 'b';
   166:C        2   gbuffer[4] := '1';
   167:C        2   gbuffer[5] := '2';
   168:C        2   gbuffer[6] := '8';
   169:C        2   gbuffer[7] := 'W';
   170:S
   171:C        2   for j := 0 to 767 do
   172:C        3     begin
   173:C        3       for i := 0 to 127 do
   174:C        4         begin
   175:C        4           result := 0;
   176:C        4           index := j*1024+i*8;
   177:C        4           bit_mask := 256;
   178:C        4           for index := index to index+7 do
   179:C        5             begin
   180:C        5               bit_mask := bit_mask div 2;
```



```
   181:C        5               if odd(row^[index])  then result := bit_mask+result;
   182:C        5             end;
   183:C        4           gbuffer[i+8] := chr(result);
   184:C        4         end;
   185:C        3         write(gfiles[4]^,gbuffer:gwidthb+7);
   186:C        3         if ioresult <> ord(inoerror) then goto 1;
   187:C        3     end;
   188:S
   189:C        2   write(gfiles[4]^,#27'*rB');   { terminate graphics sequence }
   190:C        2 1:
   191:C        2 end;
   192:S
   193:S
   194:S
   195:S
   196:D        1 procedure doupdatecursor;
   197:D    -2  2 var cursaddr: crtcmdwrd;
   198:C        2 begin
   199:C        2   cupdatecursor(xpos,ypos);
   200:C        2 end;
   201:S
   202:D        1 procedure getxy(var x,y: integer);
   203:C        2 begin
   204:C        2 x := xpos;      y := ypos;
   205:C        2 end;
   206:S
   207:D        1 procedure setxy(x, y: shortint);
   208:C        2 begin
   209:C        2   if x>=screenwidth then xpos:=maxx
   210:C        3   else if x<0 then xpos:=0
   211:C        4   else xpos := x;
   212:C        2   if y>=screenheight then ypos:=maxy
   213:C        3   else if y<0 then ypos:=0
   214:C        4   else ypos := y;
   215:C        2 end;
   216:S
   217:D        1 procedure gotoxy(x,y: integer);
   218:C        2 begin
   219:C        2   setxy(x,y);
   220:C        2   doupdatecursor;
   221:C        2 end;
   222:S
   223:S
   224:D        1 procedure clear(number: shortint);  { REVISED FOR 3.01 }
   225:D    -4  2 var x,y: shortint;
   226:D    -6  2     clearchars: shortint;
   227:C        2 begin
   228:C        2   x:=xpos; y:=ypos;
   229:C        2   while number>0 do begin
   230:C        3     if maxx-x+1<number then
   231:C        4       clearchars:=maxx-x+1
   232:C        4     else
   233:C        4       clearchars:=number;
   234:C        3     cclear(x,y,clearchars);
   235:C        3     number:=number-clearchars;
   236:C        3     x:=0; if y<maxy then y:=y+1;
   237:C        3   end;
   238:C        2 end;
   239:S
   240:D        1 procedure scrollup;
```

```
241:C        2 begin
242:C        2   cscrollup;
243:C        2 end;
244:S
245:D        1 procedure scrolldown;(new  4/30/81)
246:C        2 begin
247:C        2   cscrolldown;
248:C        2 end;
249:S
250:S
251:D        1 function maptocrt(c:char):shortint;
252:S
253:S          { Converts Katakana codes to their correct CRT font storage  codes.
254:D        2   Note that the Yen symbol overlays the USASCII backslash (\). }
255:D        2 procedure mapkanatocrt ;
256:D        3 const
257:D        3   yenromlocation = 188; { location of Yen symbol in font storage }
258:D        3   yencode=92;
259:C        3 begin
260:C        3     if ord(c) = yencode then maptocrt := yenromlocation
261:C        4     else if ord(c)<128 then maptocrt:=ord(c)
262:C        5           else maptocrt:= ord(c)+128;
263:C        3 end; { mapkanatocrt }
264:S
265:C        2 begin
266:C        2   if kbdlang = katakana_kbd then mapkanatocrt
267:C        3   else maptocrt:=ord(c);
268:C        2 end;
269:S
270:S
271:S
272:D        1 procedure docrtio(fp: fibp; request: amrequesttype; anyvar buffer: window;
273:D        2                                                    length, position: integer);
274:D    -1  2 var c: char;
275:D    -4  2     s: string[1];
276:D    -8  2     buf: charptr;
277:C        2 begin
278:C        2  ioresult := ord(inoerror);
279:C        2  buf := addr(buffer);
280:C        2  case request of
281:C        3   {uwait: ;                }
282:C        3   setcursor: gotoxy(fp^.fxpos, fp^.fypos);
283:C        3   getcursor: getxy (fp^.fxpos, fp^.fypos);
284:C        3   flush:  {do nothing};
285:C        3   uritstatus:  kbdio(fp, request, buffer, length, position);
286:C        3   clearunit: highlight := defaulthighlight;
287:C        3   readtoeol:
288:C        3     begin
289:C        3     buf := addr(buf^, 1);
290:C        3     buffer[0] := chr(0);
291:C        3     while length>0 do
292:C        4       begin
293:C        4       kbdio(fp, readtoeol,  s, 1, 0);
294:C        4       if     strlen(s)=0     then length := 0
295:C        5     { else if s[1] = chr(etx) then length := 0 }
296:C        5       else  begin
297:C        5             length := length - 1;
298:C        5             crtio(fp, writebytes, s[1], 1, 0);
299:C        5             buf := addr(buf^, 1);
300:C        5             buffer[0] := chr(ord(buffer[0])+1);
```



```
301:C        5             end;
302:C        4       end;
303:C        3     end;
304:C        3   startread,
305:C        3   readbytes:
306:C        3     begin
307:C        3     while length>0 do
308:C        4       begin
309:C        4       kbdio(fp, readbytes,  buf^, 1, 0);
310:C        4       if buf^ = chr(etx) then length := 0
311:C        5                          else length := length - 1;
312:C        4       if buf^ = eol then crtio(fp, writeeol,   buf^, 1, 0)
313:C        5                     else crtio(fp, writebytes, buf^, 1, 0);
314:C        4       buf := addr(buf^, 1);
315:C        4       end;
316:C        3     if request = startread then call(fp^.feot, fp);
317:C        3     end;
318:C        3   writeeol:   begin
319:C        3                 if ypos=maxy then scrollup;
320:C        3                 gotoxy(0, ypos+1);
321:S
322:C        3               end;
323:C        3   startwrite,
324:C        3   writebytes:
325:C        3     begin
326:C        3     while length>0 do
327:C        4      begin
328:C        4        c:=buf^; buf:=addr(buf^,1); length:=length-1;
329:S
330:C        4        case c of
331:C        5        homechar:   setxy(0,0);
332:C        5        leftchar:   if (xpos = 0) and (ypos>0) then setxy(maxx, ypos-1)
333:C        6                    else setxy(xpos-1, ypos);
334:C        5        rightchar:  if (xpos = maxx) and (ypos<maxy) then setxy(0, ypos+1)
335:C        6                    else setxy(xpos+1, ypos);
336:C        5        upchar:     begin if ypos <= 1  then scrolldown;
337:C        5                          if ypos>0 then setxy(xpos, ypos-1);
338:C        5                    end;
339:C        5        downchar:   if ypos=maxy then scrollup
340:C        6                    else setxy(xpos, ypos+1);
341:C        5        bellchar:   beep;
342:C        5        cteos:     clear(screensize-(ypos*screenwidth+xpos));
343:C        5        cteol:     clear(screenwidth-xpos);
344:C        5        clearscr:  begin setxy(0,0); clear(screensize); end;
345:C        5        eol:       setxy(0, ypos);
346:C        5        chr(etx):   length:=0;
347:C        5        otherwise   if (ord(c)>=128) and (ord(c)<144) then
348:C        6                      highlight:= (ord(c)-128)*256
349:C        6                    else
350:C        6                      begin
351:C        6                       changecursor;
352:C        6                       cchar(maptocrt(c),xpos,ypos);
353:C        6                       changecursor;
354:C        6                       if xpos = maxx then
355:C        7                         begin
356:C        7                           if ypos = maxy then scrollup;
357:C        7                           setxy(0, ypos+1);
358:C        7                         end
359:C        7                       else setxy(xpos+1, ypos);
360:C        6                     end;
```

```
  361:C        5        end;
  362:C        4      doupdatecursor;
  363:C        4      end; (while)
  364:C        3     if request = startwrite then call(fp^.feot, fp);
  365:C        3     end;
  366:C        3   otherwise ioresult := ord(ibadrequest);
  367:C        3  end; (case)
  368:C        2 end;
  369:S
  370:D        1 procedure lineops(op: crtllops; anyvar position: integer; c: char);
  371:S
  372:D    -4  2 var i,j: shortint;
  373:D    -8  2     sptr: ^string255;
  374:S
  375:C        2 begin
  376:S
  377:C        2 j:=highlight; highlight:=defaulthighlight;
  378:S
  379:C        2 case op of
  380:S
  381:C        3   cllput: cchar(maptocrt(c), position, screenheight);
  382:S
  383:C        3   cllshiftl:
  384:C        3      begin
  385:C        3        cshiftleft;
  386:C        3        cchar(ord(' '), maxx-8, screenheight);
  387:C        3      end;
  388:S
  389:C        3   cllshiftr:
  390:C        3      begin
  391:C        3        cshiftright;
  392:C        3        cchar(ord(' '), 0, screenheight);
  393:C        3      end;
  394:S
  395:C        3   cllclear: cclear(0, screenheight, maxx-7);
  396:S
  397:C        3   clldisplay:
  398:C        3      begin
  399:C        3        sptr:=addr(position);
  400:C        3        for i:=1 to strlen(sptr^) do
  401:C        4          cchar(maptocrt(sptr^[i]), i-1, screenheight);
  402:C        3        for i:=strlen(sptr^) to (maxx-8) do
  403:C        4          cchar(ord(' '), i, screenheight);
  404:C        3      end;
  405:S
  406:C        3   putstatus:  cchar(ord(c), maxx-7+position, screenheight);
  407:S
  408:C        3 end; ( of case )
  409:C        2 highlight:=j;
  410:S
  411:C        2 end;
  412:S
  413:D        1 procedure crtdebug(op: dbcrtops; var dbrec: dbcinfo);
  414:S
  415:D        2 type
  416:D        2   iptr = ^iarray;
  417:D        2   iarray = array[0..maxint] of shortint;
  418:S
  419:D    -2  2 var i: shortint;
  420:D    -6  2     j: integer;
```



```
  421:D   -10  2     tempaddr: integer;
  422:S
  423:C        2 begin
  424:C        2 with dbrec do begin
  425:C        3   case op of
  426:S
  427:C        4     dbinfo: savesize:=(xmax-xmin+1)*(ymax-ymin+1)*128; {assumes 8x14 char}
  428:S
  429:C        4     dbgotoxy: cupdatecursor(cursx, cursy);
  430:S
  431:C        4     dbscrollup: cscrollwindow( ymin, ymax, xmin, xmax-xmin+1);
  432:S
  433:C        4     dbscrolldn: cscrollwinddn(ymin, ymax, xmin, xmax-xmin+1);
  434:S
  435:C        4     dbscrolll: begin
  436:C        4                  cdbscrolll(ymin, ymax, xmin, xmax-xmin+1);
  437:C        4                  changecursor;
  438:C        4                  for i:=ymin to ymax do
  439:C        5                    cchar (ord(' '), xmax, i);
  440:C        4                  changecursor;
  441:C        4                end;
  442:S
  443:C        4     dbscrollr: begin
  444:C        4                  cdbscrollr(ymin, ymax, xmin, xmax-xmin+1);
  445:C        4                  changecursor;
  446:C        4                  for i:=ymin to ymax do
  447:C        5                    cchar (ord(' '), xmin, i);
  448:C        4                  changecursor;
  449:C        4                end;
  450:S
  451:C        4     dbhighl:  cdbhighl( ord(c), cursx, cursy);
  452:S
  453:C        4     dbput: begin
  454:C        4              changecursor;
  455:C        4              i:=highlight; highlight:=defaulthighlight;
  456:C        4              cchar( ord(c), cursx, cursy);
  457:C        4              highlight:=i;
  458:C        4              changecursor;
  459:C        4            end;
  460:S
  461:S
  462:C        4     dbclear:
  463:C        4       for i:=ymin to ymax do
  464:C        5         cclear( xmin, i, xmax-xmin+1);
  465:S
  466:C        4     dbcline: cclear( cursx, cursy, xmax-cursx+1);
  467:S
  468:C        4     dbinit:
  469:C        4       begin
  470:C        4         for j:= 0 to (savesize div 2)-1 do
  471:C        5           iptr(savearea)^[j]:=0;
  472:C        4         cursx:=xmin; cursy:=ymin;
  473:C        4         dcursoraddr:=frameaddr;
  474:C        4         areaisdbcrt:=true;
  475:C        4       end;
  476:S
  477:C        4     dbexcg:
  478:C        4       begin
  479:C        4         changecursor;
  480:C        4         cexchange( savearea, ymin, ymax, xmin, (xmax-xmin+1)*8);
```

```
   481:C         4          tempaddr:=cursoraddr;
   482:C         4          cursoraddr:=dcursoraddr;
   483:C         4          dcursoraddr:=tempaddr;
   484:C         4          changecursor;
   485:C         4          areaisdbcrt:=not areaisdbcrt;
   486:C         4        end;
   487:S
   488:C         4
   489:C         4    end; { of case }
   490:C         3  end; { of with }
   491:C         2  end; { crtdebug procedure }
   492:S
   493:D         1  procedure dummy;
   494:C         2  begin end;
   495:S
   496:S
   497:D         1  procedure gatorcrtinit;
   498:D    -2   2   var i: shortint;
   499:S
   500:C         2   begin
   501:C         2   idle:=245;                          { set io char to roman8 value }
   502:C         2   with syscom^.crtinfo do begin
   503:C         3     screenwidth:=width;
   504:C         3     screenheight:=height;
   505:C         3     maxx:=screenwidth-1;
   506:C         3     maxy:=screenheight-1;
   507:C         3     screensize:=screenwidth*screenheight;
   508:C         3     cbuildtable;
   509:C         3     highlight:=0; defaulthighlight:=0;
   510:C         3     gotoxy(0,0);
   511:C         3     dumpalphahook := dumpg;
   512:C         3     dumpgraphicshook := dumpg;
   513:C         3     updatecursorhook:=doupdatecursor;
   514:C         3     dbcrthook:=crtdebug;
   515:C         3     crtllhook:=lineops;
   516:C         3     crtiohook:=docrtio;
   517:C         3     crtinithook:=gatorcrtinit;
   518:C         3     togglealphahook:=dummy;
   519:C         3     togglegraphicshook:=dummy;
   520:C         3     currentcrt:=bitmaptype;
   521:C         3     keybuffer^.maxsize:=maxx-8;
   522:C         3   end;
   523:C         2   end;
   524:D   -18   1
   525:S
   526:S
   527:D         1  function gatorcrttype:boolean;
   528:D         2  const gatorid=25;
   529:D    -4   2  var ptr: ^shortint;
   530:D    -6   2      i: shortint;
   531:D    -8   2      dummy: shortint;
   532:D    -9   2      found: boolean;
   533:S
   534:S
   535:C         2  begin
   536:C         2  bitmapaddr:=0;
   537:C         2  found:=false;
   538:C         2  ptr:=anyptr(hex('560000'));
   539:S
   540:C         2  try
```



```
   541:C         3    dummy:=ptr^;
   542:C         3    if (dummy mod 128) = gatorid then begin
   543:C         4      found:=true; bitmapaddr:=integer(ptr);
   544:C         4    end;
   545:C         3  recover
   546:C         3    if escapecode<>-12 then escape(escapecode);
   547:S
   548:S
   549:C         2  {********** NOTE COMMENTED OUT CODE BELOW *********}
   550:S            {ptr:=anyptr(hex('680000'));
   551:S            i:=0;
   552:S            while (i<=31) and not found do begin
   553:S              try
   554:S                dummy:=ptr^;
   555:S                if (dummy mod 128) = gatorid then begin
   556:S                  found:=true; bitmapaddr:=integer(ptr);
   557:S                end;
   558:S              recover
   559:S                if escapecode<>-12 then escape(escapecode);
   560:S
   561:S              ptr:=anyptr(integer(ptr)+65536);
   562:S              i:=i+1;
   563:C         2  end; }
   564:S
   565:C         2  gatorcrttype:=found;
   566:S
   567:C         2  if  found  then begin
   568:C         3    syscom^:=environc;
   569:C         3    gatorcrtinit;
   570:C         3  end;
   571:S
   572:C         2  end;
   573:S
   574:S
   575:S
   576:S
   577:S
   578:C         1  end;  { of module -- I hope }
   579:S
   580:D         1  import crtb, loader;
   581:S
   582:C         1  begin
   583:C         1    if gatorcrttype then markuser;
   584:C         1  end.
   585:S

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

Description

G_DRV provides low-level driver support.

Usage

G_DRV is used by the 98622 interface.

Requirements

GPIO and the I/O library kernel (IODECLARATIONS, etc.)

```
     1:S                                                    (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1983.
     4:S           All rights are reserved.  Copying or other
     5:S           reproduction of this program except for archival
     6:S           purposes is prohibited without the prior
     7:S           written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                        RESTRICTED RIGHTS LEGEND
    11:S
    12:S           Use, duplication, or disclosure by the Government
    13:S           is subject to restrictions as set forth in
    14:S           paragraph (b) (3) (B) of the Rights in Technical
    15:S           Data and Computer Software clause in
    16:S           DAR 7-104.9(a).
    17:S
    18:S           HEWLETT-PACKARD COMPANY
    19:D         0 Fort Collins, Colorado                      *)
    20:S
    21:S
    22:D         0 $SEARCH 'IOLIB:KERNEL.CODE'$
    23:D         0 $MODCAL ON$
    24:D         0 $PARTIAL_EVAL ON$
    25:D         0 $STACKCHECK ON$
    26:D         0 $RANGE OFF$
    27:D         0 $DEBUG OFF$
    28:D         0 $OVFLCHECK OFF$
```



```
    29:D         0 $PAGE$
    30:S
    31:D         0 (*************************************************************************)
    32:D         0 (*                                                                       *)
    33:D         0 (*      not RELEASED        VERSION         3.0                          *)
    34:D         0 (*                                                                       *)
    35:D         0 (*************************************************************************)
    36:D         0 (*                                                                       *)
    37:D         0 (*                                                                       *)
    38:D         0 (*      IOLIB           GPIO_DRIVERS                                     *)
    39:D         0 (*                                                                       *)
    40:D         0 (*                                                                       *)
    41:D         0 (*************************************************************************)
    42:D         0 (*                                                                       *)
    43:D         0 (*                                                                       *)
    44:D         0 (*      library       -  IOLIB                                           *)
    45:D         0 (*      name          -  GPIO_DRIVERS                                    *)
    46:D         0 (*      module(s)     -  init_gpio                                       *)
    47:D         0 (*                    -  extg                                            *)
    48:D         0 (*                                                                       *)
    49:D         0 (*      author        -                                                  *)
    50:D         0 (*      phone         -                                                  *)
    51:D         0 (*                                                                       *)
    52:D         0 (*      date          -  June  1 , 1981                                  *)
    53:D         0 (*      update        -  Aug   1 , 1983                                  *)
    54:D         0 (*      release       -  ????????????                                    *)
    55:D         0 (*                                                                       *)
    56:D         0 (*      source        -  IOLIB:G_DRV.TEXT                                *)
    57:D         0 (*      object        -  IOLIB:G_DRV.CODE                                *)
    58:D         0 (*                                                                       *)
    59:D         0 (*************************************************************************)
```

```
    60:D        0 $PAGE$
    61:D        0 (*************************************************************************)
    62:D        0 (*                                                                       *)
    63:D        0 (*                                                                       *)
    64:D        0 (*      BUG FIX HISTORY          - after release 1.0                     *)
    65:D        0 (*                                                                       *)
    66:D        0 (*                                                                       *)
    67:D        0 (*      BUG #   BY  / ON         LOC             DESCRIPTION            *)
    68:D        0 (*      -----   ------------     --------------  -----------------------*)
    69:D        0 (*                                                                       *)
    70:D        0 (*      367     -----------      gpio_initialize Allow execute of driver *)
    71:D        0 (*              09/22/82                         and have it install    *)
    72:D        0 (*                                               itself in the system.  *)
    73:D        0 (*                                                                       *)
    74:D        0 (*************************************************************************)
```



```
    75:D        0 $PAGE$
    76:D        0 (*************************************************************************)
    77:D        0 (*                                                                       *)
    78:D        0 (*                                                                       *)
    79:D        0 (*      This is the source code for an external procedures library       *)
    80:D        0 (*      to be used for general purpose interfacing on the HP 9826.       *)
    81:D        0 (*                                                                       *)
    82:D        0 (*      The library consists of 3 primary sets of modules -              *)
    83:D        0 (*                                                                       *)
    84:D        0 (*              1.      KERNEL modules                                   *)
    85:D        0 (*              2.      driver modules                                   *)
    86:D        0 (*              3.      IOLIB  modules                                   *)
    87:D        0 (*                                                                       *)
    88:D        0 (*      The KERNEL modules consist of the following modules -            *)
    89:D        0 (*                                                                       *)
    90:D        0 (*              1.      iodeclarations  ( contains static r/w space )    *)
    91:D        0 (*              2.      iocomasm                                         *)
    92:D        0 (*              3.      general_0       ( initialization & low level     *)
    93:D        0 (*                                        routines like ioread/iowrite) *)
    94:D        0 (*      The KERNEL modules also have an executable program segement      *)
    95:D        0 (*      that gets executed at the time it is loaded.  This program       *)
    96:D        0 (*      initializes the static read/write memory.  This program also     *)
    97:D        0 (*      allocates the temporary storage for any card that exists -       *)
    98:D        0 (*      independent of whether there is or is not a driver for it.       *)
    99:D        0 (*                                                                       *)
   100:D        0 (*      The driver modules consist of the actual assembly or PASCAL      *)
   101:D        0 (*      routines that deal with a specific interface card.  There is     *)
   102:D        0 (*      also an executable program segment for each driver module.       *)
   103:D        0 (*      This program searches the select code table in the static r/w    *)
   104:D        0 (*      initialized by the KERNEL general_0 module for all select codes *)
   105:D        0 (*      that have the right interface card ( HPIB drivers will search    *)
   106:D        0 (*      for the 98624 interface ).  This program will then set up the    *)
   107:D        0 (*      driver tables to point to the correct drivers.                   *)
   108:D        0 (*                                                                       *)
   109:D        0 (*      The rest of the IOLIB modules are high-level modules that are    *)
   110:D        0 (*      used by an end user in his/her application program.              *)
   111:D        0 (*                                                                       *)
   112:D        0 (*      The KERNEL and some set of driver modules will exist in the      *)
   113:D        0 (*      SYSTEM.INITLIB file as object code ( not EXPORT text ).  The     *)
   114:D        0 (*      export text will reside on the SYSTEM.LIBRARY file.  The rest    *)
   115:D        0 (*      of the library will reside on the SYSTEM.LIBRARY.                *)
   116:D        0 (*                                                                       *)
   117:D        0 (*************************************************************************)
```

```
Pascal [Rev 3.0M  6/ 4/84] G_DRV.TEXT              14-Jan-85 16:27:34 Page 5

   118:D        0 $PAGE$
   119:D        0 (*************************************************************************)
   120:D        0 (*                                                                       *)
   121:D        0 (*                                                                       *)
   122:D        0 (*      REFERENCES :                                                     *)
   123:D        0 (*                                                                       *)
   124:D        0 (*                                                                       *)
   125:D        0 (*      1.  9826 I/O Designers Guide            ( ---------- )         *)
   126:D        0 (*                                                                       *)
   127:D        0 (*      2.  68000 Manual                        ( Motorola )           *)
   128:D        0 (*                                                                       *)
   129:D        0 (*      3.  Pascal alpha site ERS               ( ---------- )         *)
   130:D        0 (*                                                                       *)
   131:D        0 (*      4.  Pascal I/O Library ERS              ( ---------- )         *)
   132:D        0 (*                                                                       *)
   133:D        0 (*      5.  9826 HPL EIO & IOD listings         ( ---------- )         *)
   134:D        0 (*                                                                       *)
   135:D        0 (*      6.  9826 HPL Misc. I/O Doc.             ( ---------- )         *)
   136:D        0 (*                                                                       *)
   137:D        0 (*      7.  9826 card documentation             ( Mfg. Specs. )        *)
   138:D        0 (*                                                                       *)
   139:D        0 (*      8.  Pascal I/O Library IRS              ( ---------- )         *)
   140:D        0 (*                                                                       *)
   141:D        0 (*                                                                       *)
   142:D        0 (*************************************************************************)
```

```
Pascal [Rev 3.0M  6/ 4/84] G_DRV.TEXT              14-Jan-85 16:27:34 Page 6

   143:D        0 $PAGE$
   144:D        0 PROGRAM gpio_initialize ( INPUT , OUTPUT );
```

```
   145:D        1 $PAGE$
   146:D        1 (*************************************************************************)
   147:D        1 (*                                                                       *)
   148:D        1 (*                                                                       *)
   149:D        1 (*      GPIO DRIVERS                                                     *)
   150:D        1 (*                                                                       *)
   151:D        1 (*                                                                       *)
   152:D        1 (*************************************************************************)
   153:D        1 EXTERNAL MODULE extg;
   154:S                  {
   155:S                    date    08/25/81
   156:S                    update  10/13/81
   157:S
   158:S                    purpose This module is a declaration of the importation text for
   159:S                            the external drivers.
   160:S
   161:S                    note    The assembly language code that is imported needs to be
   162:S                            called 'extg'.  The routines need to be called
   163:S                            'extg_@@@@@@' - eg_init referenced below would be
   164:S                            extg_eg_init.  'eg' refers to GPIO.
   165:D        1           }
   166:S
   167:D        1 IMPORT  sysglobals ,
   168:D        1           iodeclarations ;
   169:S
   170:D        1 EXPORT
   171:S
   172:D        1   PROCEDURE eg_init  ( temp : ANYPTR );
   173:D        1   PROCEDURE eg_isr   ( temp : PISRIB );
   174:D        1   PROCEDURE eg_rdb   ( temp : ANYPTR ;  VAR x : CHAR);
   175:D        1   PROCEDURE eg_wtb   ( temp : ANYPTR ;  val   : CHAR);
   176:D        1   PROCEDURE eg_rdw   ( temp : ANYPTR ;  VAR x : io_word);
   177:D        1   PROCEDURE eg_wtw   ( temp : ANYPTR ;  val   : io_word);
   178:D        1   PROCEDURE eg_rds   ( temp : ANYPTR ;  reg   : io_word;
   179:D        2                                         VAR x : io_word);
   180:D        1   PROCEDURE eg_wtc   ( temp : ANYPTR ;  reg   : io_word;
   181:D        2                                         val   : io_word );
   182:D        1   PROCEDURE eg_tfr   ( temp : ANYPTR ;  bcb   : ANYPTR );
   183:D        1   PROCEDURE eg_clr   ( temp : ANYPTR ;  line  : io_bit );
   184:D        1   PROCEDURE eg_set   ( temp : ANYPTR ;  line  : io_bit );
   185:D        1   PROCEDURE eg_test  ( temp : ANYPTR ;  line  : io_bit ;
   186:D        2                                         VAR x : BOOLEAN );
   187:S
   188:D        1 END; { of extg }
```



```
   189:D        1 $PAGE$
   190:S
   191:S
   192:D        1 MODULE init_gpio;
   193:S                  {
   194:S                    date    08/25/81
   195:S                    update  10/04/82
   196:S                             8/01/83   change rev number
   197:S
   198:S                    purpose This module initializes the gpio drivers.
   199:S
   200:D        1           }
   201:S
   202:S
   203:D        1 IMPORT    iodeclarations ;
   204:S
   205:D        1 EXPORT
   206:S
   207:S
   208:D        1   VAR
   209:D  -120  1     gpio_drivers  : drv_table_type;
   210:S
   211:D  -120  1   PROCEDURE io_init_gpio;
   212:S
   213:S
   214:S
   215:D  -120  1 IMPLEMENT
   216:S
   217:S
   218:D  -120  1   IMPORT  sysglobals ,
   219:D  -120  1           isr ,
   220:D  -120  1           general_0 ,
   221:D  -120  1           extg ;
   222:S
   223:S
   224:D        1   PROCEDURE io_init_gpio;
   225:D        2   VAR
   226:D    -2  2     io_isc        : type_isc;
   227:D    -6  2     dummy         : INTEGER;
   228:D    -8  2     io_lvl        : io_byte;
   229:C        2   BEGIN
   230:S
   231:C        2     io_revic := io_revid + ' G3.0';     { GPIO revision added 2/5/82 TM }
   232:S
   233:C        2     { set up the driver tables }
   234:S
   235:C        2     WITH gpio_drivers DO BEGIN
   236:C        3       gpio_drivers := dummy_drivers ;
   237:C        3       iod_init  := eg_init;
   238:C        3       iod_isr   := eg_isr;
   239:C        3       iod_rdb   := eg_rdb;
   240:C        3       iod_wtb   := eg_wtb;
   241:C        3       iod_rdw   := eg_rdw;
   242:C        3       iod_wtw   := eg_wtw;
   243:C        3       iod_rds   := eg_rds;
   244:C        3       iod_wtc   := eg_wtc;
   245:C        3       iod_tfr   := eg_tfr;
   246:C        3       iod_set   := eg_set;
   247:C        3       iod_clr   := eg_clr;
   248:C        3       iod_test  := eg_test;
```

```
249:C  3     END; { of WITH }
250:S
251:S
252:C  2     { set up drivers for the interfaces }
253:S
254:C  2     FOR io_isc:=iominisc TO iomaxisc DO
255:C  3       WITH isc_table[io_isc] DO BEGIN
256:S
257:C  4         IF card_id = hp98622
258:C  5           THEN BEGIN
259:S
260:C  5             io_drv_ptr:=ADDR(gpio_drivers);
261:S
262:C  5             WITH io_drv_ptr^, io_tmp_ptr^ DO BEGIN
263:S
264:C  6               { if the card exists then link in an ISR for it }
265:C  6               { ??? - what happens if an ISR fires during init }
266:S
267:C  6               io_lvl:=((ioread_byte(io_isc,3) DIV 16) MOD 4)+3;
268:C  6               IF myisrib.INTREGADDR <> NIL
269:C  7                 THEN BEGIN
270:C  7                   { if isr exists then unlink it }
271:C  7                   ISRUNLINK(io_lvl,              { interrupt level       }
272:C  7                             addr(myisrib));      { ptr to the isrib      }
273:C  7                 END; { of IF }
274:S
275:C  6               PERMISRLINK(iod_isr,                              { isr           }
276:C  6                       ANYPTR(INTEGER(card_ptr)+3),              { int reg addr  }
277:C  6                       192,                                      { int reg mask  }
278:C  6                       192,                                      { int reg value }
279:C  6                       io_lvl,                                   { int level     }
280:C  6                       ADDR(myisrib));                           { isrib info    }
281:S
282:C  6             END; { of WITH BEGIN }
283:S
284:C  5           END; { of IF card_type = gpio_card }
285:S
286:C  4       END; { of FOR io_isc WITH isc_table[io_isc] BEGIN }
287:S
288:S
289:C  2     { call the actual driver initialization }
290:S
291:C  2     FOR io_isc:=iominisc TO iomaxisc DO
292:C  3       WITH isc_table[io_isc] DO
293:C  4       IF card_id = hp98622
294:C  5         THEN BEGIN
295:C  5           CALL(io_drv_ptr^.iod_init , io_tmp_ptr);
296:C  5         END; { of WITH IF }
297:S
298:S
299:C  2   END; { of io_init_gpio }
300:S
301:C  1 END; { of MODULE init_gpio }
```



```
302:D  1 $PAGE$
303:S
304:D  1 IMPORT    init_gpio ,
305:D  1           LOADER ;                                          { 367 TM 9/22/82 }
306:S
307:S
308:C  1 BEGIN
309:C  1   io_init_gpio;
310:C  1   MARKUSER;                                                 { 367 TM 9/22/82 }
311:C  1 END. { of gpio_initialize }

No errors. No warnings.
          ***** Nonstandard language features enabled *****
```

Description

GEN contains most of the viewing transformations and color model transformations for DGL.

Requirements

DGL_TYPES, ASM, DGL_VARS, GLE_TYPES, and GLE_GEN.

```
    1:D        0 {                                                                    }
    2:D        0 { Pascal work station graphics library                               }
    3:D        0 {                                                                    }
    4:D        0 { Module     = DGL_GEN                                               }
    5:D        0 { Programer  = BJS                                                   }
    6:D        0 { Date       = 2/1/81                                                }
    7:S
    8:D        0 { Purpose:  To hold most internal routines                           }
    9:S
   10:D        0 { Rev history                                                        }
   11:D        0 {   6-15-82  BJS - Added moonunit dump graphics stuff                }
   12:D        0 {   7-05-82  BJS - Removed HPGL clipping, now uses ASM_RAS clipping  }
   13:D        0 {   7-05-82  BJS - Changes to add 9836C proto support                }
   14:D        0 {   8-25-82  BJS - Major mods for GLE                                }
   15:S
   16:S          {      (c) Copyright Hewlett-Packard Company, 1983.
   17:S                 All rights are reserved.  Copying or other
   18:S                 reproduction of this program except for archival
   19:S                 purposes is prohibited without the prior
   20:S                 written consent of Hewlett-Packard Company.
   21:S
   22:S
   23:S                            RESTRICTED RIGHTS LEGEND
   24:S
   25:S                 Use, duplication, or disclosure by the Government
   26:S                 is subject to restrictions as set forth in
   27:S                 paragraph (b) (3) (B) of the Rights in Technical
   28:S                 Data and Computer Software clause in
   29:S                 DAR 7-104.9(a).
   30:S
   31:S                 HEWLETT-PACKARD COMPANY
   32:D        0        Fort Collins, Colorado                                 }
   33:S
   34:D        0 $medcal$
   35:D        0 $ref 50$
   36:S
   36:D        0 $include 'OPTIONS'$
   37:S
   38:S           { This include file specifies range checking, debug and other compiler
   39:D        0    options for the graphics library                                  }
   40:S
   41:D        0 $debug OFF$
   42:D        0 $range OFF$
   43:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
   44:D        0 $FLOAT_HDW TEST$
   45:S
   46:S
   47:S
   48:S
   49:D        0 $include 'OPTIONS'$
 3000:D        0 $linenum 3000$
 3001:S
 3002:D        0 $search 'TYPES',
 3003:D        0         'DGL_VARS',
 3004:D        0         'GLE_LIB'$
 3005:S
 3006:D        0 module DGL_GEN;
 3007:S
 3008:D        1 import DGL_TYPES;
 3009:S
```



```
 3010:D        1 export
 3011:S
 3012:D        1 procedure error ( error_number : integer );
 3013:S
 3014:D        1 function min(p1,p2:integer) : integer;
 3015:S
 3016:D        1 function max(p1,p2:integer) : integer;
 3017:S
 3018:D        1 procedure locator_limits (lxmin, lxmax,
 3019:D        2                           lymin, lymax : real);
 3020:S
 3021:D        1 procedure calculate_viewing;
 3022:S
 3023:D        1 procedure display_limits ( dxmin, dxmax,
 3024:D        2                            dymin, dymax : real);
 3025:S
 3026:D        1 procedure convert_intwtod (x, y : gshortint;
 3027:D        2                    var  dx, dy : integer );
 3028:S
 3029:D        1 procedure convert_wtod (x, y : real;
 3030:D        2                    var  dx, dy : integer );
 3031:S
 3032:D        1 procedure convert_ltod (lx, ly : integer;
 3033:D        2                    var  dx, dy : integer );
 3034:S
 3035:D        1 procedure convert_dtow (dx, dy : integer;
 3036:D        2                    var  wx, wy : real);
 3037:S
 3038:D        1 procedure ck_system_init;
 3039:S
 3040:D        1 procedure ck_display_init;
 3041:S
 3042:D        1 procedure ck_locator_init;
 3043:S
 3044:D        1 procedure adjust_return_echo ( var rx,ry : real );
 3045:S
 3046:D        1 function opcode_ck(opcode,num_integer,num_real : integer) : integer;
 3047:S
 3048:D        1 procedure convert_hsl_to_rgb ( hue,sat,lite : real; var r,g,b : real);
 3049:D        1 procedure convert_rgb_to_hsl (r,g,b : real; var hue,sat,lite : real);
 3050:S
 3051:D        1 implement
 3052:S
 3053:D        1 import asm,
 3054:D        1        dgl_vars,
 3055:D        1        gle_gen,
 3056:D        1        GLE_TYPES;
 3057:S
 3058:D        1 { ************************************************************************ }
 3059:D        1 {                                                                        }
 3060:D        1 {                  General procedures and functions                      }
 3061:D        1 {                                                                        }
 3062:D        1 { ************************************************************************ }
 3063:S
 3064:D        1 procedure error ( error_number : integer);
 3065:S
 3066:D        2 { Purpose : To log a graphics error, and perform a graphics error escape }
 3067:S
 3068:C        2 begin
 3069:C        2   graphics_error := error_number;
```

```
3070:C        2   escape(graphics_error_number);
3071:C        2 end; { error }
3072:S
3073:D        1 procedure ck_system_init;
3074:S
3075:D        2 { Purpose : To report an error if the system is not initialized        }
3076:S
3077:C        2 begin
3078:C        2   if not system_init then error (err_sys_int);
3079:C        2 end; { ck_sytem_init }
3080:S
3081:D        1 procedure ck_display_init;
3082:S
3083:D        2 { Purpose : To report an error if the display is not initialized        }
3084:S
3085:C        2 begin
3086:C        2   if not disp_init then error (err_dis_int);
3087:C        2 end; { ck_display_init }
3088:S
3089:D        1 procedure ck_locator_init;
3090:S
3091:D        2 { Purpose : To report an error if the locator is not initialized        }
3092:S
3093:C        2 begin
3094:C        2   if not loc_init then error (err_loc_int);
3095:C        2 end; { ck_locator_init }
3096:S
3097:D        1 function min(p1,p2:integer):integer;
3098:S
3099:D        2 { Purpose : To return the minimum of two integers                      }
3100:S
3101:C        2 begin
3102:C        2   if (p1 >= p2) then min := p2
3103:C        3     else min := p1;
3104:C        2 end; { min }
3105:S
3106:D        1 function max(p1,p2:integer):integer;
3107:S
3108:D        2 { Purpose : To return the maximum of two integers                      }
3109:S
3110:S
3111:C        2 begin
3112:C        2   if (p1 <= p2) then max := p2
3113:C        3     else max := p1;
3114:C        2 end; { max }
3115:S
3116:D        1 procedure order_values (    p1,p2 : real;
3117:D   -16  2                          var rp1, rp2 : real);
3118:S
3119:S          { Purpose : To return the passed in values ordered such that the smaller
3120:D   -16  2              value is in rp1.                                          }
3121:S
3122:C        2 begin
3123:C        2   if p1 <= p2 then
3124:C        3     begin
3125:C        3       rp1 := p1;
3126:C        3       rp2 := p2;
3127:C        3     end
3128:C        3   else
3129:C        3     begin
```



```
3130:C        3       rp2 := p1;
3131:C        3       rp1 := p2;
3132:C        3     end;
3133:C        2 end; { order_values }
3134:S
3135:S
3136:D        1 function opcode_ck(opcode,num_integer,num_real : integer) : integer;
3137:S
3138:D        2 { Purpose:  To ck for a valid opcode passed.  Opcodes are all defined such   }
3139:S        { }           that:
3140:S                       the hundreds digit = # of reals passed
3141:S                       the 1000's digit = # of integers passed
3142:S
3143:S                     This function returns values as follows:
3144:S
3145:S                       = 0 ; ok
3146:S                       = 2 ; wrong i size
3147:D        2              = 3 ; wrong r size                                       }
3148:S
3149:S
3150:D        2 var
3151:D   -10  2   digit : array [0..4] of gbyte;
3152:D   -12  2   i : gshortint;
3153:S
3154:C        2 begin
3155:C        2   for i := 0 to 4 do
3156:C        3     begin
3157:C        3       digit[i] := opcode - (opcode div 10) * 10;
3158:C        3       opcode := opcode div 10;
3159:C        3     end;
3160:S
3161:C        2   opcode_ck := 0;
3162:C        2   if digit[2] <> num_real then opcode_ck := 3;
3163:C        2   if digit[3] <> num_integer then opcode_ck := 2;
3164:C        2 end;
3165:S
3166:D   -24  1 procedure convert_hsl_to_rgb ( hue,sat,lite : real; var r,g,b : real);
3167:S
3168:D   -24  2 var
3169:D   -64  2   lx,ly,lz,frac,h : real;
3170:D   -68  2   j : integer;
3171:S
3172:C        2 begin
3173:C        2   h := 6*hue;
3174:C        2   j := trunc(h);
3175:C        2   frac := h-j;
3176:C        2   lx := lite*(1.0-sat);
3177:C        2   ly := lite*(1.0-sat*frac);
3178:C        2   lz := lite*(1.0-sat*(1-frac));
3179:S
3180:C        2   case j of
3181:C        3     0,6 : begin r := lite; g := lz;   b := lx;   end;
3182:C        3       1 : begin r := ly;   g := lite; b := lx;   end;
3183:C        3       2 : begin r := lx;   g := lite; b := lz;   end;
3184:C        3       3 : begin r := lx;   g := ly;   b := lite; end;
3185:C        3       4 : begin r := lz;   g := lx;   b := lite; end;
3186:C        3       5 : begin r := lite; g := lx;   b := ly;   end;
3187:C        3   end; { of case }
3188:C        2 end;
3189:S
```

```
3190:D   -24  1 procedure convert_rgb_to_hsl (r,g,b : real; var hue,sat,lite : real);
3191:S
3192:D   -32  2   var x : real;
3193:D   -56  2   tr,tg,tb : real;
3194:S
3195:D   -24  2   function max3(a,b,c : real) : real;
3196:S
3197:C        3   begin
3198:C        3     if (a >= b) and (a >= c) then max3 := a
3199:C        4     else
3200:C        4     if (b >= a) and (b >= c) then max3 := b
3201:C        5     else
3202:C        5     max3 := c;
3203:C        3   end;
3204:S
3205:D   -24  2   function min3(a,b,c : real) : real;
3206:S
3207:C        3   begin
3208:C        3     if (a <= b) and (a <= c) then min3 := a
3209:C        4     else
3210:C        4     if (b <= a) and (b <= c) then min3 := b
3211:C        5     else
3212:C        5     min3 := c;
3213:C        3   end;
3214:S
3215:C        2 begin
3216:C        2   hue := 0;                 { init values given, since values may be undefined }
3217:C        2   sat := 1;
3218:S
3219:C        2   lite := max3(r,g,b);
3220:C        2   x    := min3(r,g,b);
3221:C        2   if lite <> 0 then { calc hue and sat only if defined }
3222:C        3     begin
3223:C        3       sat  := (lite - x) / lite;
3224:C        3       if sat <> 0 then { calc hue only if defined, other wise defaults to old }
3225:C        4         begin
3226:S
3227:C        4           tr := (lite-r)/(lite-x);
3228:C        4           tg := (lite-g)/(lite-x);
3229:C        4           tb := (lite-b)/(lite-x);
3230:S
3231:C        4           if r=lite then
3232:C        5             begin
3233:C        5               if g=x then hue := 5+tb
3234:C        6                      else hue := 1-tg;
3235:C        5             end
3236:C        5           else
3237:C        5           if g=lite then
3238:C        6             begin
3239:C        6               if b=x then hue := 1+tr
3240:C        7                      else hue := 3-tb;
3241:C        6             end
3242:C        6           else
3243:C        6             begin
3244:C        6               if r=x then hue := 3+tg
3245:C        7                      else hue := 5-tr;
3246:C        6             end;
3247:C        4           hue := hue/6;
3248:C        4         end;
3249:C        3     end;
```



```
3250:C        2 end.
3251:S
3252:D        1 procedure adjust_return_echo ( var rx,ry : real );
3253:S
3254:D        2 { Purpose : ... st return echo values for the effects of rubber band snap }
3255:S
3256:C        2 begin
3257:C        2   case current_echo_type of
3258:C        3     5: ry := w_loc_echo_y;      { Horizontal rubber band}
3259:C        3     6: rx := w_loc_echo_x;      { Vertical rubber band   }
3260:C        3     7:                          { snap horz / vert rubber band }
3261:C        3        if abs(rx-w_loc_echo_x) >= abs(ry-w_loc_echo_y) then
3262:C        4          ry := w_loc_echo_y
3263:C        4        else rx := w_loc_echo_x;
3264:C        3     otherwise ;                 { no adjustment needed for others }
3265:C        3   end;  { of case }
3266:C        2 end; { adjust_return_echo }
3267:S
3268:D        1 { ************************************************************************** }
3269:D        1 {                                                                          }
3270:D        1 {              General Viewing transformation routines                     }
3271:D        1 {                                                                          }
3272:D        1 { ************************************************************************** }
3273:S
3274:D        1 procedure convert_intwtod(    x,y: gshortint;
3275:D        2                           var dx,dy: integer);
3276:S
3277:D        2 { Purpose : To convert form world to display cord   }
3278:S
3279:D        2 var
3280:D     -8 2   tx, ty : integer;
3281:D    -12 2   lx,ly : gshortint;
3282:S
3283:C        2 begin
3284:C        2   if short_defaults then
3285:C        3     begin
3286:C        3       dx := x;
3287:C        3       dy := y;
3288:C        3     end
3289:C        3   else
3290:C        3     with scalef do
3291:C        4       begin
3292:C        4         tx := x * x_display_delta;
3293:C        4         lx := tx div x_window_delta + x_display_offset;
3294:C        4         ty := y * y_display_delta;
3295:C        4         ly := ty div y_window_delta + y_display_offset;
3296:C        4         dx := lx;
3297:C        4         dy := ly;
3298:C        4       end;
3299:S
3300:C        2 end; { convert_intwtod }
3301:S
3302:D        1 procedure convert_wtod(    x,y:real;
3303:D    -16 2                           var dx,dy:integer);
3304:S
3305:D    -16 2 { Purpose : To convert form world to display cord   }
3306:S
3307:C        2 begin
3308:C        2   dx := trunc(x * xwtod_scale + xwtod_offset);
3309:C        2   dy := trunc(y * ywtod_scale + ywtod_offset);
```

```
3310:C        2   (WRITELN('WTOD (',X:8:5,Y:8:5,')=',DX:8,DY:8);)
3311:C        2 end; { convert_wtod }
3312:S
3313:D        1 procedure convert_ltod(    lx,ly : integer;
3314:D        2                         var dx,dy : integer );
3315:S
3316:D        2 { Purpose : To convert from locator to display units }
3317:S
3318:C        2 begin
3319:C        2   with gcb^ do
3320:C        3     begin
3321:C        3       dx := trunc( ((lx - log_loc_lim.xmin) * xltod_scale) +
3322:C        3             cur_disp_lim.xmin + 0.5);
3323:C        3       dy := trunc( ((ly - log_loc_lim.ymin) * yltod_scale) +
3324:C        3             cur_disp_lim.ymin + 0.5);
3325:C        3     end;
3326:C        2 end; { convert_ltod }
3327:S
3328:D        1 procedure convert_dtow(    dx,dy : integer;
3329:D        2                         var wx,wy : real);
3330:S
3331:D        2 { Purpose : To convert from display to world units }
3332:S
3333:C        2 begin
3334:S
3335:C        2   wx := (dx + xdtow_offset) * xdtow_scale;
3336:C        2   wy := (dy + ydtow_offset) * ydtow_scale;
3337:C        2 end; { convert_dtow }
3338:S
3339:S
3340:D        1 procedure locator_limits (lxmin,lxmax,
3341:D   -32   2                           lymin,lymax : real);
3342:S
3343:D   -32   2 { Purpose : To set the locator limits ( in device units )              }
3344:D   -32   2 { Note: This routine does not perform error cking    )
3345:S
3346:C        2 begin
3347:C        2   with gcb^ do
3348:C        3     begin
3349:C        3       { set the logical limits }
3350:C        3       with log_loc_lim do
3351:C        4         begin
3352:C        4           xmin := lxmin;
3353:C        4           xmax := lxmax;
3354:C        4           ymin := lymin;
3355:C        4           ymax := lymax;
3356:C        4         end;
3357:S
3358:C        3       { calculate the display / locator xforms }
3359:S
3360:C        3       xltod_scale := (cur_disp_lim.xmax - cur_disp_lim.xmin) /
3361:C        3                      (log_loc_lim.xmax - log_loc_lim.xmin);
3362:S
3363:C        3       yltod_scale := (cur_disp_lim.ymax - cur_disp_lim.ymin) /
3364:C        3                      (log_loc_lim.ymax - log_loc_lim.ymin);
3365:S
3366:S
3367:C        3       { set the locator echo position to the center of the window }
3368:S
3369:C        3       with window_lim do
```



```
3370:C        4         begin
3371:C        4           w_loc_echo_x := ((xmax - xmin) / 2.0) + xmin;
3372:C        4           w_loc_echo_y := ((ymax - ymin) / 2.0) + ymin;
3373:C        4         end;
3374:S
3375:C        3       { convert to device units }
3376:S
3377:C        3       convert_wtod ( w_loc_echo_x, w_loc_echo_y,
3378:C        3                      d_loc_echo_x, d_loc_echo_y );
3379:S
3380:C        3     end;
3381:S
3382:C        2 end; { locator_limits }
3383:S
3384:S
3385:D        1 procedure calculate_viewing;
3386:S
3387:D        2 { Purpose : To calculate a new viewing transformation                  }
3388:S
3389:D        2 var
3390:D    -8  2   temp : real;           { temp work var }
3391:S
3392:D        2   procedure int_test;
3393:S
3394:S          { Purpose : To see if conditions are right, and setup if so, for int_move
3395:D        3             and int_line to call fast integer device routines          }
3396:S
3397:D        3   var
3398:D    -4  3     sxmin : integer;
3399:D    -8  3     sxmax : integer;
3400:D   -12  3     symin : integer;
3401:D   -16  3     symax : integer;
3402:D   -18  3     gsxmin : gshortint;
3403:D   -20  3     gsxmax : gshortint;
3404:D   -22  3     gsymin : gshortint;
3405:D   -24  3     gsymax : gshortint;
3406:S
3407:D   -28  3     window_delta_x : integer;
3408:D   -32  3     window_delta_y : integer;
3409:S
3410:C        3   begin
3411:C        3     short_flag := false;
3412:S
3413:C        3       with gcb^ do
3414:C        4         begin
3415:C        4           if disp_init then
3416:C        5             try
3417:C        6  $range on$          { range cking done here }
3418:S
3419:C        6               { find device points for the corners of the window }
3420:C        6               convert_wtod(window_lim.xmin,window_lim.ymin,sxmin,symin);
3421:C        6               convert_wtod(window_lim.xmax,window_lim.ymax,sxmax,symax);
3422:S
3423:C        6               { convert to non-fix point form }
3424:C        6               gsxmin := sxmin;
3425:C        6               gsxmax := sxmax;
3426:C        6               gsymin := symin;
3427:C        6               gsymax := symax;
3428:S
3429:C        6               {set up scale info
```

```
3430:C        6                with window_lim do
3431:C        7                  begin
3432:C        7                    window_delta_x := trunc(xmax) - trunc(xmin);
3433:C        7                    window_delta_y := trunc(ymax) - trunc(ymin);
3434:S
3435:C        7                    { ck delta window values in range }
3436:C        7                    if (abs(window_delta_x) > 32767) or
3437:C        8                       (abs(window_delta_y) > 32767) then
3438:C        8                       escape(-4); { force integer overflow }
3439:S
3440:C        7                    with scalef do
3441:C        8                      begin
3442:C        8                        x_display_delta := gsxmax - gsxmin;
3443:C        8                        x_window_delta := window_delta_x;
3444:C        8                        x_display_offset :=
3445:C        8                          ((-trunc(xmin) * x_display_delta) div x_window_delta) + gsxmin;
3446:C        8                        y_display_delta := gsymax - gsymin;
3447:C        8                        y_window_delta := window_delta_y;
3448:C        8                        y_display_offset :=
3449:C        8                          ((-trunc(ymin) * y_display_delta) div y_window_delta) + gsymin;
3450:S
3451:S
3452:C        8                        if (x_display_offset = 0) and
3453:C        9                          (x_window_delta = x_display_delta) and
3454:C        9                          (y_display_offset = 0) and
3455:C        9                          (y_window_delta = y_display_delta) then
3456:C        9                        short_defaults := true
3457:C        9                      else
3458:C        9                        short_defaults := false;
3459:S
3460:C        8                    end;
3461:C        7                end;
3462:C        6              short_flag := true;
3463:S
3464:C        6            recover { can't use int_move / int_line internal routines }
3465:C        6              if (escapecode > -4) or { igonor all math and range errors }
3466:C        7                 (escapecode < -8) then escape(escapecode);
3467:C        4 $range off$
3468:C        4          end;
3469:C        3   end;      { of procedure int_test }
3470:S
3471:S
3472:C        2 begin
3473:C        2   with gcb^ do
3474:C        3     begin
3475:C        3       { set view surface limits to the logical display limits              }
3476:C        3       cur_disp_lim := log_disp_lim;
3477:S
3478:S                 { if desired aspect = aspect of logical limits then we don't need to
3479:C        3         redefine the limits as set above                                   }
3480:C        3       if (aspect_ratio <> log_aspect) then
3481:C        4         begin
3482:S
3483:C        4           {determine which device limits to change}
3484:C        4           if (aspect_ratio <= log_aspect) then
3485:C        5             begin
3486:S
3487:C        5               {check display justification}
3488:C        5               if (disp_just = centered) then
3489:S
```



```
3490:S                           {set vertical limits for device on which view space is
3491:C        6                   centered within the logical display limits           }
3492:C        6                 begin
3493:C        6                   temp := 0.5 * (dyunits - (dxunits * aspect_ratio) );
3494:C        6                   cur_disp_lim.ymin := log_disp_lim.ymin + temp;
3495:C        6                   cur_disp_lim.ymax := log_disp_lim.ymax - temp;
3496:C        6                 end
3497:S
3498:C        6               else  { not centered }
3499:S
3500:S                           {set vertical limits for device on which view space is
3501:C        6                   lower left justified within the logical display limits   }
3502:C        6                 begin
3503:C        6                   cur_disp_lim.ymax :=
3504:C        6                     log_disp_lim.ymin + (dxunits * aspect_ratio);
3505:C        6                 end;
3506:C        5             end
3507:S
3508:C        5           else  { current aspect ratio > aspect ratio of logical limits   }
3509:C        5             begin
3510:S
3511:C        5               {check display justification}
3512:C        5               if (disp_just = centered) then
3513:S
3514:S                           {set horizontal limits for device on which view space is
3515:C        6                   centered within the logical display limits             }
3516:C        6                 begin
3517:C        6                   temp := 0.5 * (dxunits - (dyunits / aspect_ratio) );
3518:C        6                   cur_disp_lim.xmin := log_disp_lim.xmin + temp;
3519:C        6                   cur_disp_lim.xmax := log_disp_lim.xmax - temp;
3520:C        6                 end
3521:S
3522:C        6               else  { not centered }
3523:S
3524:S                           {set horizontal limits for device on which view space is
3525:C        6                   lower left justified within the logical display limits    }
3526:C        6                 begin
3527:C        6                   cur_disp_lim.xmax :=
3528:C        6                     log_disp_lim.xmin + (dyunits / aspect_ratio);
3529:C        6                 end;
3530:C        5             end;
3531:C        4         end;
3532:S
3533:S
3534:C        3       { set clipping limits here }
3535:C        3       if disp_init then
3536:C        4         with gle_gcb^,cur_disp_lim do
3537:C        5           begin
3538:S                     {WRITELN('C =',XMIN:19:18,XMAX:19:18,YMIN:19:18,YMAX:19:18);
3539:S                     WRITELN('CI=',TRUNC(XMIN+0.5):19,TRUNC(XMAX+0.5):19,
3540:C        5            TRUNC(YMIN+0.5):19,TRUNC(YMAX+0.5):19);}
3541:C        5            info1 := trunc(xmin+0.5);
3542:C        5            info2 := trunc(xmax+0.5);
3543:C        5            info3 := trunc(ymin+0.5);
3544:C        5            info4 := trunc(ymax+0.5);
3545:C        5            gle_clip_limits (gle_gcb);
3546:C        5           end;
3547:S
3548:C        3       { recalculate the world to display transformation constants          }
3549:S
```

```
3550:C        3       { x scale and offset                                                }
3551:C        3       with cur_disp_lim do
3552:C        4         temp := (xmax - xmin ) / cur_vir_lim.xlim;
3553:S
3554:C        3       xwtod_scale := temp * ((viewport_lim.xmax - viewport_lim.xmin) /
3555:C        3                              (window_lim.xmax - window_lim.xmin));
3556:S
3557:C        3       xwtod_offset := cur_disp_lim.xmin + (viewport_lim.xmin * temp) -
3558:C        3                              (window_lim.xmin * xwtod_scale);
3559:S
3560:C        3       xdtow_scale := 1.0 / xwtod_scale;
3561:S
3562:C        3       xdtow_offset := - xwtod_offset;
3563:C        3       xwtod_offset := xwtod_offset + 0.5 (0.00000001);
3564:S
3565:C        3       { y scale and offset                                                }
3566:C        3       with cur_disp_lim do
3567:C        4         temp := (ymax - ymin ) / cur_vir_lim.ylim;
3568:S
3569:C        3       ywtod_scale := temp * ((viewport_lim.ymax - viewport_lim.ymin) /
3570:C        3                              (window_lim.ymax - window_lim.ymin));
3571:S
3572:S
3573:C        3       ywtod_offset := cur_disp_lim.ymin + (viewport_lim.ymin * temp) -
3574:C        3                              (window_lim.ymin * ywtod_scale);
3575:S
3576:C        3       ydtow_scale := 1.0 / ywtod_scale;
3577:S
3578:C        3       ydtow_offset := - ywtod_offset;
3579:C        3       ywtod_offset := ywtod_offset + 0.5 (0.00000001);
3580:S
3581:C        3       if disp_eq_loc then
3582:C        4         begin
3583:S                     {if display and locator are the same physical device then set
3584:C        4            the locator limits to the display limits                       }
3585:S
3586:C        4            with cur_disp_lim do
3587:C        5              locator_limits(xmin,xmax,ymin,ymax);
3588:C        4         end
3589:C        4       else
3590:C        4         begin
3591:C        4           {set the locator limits to current locator limits             }
3592:S
3593:C        4           with log_loc_lim do
3594:C        5             locator_limits(xmin,xmax,ymin,ymax);
3595:C        4         end;
3596:S
3597:C        3       { set up for int_move / int_line scale }
3598:C        3       int_test;
3599:C        3     end;
3600:C        2 end; { calculate_viewing }
3601:S
3602:S
3603:D    -32  1 procedure display_limits ( dxmin, dxmax, dymin, dymax : real );
3604:S
3605:D    -32  2 { Purpose : To set the logical display limits in device units }
3606:S
3607:C        2 begin
3608:S            {WRITELN('NEW DISPLAY LIM = ',DXMIN:19:18,DXMAX:19:18,DYMIN:19:18,DYMAX:19:18);
3609:C        2   }
```



```
3610:C        2   with gcb^ do
3611:C        3     begin
3612:C        3       { set new log limits }
3613:C        3       with log_disp_lim do
3614:C        4         begin
3615:C        4           xmin := dxmin;
3616:C        4           xmax := dxmax;
3617:C        4           ymin := dymin;
3618:C        4           ymax := dymax;
3619:C        4         end;
3620:S
3621:C        3       { set the display width and height }
3622:S
3623:C        3       dxunits := dxmax - dxmin;
3624:C        3       dyunits := dymax - dymin;
3625:S
3626:C        3       { calculate aspect ratio of new log lim }
3627:S
3628:C        3       log_aspect := dyunits / dxunits;
3629:S
3630:C        3       { set up new viewing transformation }
3631:C        3       calculate_viewing;
3632:S
3633:C        3     end;
3634:S
3635:C        2 end; { display_limits }
3636:S
3637:C        1 end.  { of module DGL_GEN }

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

GLE_FILE

Description

GLE_FILE contains file drivers which interface the graphics drivers to the file system.

Usage

Used with display spooling for HPGL device models. Spooling for non-HPGL devices is not supported in DGL.

Requirements

GLE_TYPES and GLE_UTLS.

Usage

Used in display spooling for HPGL device models. Spooling for non-HPGL devices is not supported in DGL.

```
    1:D        0 {                                                                    }
    2:D        0 { Graphics Low End                                                   }
    3:D        0 {                                                                    }
    4:D        0 { Module    = GLE_FILE                                               }
    5:D        0 { Programer = BJS                                                    }
    6:D        0 { Date      = 10-10-82                                               }
    7:D        0 {                                                                    }
    8:D        0 { Purpose: To provide general file drivers for ascii device handlers.}
    9:S
   10:D        0 { Rev history                                                        }
   11:D        0 {  Created  - 10-10-82                                               }
   12:D        0 {  Modified - XX-XX-XX                                               }
   13:S
   14:S          {     (c) Copyright Hewlett-Packard Company, 1983.
   15:S                All rights are reserved.  Copying or other
   16:S                reproduction of this program except for archival
   17:S                purposes is prohibited without the prior
   18:S                written consent of Hewlett-Packard Company.
   19:S
   20:S
   21:S                            RESTRICTED RIGHTS LEGEND
   22:S
   23:S                Use, duplication, or disclosure by the Government
   24:S                is subject to restrictions as set forth in
   25:S                paragraph (b) (3) (B) of the Rights in Technical
   26:S                Data and Computer Software clause in
   27:S                DAR 7-104.9(a).
   28:S
   29:S                HEWLETT-PACKARD COMPANY
   30:D        0       Fort Collins, Colorado                                 }
   31:S
   32:D        0 $SEARCH 'GLE_TYPES','GLE_UTLS'$
   33:D        0 $modcal$
   33:D        0 $include 'OPTIONS'$  { ***************** COMPILER OPTIONS ***************** }
   34:S
   35:S           { This include file specifies range checking, debug and other compiler
   36:D        0    options for the graphics library                                  }
   37:S
   38:D        0 $debug OFF$
   39:D        0 $range OFF$
   40:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
   41:D        0 $FLOAT_HDW TEST$
   42:S
   43:S
   44:S
   45:S
   46:D        0 $include 'OPTIONS'$  { ***************** COMPILER OPTIONS ***************** }
 9000:D        0 $LINENUM 9000$
 9001:S
 9002:D        0 module gle_file_io;
 9003:S
 9004:D        1 import gle_types;
 9005:S
 9006:D        1 export
 9007:S
 9008:D        1 type
 9009:D        1   file_iocb_ptr = ^file_iocb;
 9010:D        1   file_iocb = record
 9011:D        1                 file_name : anyptr;
 9012:D        1                 name_size : gle_shortint;
```



```
 9013:D        1                 output_file : text;
 9014:D        1                 lock_on_close : gle_shortint; { 1 = lock }
 9015:D        1               end;
 9016:S
 9017:D        1 procedure file_init  (  anyvar iocb_ptr : anyptr );
 9018:D        1 procedure file_write (  anyvar iocb_ptr, data_ptr : anyptr );
 9019:D        1 procedure file_inq_timeout (  anyvar iocb_ptr: anyptr; var value : integer );
 9020:D        1 procedure file_set_timeout (  anyvar iocb_ptr: anyptr; value : integer );
 9021:D        1 procedure file_term  (  anyvar iocb_ptr : anyptr );
 9022:S
 9023:D        1 implement
 9024:S
 9025:D        1 import gle_utls;
 9026:S
 9027:D        1 { The following types must match the types declared in GLE_HPGL, GLE_HPGLI }
 9028:D        1 type
 9029:D        1   ascii_buffer_ptr = ^ascii_buffer;
 9030:S
 9031:D        1   ascii_buffer = packed record
 9032:D        1                    maximum : integer;
 9033:D        1                    current : integer;
 9034:D        1                    data    : packed array [1..32767] of char;
 9035:D        1                  end;
 9036:S
 9037:D        1 procedure file_inq_timeout (  anyvar iocb_ptr : anyptr; var value : integer );
 9038:S
 9039:D        2 { Dummy procedures for file io }
 9040:C        2 begin
 9041:C        2 end;
 9042:S
 9043:D        1 procedure file_set_timeout (  anyvar iocb_ptr : anyptr; value : integer );
 9044:S
 9045:D        2 { Dummy procedures for file io }
 9046:C        2 begin
 9047:C        2 end;
 9048:S
 9049:D        1 procedure file_init  (  anyvar iocb_ptr : anyptr );
 9050:S
 9051:D        2 var
 9052:D  -256  2   name : string[255];
 9053:S
 9054:C        2 begin
 9055:C        2   with file_iocb_ptr(iocb_ptr)^ do
 9056:C        3     begin
 9057:C        3       gle_copy_to_string(file_name,name_size,name);
 9058:C        3       rewrite(output_file,name);
 9059:C        3     end;
 9060:C        2 end;
 9061:S
 9062:D        1 procedure file_write (  anyvar iocb_ptr, data_ptr : anyptr );
 9063:S
 9064:C        2 begin
 9065:C        2   with file_iocb_ptr(iocb_ptr)^,ascii_buffer_ptr(data_ptr)^  do
 9066:C        3     begin
 9067:C        3       writeln(output_file,data:current);
 9068:C        3       current := 0;
 9069:C        3     end;
 9070:C        2 end;
 9071:S
 9072:D        1 procedure file_term  (  anyvar iocb_ptr : anyptr );
```

```
  9073:S
  9074:C        2 begin
  9075:C        2   with file_iocb_ptr(iocb_ptr)^ do
  9076:C        3     begin
  9077:C        3       if lock_on_close = 1 then close (output_file,'lock')
  9078:C        4       else                      close (output_file);
  9079:C        3     end;
  9080:C        2 end;
  9081:S
  9082:C        1 end. { of module gle_file_io }
  9083:S

No errors. No warnings.
             ***** Nonstandard language features enabled *****
```

GLE_GEN

Description

GLE_GEN provides an interface between the GLE caller and the output procedure variables in the GCB. It also contains a procedure used to initialize the GCB.

Requirements

GLE_TYPES.

```
    1:D        0 {                                                                    }
    2:D        0 { Graphics Low End                                                   }
    3:D        0 {                                                                    }
    4:D        0 { Module    = GLE_GEN                                                }
    5:D        0 { Programer = BJS                                                    }
    6:D        0 { Date      = 10- 5-82                                               }
    7:D        0 {                                                                    }
    8:D        0 { Purpose: To provide an interface between the GLE caller and the procedure }
    9:D        0 {          variables in the GCB.  The module also initializes the GCB.  }
   10:S
   11:D        0 { Rev history                                                        }
   12:D        0 {  Created  - 10- 5-82                                               }
   13:D        0 {  Modified - XX-XX-XX                                               }
   14:S
   15:S          {     (c) Copyright Hewlett-Packard Company, 1983.
   16:S                All rights are reserved.  Copying or other
   17:S                reproduction of this program except for archival
   18:S                purposes is prohibited without the prior
   19:S                written consent of Hewlett-Packard Company.
   20:S
   21:S
   22:S                           RESTRICTED RIGHTS LEGEND
   23:S
   24:S                Use, duplication, or disclosure by the Government
   25:S                is subject to restrictions as set forth in
   26:S                paragraph (b) (3) (B) of the Rights in Technical
   27:S                Data and Computer Software clause in
   28:S                DAR 7-104.9(a).
   29:S
   30:S                HEWLETT-PACKARD COMPANY
   31:D        0       Fort Collins, Colorado                                 }
   32:S
   33:D        0 $modcal$
   34:D        0 $search 'GLE_TYPES'$
   34:D        0 $include 'OPTIONS'$  { ****************** compiler options  ****************** }
   35:S
   36:S           { This include file specifies range checking, debug and other compiler
   37:D        0    options for the graphics library                                  }
   38:S
   39:D        0 $debug OFF$
   40:D        0 $range OFF$
   41:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
   42:D        0 $FLOAT_HDW TEST$
   43:S
   44:S
   45:S
   46:S
   47:D        0 $include 'OPTIONS'$  { ****************** compiler options  ****************** }
 2000:D        0 $linenum 2000$
 2001:S
 2002:D        0 module gle_gen;
 2003:S
 2004:D        1 import gle_types;
 2005:S
 2006:D        1 export
 2007:S
 2008:D        1 procedure gle_move                ( gcb : graphics_control_block_ptr );
 2009:D        1 procedure gle_draw                ( gcb : graphics_control_block_ptr );
 2010:D        1 procedure gle_text                ( gcb : graphics_control_block_ptr );
 2011:D        1 procedure gle_char_size           ( gcb : graphics_control_block_ptr );
```



```
 2012:D        1 procedure gle_text_spacing        ( gcb : graphics_control_block_ptr );
 2013:D        1 procedure gle_text_dir            ( gcb : graphics_control_block_ptr );
 2014:D        1 procedure gle_text_just           ( gcb : graphics_control_block_ptr );
 2015:D        1 procedure gle_marker              ( gcb : graphics_control_block_ptr );
 2016:D        1 procedure gle_index_color         ( gcb : graphics_control_block_ptr );
 2017:D        1 procedure gle_clear               ( gcb : graphics_control_block_ptr );
 2018:D        1 procedure gle_clip_limits         ( gcb : graphics_control_block_ptr );
 2019:D        1 procedure gle_init_gcb            ( gcb : graphics_control_block_ptr );
 2020:D        1 procedure gle_get_plp2            ( gcb : graphics_control_block_ptr );
 2021:D        1 procedure gle_get_polygon_info    ( gcb : graphics_control_block_ptr );
 2022:D        1 procedure gle_set_marker          ( gcb : graphics_control_block_ptr );
 2023:D        1 procedure gle_marker_size         ( gcb : graphics_control_block_ptr );
 2024:D        1 procedure gle_flush_buffer        ( gcb : graphics_control_block_ptr );
 2025:D        1 procedure gle_term                ( gcb : graphics_control_block_ptr );
 2026:D        1 procedure gle_cursor              ( gcb : graphics_control_block_ptr );
 2027:D        1 procedure gle_polygon             ( gcb : graphics_control_block_ptr );
 2028:D        1 procedure gle_fill_index_color    ( gcb : graphics_control_block_ptr );
 2029:D        1 procedure gle_define_color_map    ( gcb : graphics_control_block_ptr );
 2030:D        1 procedure gle_define_drawing_mode ( gcb : graphics_control_block_ptr );
 2031:D        1 procedure gle_linestyle           ( gcb : graphics_control_block_ptr );
 2032:D        1 procedure gle_linewidth           ( gcb : graphics_control_block_ptr );
 2033:D        1 procedure gle_buffer_mode         ( gcb : graphics_control_block_ptr );
 2034:D        1 procedure gle_graphics_on_off     ( gcb : graphics_control_block_ptr );
 2035:D        1 procedure gle_gload               ( gcb : graphics_control_block_ptr );
 2036:D        1 procedure gle_gstore              ( gcb : graphics_control_block_ptr );
 2037:D        1 procedure gle_get_raster          ( gcb : graphics_control_block_ptr );
 2038:D        1 procedure gle_get_color_map       ( gcb : graphics_control_block_ptr );
 2039:D        1 procedure gle_output_escapei      ( gcb : graphics_control_block_ptr );
 2040:D        1 procedure gle_output_escapeo      ( gcb : graphics_control_block_ptr );
 2041:D        1 procedure gle_await_blanking      ( gcb : graphics_control_block_ptr );
 2042:S
 2043:D        1 implement
 2044:S
 2045:D        1 procedure gle_output_escapeo ( gcb : graphics_control_block_ptr );
 2046:S
 2047:C        2 begin
 2048:C        2   call (gcb^.output_escapeo,gcb);
 2049:C        2 end;
 2050:S
 2051:D        1 procedure gle_output_escapei ( gcb : graphics_control_block_ptr );
 2052:S
 2053:C        2 begin
 2054:C        2   call (gcb^.output_escapei,gcb);
 2055:C        2 end;
 2056:S
 2057:D        1 procedure gle_move ( gcb : graphics_control_block_ptr );
 2058:S
 2059:C        2 begin
 2060:C        2   call (gcb^.move,gcb);
 2061:C        2 end;
 2062:S
 2063:D        1 procedure gle_draw ( gcb : graphics_control_block_ptr );
 2064:S
 2065:C        2 begin
 2066:C        2   call (gcb^.draw,gcb);
 2067:C        2 end;
 2068:S
 2069:D        1 procedure gle_text ( gcb : graphics_control_block_ptr );
 2070:S
 2071:C        2 begin
```

```
2072:C        2   call (gcb^.text,gcb);
2073:C        2 end;
2074:S
2075:D        1 procedure gle_char_size ( gcb : graphics_control_block_ptr );
2076:S
2077:C        2 begin
2078:C        2   call (gcb^.char_size,gcb);
2079:C        2 end;
2080:S
2081:D        1 procedure gle_text_spacing ( gcb : graphics_control_block_ptr );
2082:S
2083:C        2 begin
2084:C        2   call (gcb^.text_spacing,gcb);
2085:C        2 end;
2086:S
2087:D        1 procedure gle_text_dir ( gcb : graphics_control_block_ptr );
2088:S
2089:C        2 begin
2090:C        2   call (gcb^.text_dir,gcb);
2091:C        2 end;
2092:S
2093:D        1 procedure gle_clear ( gcb : graphics_control_block_ptr );
2094:S
2095:C        2 begin
2096:C        2   call (gcb^.clear,gcb);
2097:C        2 end;
2098:S
2099:D        1 procedure gle_text_just ( gcb : graphics_control_block_ptr );
2100:S
2101:C        2 begin
2102:C        2   call (gcb^.text_dir,gcb);
2103:C        2 end;
2104:S
2105:D        1 procedure gle_marker ( gcb : graphics_control_block_ptr );
2106:S
2107:C        2 begin
2108:C        2   call (gcb^.marker,gcb);
2109:C        2 end;
2110:S
2111:D        1 procedure gle_index_color ( gcb : graphics_control_block_ptr );
2112:S
2113:C        2 begin
2114:C        2   call (gcb^.index_color,gcb);
2115:C        2 end;
2116:S
2117:D        1 procedure gle_clip_limits ( gcb : graphics_control_block_ptr );
2118:S
2119:C        2 begin
2120:C        2   call (gcb^.clip_limits,gcb);
2121:C        2 end;
2122:S
2123:D        1 procedure gle_init_gcb ( gcb : graphics_control_block_ptr );
2124:S
2125:C        2 begin
2126:C        2   with gcb^ do
2127:C        3     begin
2128:C        3       display_name_char_count := 0;
2129:C        3       display_handler_char_count := 0;
2130:C        3       iocb := nil;
2131:C        3       device_buf := nil;
```



```
2132:C        3       dev_dep_stuff := nil;
2133:C        3       device_info := nil;
2134:C        3       error_return := 0;
2135:C        3       spooling := 0;
2136:C        3     end;
2137:C        2 end;
2138:S
2139:D        1 procedure gle_get_polygon_info ( gcb : graphics_control_block_ptr );
2140:S
2141:C        2 begin
2142:C        2   call(gcb^.get_polygon_info,gcb);
2143:C        2 end;
2144:S
2145:D        1 procedure gle_get_p1p2 ( gcb : graphics_control_block_ptr );
2146:S
2147:C        2 begin
2148:C        2   call(gcb^.inq_p1p2,gcb);
2149:C        2 end;
2150:S
2151:D        1 procedure gle_set_marker ( gcb : graphics_control_block_ptr );
2152:S
2153:C        2 begin
2154:C        2   call(gcb^.set_marker,gcb);
2155:C        2 end;
2156:S
2157:D        1 procedure gle_marker_size ( gcb : graphics_control_block_ptr );
2158:S
2159:C        2 begin
2160:C        2   call(gcb^.marker_size,gcb);
2161:C        2 end;
2162:S
2163:D        1 procedure gle_flush_buffer( gcb : graphics_control_block_ptr );
2164:S
2165:C        2 begin
2166:C        2   with gcb^ do
2167:C        3     call(gcb^.flush_buffer,gcb);
2168:C        2 end;
2169:S
2170:D        1 procedure gle_term ( gcb : graphics_control_block_ptr );
2171:S
2172:C        2 begin
2173:C        2     with gcb^ do
2174:C        3       call(io_term, iocb);
2175:C        2 end;
2176:S
2177:D        1 procedure gle_cursor ( gcb : graphics_control_block_ptr );
2178:S
2179:C        2 begin
2180:C        2   call (gcb^.cursor,gcb);
2181:C        2 end;
2182:S
2183:D        1 procedure gle_polygon ( gcb : graphics_control_block_ptr );
2184:S
2185:C        2 begin
2186:C        2   call (gcb^.polygon,gcb);
2187:C        2 end;
2188:S
2189:D        1 procedure gle_fill_index_color ( gcb : graphics_control_block_ptr );
2190:S
2191:C        2 begin
```

```
2192:C        2   call (gcb^.fill_index_color,gcb);
2193:C        2 end;
2194:S
2195:D        1 procedure gle_define_color_map( gcb : graphics_control_block_ptr );
2196:S
2197:C        2 begin
2198:C        2   call (gcb^.define_color_map,gcb);
2199:C        2 end;
2200:S
2201:D        1 procedure gle_define_drawing_mode ( gcb : graphics_control_block_ptr );
2202:S
2203:C        2 begin
2204:C        2   call (gcb^.define_drawing_mode,gcb);
2205:C        2 end;
2206:S
2207:D        1 procedure gle_linestyle ( gcb : graphics_control_block_ptr );
2208:S
2209:C        2 begin
2210:C        2   call (gcb^.linestyle,gcb);
2211:C        2 end;
2212:S
2213:D        1 procedure gle_linewidth ( gcb : graphics_control_block_ptr );
2214:S
2215:C        2 begin
2216:C        2   call (gcb^.linewidth,gcb);
2217:C        2 end;
2218:S
2219:D        1 procedure gle_buffer_mode ( gcb : graphics_control_block_ptr );
2220:S
2221:C        2 begin
2222:C        2   call (gcb^.buffer_mode,gcb);
2223:C        2 end;
2224:S
2225:D        1 procedure gle_graphics_on_off ( gcb : graphics_control_block_ptr );
2226:S
2227:C        2 begin
2228:C        2   call (gcb^.graphics_on_off,gcb);
2229:C        2 end;
2230:S
2231:D        1 procedure gle_gload ( gcb : graphics_control_block_ptr );
2232:S
2233:C        2 begin
2234:C        2   call (gcb^.gload,gcb);
2235:C        2 end;
2236:S
2237:D        1 procedure gle_gstore ( gcb : graphics_control_block_ptr );
2238:S
2239:C        2 begin
2240:C        2   call (gcb^.gstore,gcb);
2241:C        2 end;
2242:S
2243:D        1 procedure gle_get_raster ( gcb : graphics_control_block_ptr );
2244:S
2245:C        2 begin
2246:C        2   call (gcb^.get_raster,gcb);
2247:C        2 end;
2248:S
2249:D        1 procedure gle_get_color_map ( gcb : graphics_control_block_ptr );
2250:S
2251:C        2 begin
```



```
2252:C        2   call (gcb^.get_color_map,gcb);
2253:C        2 end;
2254:S
2255:D        1 procedure gle_await_blanking ( gcb : graphics_control_block_ptr );
2256:S
2257:C        2 begin
2258:C        2   call (gcb^.await_blanking,gcb);
2259:C        2 end;
2260:S
2261:S
2262:C        1 end. { of module gle_gen }

No errors. No warnings.
          ***** Nonstandard language features enabled *****
```

Description

GLE_GENI provides an interface between the GLE caller and the input procedure variables in the GCB. It also contains a procedure to initialize the GCB.

Requirements

GLE_TYPES.

```
     1:D        0 {                                                                     }
     2:D        0 { Graphics Low End                                                    }
     3:D        0 {                                                                     }
     4:D        0 { Module    = GLE_GENI                                                }
     5:D        0 { Programer = BJS                                                     }
     6:D        0 { Date      = 10-10-82                                                }
     7:D        0 {                                                                     }
     8:D        0 { Purpose: To provide interface procedure to the procedure variables in }
     9:D        0 {          the GCB and to provide GCB initialization.                 }
    10:S
    11:D        0 { Rev history                                                         }
    12:D        0 {  Created  - 10-10-82                                                }
    13:D        0 {  Modified - XX-XX-XX                                                }
    14:S
    15:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    16:S                All rights are reserved.  Copying or other
    17:S                reproduction of this program except for archival
    18:S                purposes is prohibited without the prior
    19:S                written consent of Hewlett-Packard Company.
    20:S
    21:S
    22:S                            RESTRICTED RIGHTS LEGEND
    23:S
    24:S                Use, duplication, or disclosure by the Government
    25:S                is subject to restrictions as set forth in
    26:S                paragraph (b) (3) (B) of the Rights in Technical
    27:S                Data and Computer Software clause in
    28:S                DAR 7-104.9(a).
    29:S
    30:S                HEWLETT-PACKARD COMPANY
    31:D        0       Fort Collins, Colorado                                  }
    32:S
    33:D        0 $search 'GLE_TYPES'$
    34:D        0 $modcal$
    34:D        0 $include 'OPTIONS'$  { ****************** COMPILER OPTIONS ****************** }
    35:S
    36:S           { This include file specifies range checking, debug and other compiler
    37:D        0    options for the graphics library                                   }
    38:S
    39:D        0 $debug OFF$
    40:D        0 $range OFF$
    41:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    42:D        0 $FLOAT_HDW TEST$
    43:S
    44:S
    45:S
    46:S
    47:D        0 $include 'OPTIONS'$  { ****************** COMPILER OPTIONS ****************** }
  3000:D        0 $linenum 3000$
  3001:S
  3002:D        0 module gle_geni;
  3003:S
  3004:D        1 import gle_types;
  3005:S
  3006:D        1 export
  3007:S
  3008:D        1 procedure gle_init_input_gcb     ( gcbi : graphics_input_control_block_ptr );
  3009:D        1 procedure gle_get_input_p1p2     ( gcbi : graphics_input_control_block_ptr );
  3010:D        1 procedure gle_input_term         ( gcbi : graphics_input_control_block_ptr );
  3011:D        1 procedure gle_input_echo         ( gcbi : graphics_input_control_block_ptr );
```



```
  3012:D        1 procedure gle_input_escapei      ( gcbi : graphics_input_control_block_ptr );
  3013:D        1 procedure gle_input_escapeo      ( gcbi : graphics_input_control_block_ptr );
  3014:D        1 procedure gle_sample             ( gcbi : graphics_input_control_block_ptr );
  3015:D        1 procedure gle_start_digitize     ( gcbi : graphics_input_control_block_ptr );
  3016:D        1 procedure gle_get_digitize       ( gcbi : graphics_input_control_block_ptr );
  3017:S
  3018:D        1 implement
  3019:S
  3020:D        1 procedure gle_init_input_gcb (  gcbi : graphics_input_control_block_ptr );
  3021:S
  3022:C        2 begin
  3023:C        2   with gcbi^ do
  3024:C        3     begin
  3025:C        3       input_name_char_count := 0;
  3026:C        3       input_handler_char_count := 0;
  3027:S
  3028:C        3       iocb := nil;
  3029:C        3       device_buf := nil;
  3030:C        3       dev_dep_stuff := nil;
  3031:C        3       device_info := nil;
  3032:C        3       error_return := 0;
  3033:C        3     end;
  3034:C        2 end;
  3035:S
  3036:D        1 procedure gle_get_input_p1p2 ( gcbi : graphics_input_control_block_ptr );
  3037:S
  3038:C        2 begin
  3039:C        2   call(gcbi^.inq_p1p2,gcbi);
  3040:C        2 end;
  3041:S
  3042:D        1 procedure gle_input_term ( gcbi : graphics_input_control_block_ptr );
  3043:S
  3044:C        2 begin
  3045:C        2     with gcbi^ do
  3046:C        3       call(io_term, iocb);
  3047:C        2 end;
  3048:S
  3049:D        1 procedure gle_input_echo (  gcbi : graphics_input_control_block_ptr );
  3050:S
  3051:C        2 begin
  3052:C        2   call (gcbi^.input_echo,gcbi);
  3053:C        2 end;
  3054:S
  3055:D        1 procedure gle_input_escapei (  gcbi : graphics_input_control_block_ptr );
  3056:S
  3057:C        2 begin
  3058:C        2   call (gcbi^.input_escapei,gcbi);
  3059:C        2 end;
  3060:S
  3061:D        1 procedure gle_input_escapeo (  gcbi : graphics_input_control_block_ptr );
  3062:S
  3063:C        2 begin
  3064:C        2   call (gcbi^.input_escapeo,gcbi);
  3065:C        2 end;
  3066:S
  3067:D        1 procedure gle_sample (  gcbi : graphics_input_control_block_ptr );
  3068:S
  3069:C        2 begin
  3070:C        2   call (gcbi^.sample,gcbi);
  3071:C        2 end;
```

```
3072:S
3073:D        1 procedure gle_start_digitize (  gcbi : graphics_input_control_block_ptr );
3074:S
3075:C        2 begin
3076:C        2   call (gcbi^.start_digitize,gcbi);
3077:C        2 end;
3078:S
3079:D        1 procedure gle_get_digitize (  gcbi : graphics_input_control_block_ptr );
3080:S
3081:C        2 begin
3082:C        2   call (gcbi^.get_digitize,gcbi);
3083:C        2 end;
3084:S
3085:C        1 end. { of module gle_geni }

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

GLE_HPGL

Description

GLE_HPGL contains the driver routines to drive HPGL output devices.

Requirements

GLE_TYPES, GLE_STEXT, GLE_ASTEXT, GLE_SCLIP, GLE_ASCLIP, GLE_SMARK, GLE_UTLS, and GLE_AUTL.

```
     1:D        0 {                                                                          }
     2:D        0 { Graphics Low End                                                         }
     3:D        0 {                                                                          }
     4:D        0 { Module    = GLE_HPGL_OUT                                                 }
     5:D        0 { Programer = BJS                                                          }
     6:D        0 { Date      = 10- 5-82                                                     }
     7:D        0 {                                                                          }
     8:D        0 { Purpose: To provide device dependent routines to drive hpgl output       }
     9:D        0 {          plotters.                                                       }
    10:S
    11:D        0 { Rev history                                                              }
    12:D        0 {  Created  - 10- 5-82  BJS                                                }
    13:D        0 {  Modified - 11-24-82  BJS  removed refs to str routines                  }
    14:D        0 {               6-28-83  BJS  added import of gle_astext (routine moved from }
    15:D        0 {                             gle_stext                                    }
    16:D        0 {               3-14-84  BDS  added identifiers for 7586B, 7550A, 7090, 7440 }
    17:S
    18:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    19:S                All rights are reserved.  Copying or other
    20:S                reproduction of this program except for archival
    21:S                purposes is prohibited without the prior
    22:S                written consent of Hewlett-Packard Company.
    23:S
    24:S
    25:S                           RESTRICTED RIGHTS LEGEND
    26:S
    27:S                Use, duplication, or disclosure by the Government
    28:S                is subject to restrictions as set forth in
    29:S                paragraph (b) (3) (B) of the Rights in Technical
    30:S                Data and Computer Software clause in
    31:S                DAR 7-104.9(a).
    32:S
    33:S                HEWLETT-PACKARD COMPANY
    34:D        0       Fort Collins, Colorado                                 }
    35:S
    36:D        0 $search 'GLE_TYPES',
    37:D        0         'GLE_SCLIP',
    38:D        0         'ASM_SCLIP',
    39:D        0         'GLE_STEXT',
    40:D        0         'ASM_STEXT',
    41:D        0         'GLE_SMARK',
    42:D        0         'GLE_UTLS'$
    43:D        0 $mocal$
    43:D        0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS ****************** }
    44:S
    45:S           { This include file specifies range checking, debug and other compiler
    46:D        0    options for the graphics library                                       }
    47:S
    48:D        0 $debug OFF$
    49:D        0 $range OFF$
    50:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    51:D        0 $FLOAT_HDW TEST$
    52:S
    53:S
    54:S
    55:S
    56:D        0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS ****************** }
  7000:D        0 $linenum 7000$
  7001:S
  7002:D        0 module GLE_HPGL_OUT;
```



```
  7003:S
  7004:D        1 import gle_types;
  7005:S
  7006:D        1 export
  7007:S
  7008:D        1 const
  7009:D        1   max_buffer = 255;
  7010:D        1   buffer_fudge = 32;
  7011:S
  7012:D        1 type
  7013:D        1   ascii_buffer_ptr = ^ascii_buffer;
  7014:S
  7015:D        1   ascii_buffer = packed record
  7016:D        1                    maximum : integer;
  7017:D        1                    current : integer;
  7018:D        1                    data    : packed array [1..max_buffer] of char;
  7019:D        1                  end;
  7020:S
  7021:D        1   driver_state_def = (moving,drawing,start_of_buffer,unknown);
  7022:S
  7023:D        1   hpgl_device_rec_ptr= ^ hpgl_device_rec;
  7024:D        1   hpgl_device_rec =
  7025:D        1     record
  7026:D        1       driver_state : driver_state_def;
  7027:D        1     end;
  7028:S
  7029:D        1 procedure gle_init_hpgl_output  (  gcb : graphics_control_block_ptr );
  7030:S
  7031:D        1 implement
  7032:S
  7033:D        1 import gle_stext,
  7034:D        1        gle_astext,
  7035:D        1        gle_asclip,
  7036:D        1        gle_sclip,
  7037:D        1        gle_smark,
  7038:D        1        gle_utls;
  7039:S
  7040:D        1 procedure hpgl_flush_buffer ( gcb : graphics_control_block_ptr );
  7041:S
  7042:C        2 begin
  7043:C        2   with gcb^,
  7044:C        3        ascii_buffer_ptr(device_buf)^,
  7045:C        3        hpgl_device_rec_ptr(dev_dep_stuff)^ do
  7046:C        3     begin
  7047:C        3       if current <> 0 then call (io_write,iocb,device_buf);
  7048:C        3       driver_state := start_of_buffer;
  7049:C        3     end;
  7050:C        2 end;
  7051:S
  7052:D        1 procedure dummy (  gcb : graphics_control_block_ptr );
  7053:S
  7054:C        2 begin
  7055:C        2 end;
  7056:S
  7057:D        1 procedure buffer_cleanup ( gcb : graphics_control_block_ptr );
  7058:S
  7059:C        2 begin
  7060:C        2   with gcb^,ascii_buffer_ptr(gcb^.device_buf)^ do
  7061:C        3     if (current > maximum - buffer_fudge) or (current_buffer_mode = 0) then
  7062:C        4       hpgl_flush_buffer ( gcb );
```

```
  7063:C        2 end;
  7064:S
  7065:D        1 procedure add_char_data (  gcb : graphics_control_block_ptr;
  7066:D        2                            count : gle_shortint;
  7067:D        2                            s : anychar_ptr );
  7068:D        2 var
  7069:D    -2  2   i : gle_shortint;
  7070:S
  7071:C        2 begin
  7072:C        2   with gcb^,ascii_buffer_ptr(gcb^.device_buf)^ do
  7073:C        3     begin
  7074:C        3       for i := 1 to count do
  7075:C        4         data[i+current] := s^[i];
  7076:C        3       current := current + count;
  7077:C        3     end;
  7078:C        2 end;
  7079:S
  7080:D        1 procedure add_parm_data ( gcb : graphics_control_block_ptr;
  7081:D        2                           value : gle_shortint);
  7082:D        2 var
  7083:D    -2  2   count : gle_shortint;
  7084:S
  7085:C        2 begin
  7086:C        2   with gcb^,ascii_buffer_ptr(gcb^.device_buf)^ do
  7087:C        3     begin
  7088:C        3       gle_write_integer (value,count,addr(data[current+1]));
  7089:C        3       current := current + count;
  7090:C        3     end;
  7091:C        2 end;
  7092:S
  7093:D        1 procedure change_state ( gcb : graphics_control_block_ptr;
  7094:D        2                          new_state : driver_state_def );
  7095:S
  7096:D        2 var
  7097:D    -1  2   change : boolean;
  7098:S
  7099:C        2 begin
  7100:C        2   with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ do
  7101:C        3     begin
  7102:C        3       change := new_state <> driver_state;
  7103:C        3       if (driver_state = unknown) or
  7104:C        4          (((driver_state = moving) or (driver_state = drawing)) and change) then
***WARNING: (line 7105): 'ADDR' of a constant may not be supported on other implementations
  7105:C        4         add_char_data(gcb,1,addr('; '));
  7106:C        3       if change then
  7107:C        4         begin
  7108:C        4           driver_state := new_state;
  7109:C        4           case driver_state of
***WARNING: (line 7110): 'ADDR' of a constant may not be supported on other implementations
  7110:C        5             moving  : add_char_data ( gcb, 5, addr('PU;PA') );
***WARNING: (line 7111): 'ADDR' of a constant may not be supported on other implementations
  7111:C        5             drawing : add_char_data ( gcb, 5, addr('PD;PA') );
  7112:C        5             start_of_buffer,unknown : ;
  7113:C        5           end; { of case }
  7114:C        4         end;
  7115:C        3     end;
  7116:C        2 end;
  7117:S
  7118:D        1 procedure hpgl_output_escapeo ( gcb : graphics_control_block_ptr );
  7119:S
```



```
  7120:C        2 begin
  7121:C        2   with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ ,
  7122:C        3        ascii_buffer_ptr(gcb^.device_buf)^do
  7123:C        3     begin
  7124:C        3       try
  7125:C        4        change_state ( gcb, unknown );
  7126:C        4        add_char_data(gcb,info1,anyptr(info_ptr1));
  7127:C        4        buffer_cleanup ( gcb );
  7128:C        4       recover
  7129:C        4         if escapecode = -20 then
  7130:C        5            begin
  7131:C        5              current := 0;
  7132:C        5              escape(-20);
  7133:C        5            end;
  7134:C        3     end;
  7135:C        2 end;
  7136:S
  7137:D        1 procedure hpgl_output_escapei ( gcb : graphics_control_block_ptr );
  7138:S
  7139:D        2 var
  7140:D    -2  2   i : gle_shortint;
  7141:D    -6  2   sptr : anychar_ptr;
  7142:S
  7143:C        2 begin
  7144:C        2   with gcb^,ascii_buffer_ptr(gcb^.device_buf)^ do
  7145:C        3     begin
  7146:C        3      try
  7147:C        4       call (io_read, iocb,  device_buf);
  7148:C        4       sptr := anyptr(info_ptr1);
  7149:C        4       info1 := current;
  7150:C        4       for i := 1 to current do
  7151:C        5         sptr^[i] := data[i];
  7152:C        4       current := 0;                    { reset buffer counter }
  7153:C        4      recover
  7154:C        4       if escapecode = -20 then
  7155:C        5          begin
  7156:C        5            current := 0;
  7157:C        5            escape(-20);
  7158:C        5          end;
  7159:C        3     end;
  7160:C        2 end;
  7161:S
  7162:D        1 procedure hpgl_get_p1p2 (  gcb : graphics_control_block_ptr );
  7163:S
  7164:D        2 var
  7165:D    -2  2   cnt    : gle_shortint;
  7166:D    -4  2   tcnt   : gle_shortint;
  7167:D    -8  2   temp   : integer;
  7168:S
  7169:C        2 begin
  7170:C        2   with gcb^,
  7171:C        3   ascii_buffer_ptr(gcb^.device_buf)^,
  7172:C        3   hpgl_device_rec_ptr(dev_dep_stuff)^ do
  7173:C        3    begin
  7174:C        3     try
  7175:C        4      if spooling = 0 then
  7176:C        5        begin
  7177:C        5          change_state ( gcb, unknown );
***WARNING: (line 7178): 'ADDR' of a constant may not be supported on other implementations
  7178:C        5          add_char_data ( gcb, 2, addr('CP') );
```

```
  7179:C        5           hpgl_flush_buffer ( gcb );
  7180:C        5           call(io_read, iocb,  device_buf);
  7181:C        5           tcnt := I;
  7182:C        5           info1 := gle_read_integer (current,addr(data[1]),cnt);
  7183:C        5           cnt := cnt + tcnt;
  7184:C        5           info3 := gle_read_integer (current,addr(data[cnt]),tcnt);
  7185:C        5           cnt := cnt + tcnt;
  7186:C        5           info2 := gle_read_integer (current,addr(data[cnt]),tcnt);
  7187:C        5           cnt := cnt + tcnt;
  7188:C        5           info4 := gle_read_integer (current,addr(data[cnt]),tcnt);
  7189:C        5           current := 0;
  7190:C        5           if info1 > info2 then     { make xmin <= xmax }
  7191:C        6             begin
  7192:C        6               temp := info1;
  7193:C        6               info1 := info2;
  7194:C        6               info2 := temp;
  7195:C        6             end;
  7196:C        5           if info3 > info4 then     { make ymin <= ymax }
  7197:C        6             begin
  7198:C        6               temp := info3;
  7199:C        6               info3 := info4;
  7200:C        6               info4 := temp;
  7201:C        6             end;
  7202:C        5         end
  7203:C        5       else
  7204:C        5         begin
  7205:C        5           info1 := display_min_x;
  7206:C        5           info2 := display_max_x;
  7207:C        5           info3 := display_min_y;
  7208:C        5           info4 := display_max_y;
  7209:C        5         end;
  7210:C        4     recover
  7211:C        4       if escapecode = -20 then
  7212:C        5         begin
  7213:C        5           current := 0;
  7214:C        5           escape(-20);
  7215:C        5         end;
  7216:C        3   end;
  7217:C        2 end;
  7218:S
  7219:D        1 procedure hpgl_get_hard_clip (  gcb : graphics_control_block_ptr );
  7220:S
  7221:D        2 var
  7222:D    -4  2   tcnt,cnt    : gle_shortint;
  7223:D   -12  2   tp1x,tp1y,tp2x,tp2y : gle_shortint;
  7224:S
  7225:C        2 begin
  7226:S
  7227:C        2   with gcb^,
  7228:C        3   ascii_buffer_ptr(gcb^.device_buf)^,
  7229:C        3   hpgl_device_rec_ptr(dev_dep_stuff)^ do
  7230:C        3     begin
  7231:C        3     try
***WARNING: (line 7232): 'ADDR' of a constant may not be supported on other implementations
  7232:C        4       if gle_match(4,addr(display_name),4,addr('7580')) or
***WARNING: (line 7233): 'ADDR' of a constant may not be supported on other implementations
  7233:C        5          gle_match(4,addr(display_name),4,addr('7585')) or
***WARNING: (line 7234): 'ADDR' of a constant may not be supported on other implementations
  7234:C        5          gle_match(4,addr(display_name),4,addr('7586')) then
  7235:C        5         begin
```



```
  7236:C        5           if spooling = 0 then
  7237:C        6             begin
***WARNING: (line 7238): 'ADDR' of a constant may not be supported on other implementations
  7238:C        6               add_char_data ( gcb, 2, addr('0H') );
  7239:C        6               hpgl_flush_buffer ( gcb );
  7240:C        6               call (io_read, iocb,  device_buf);
  7241:C        6               tcnt := I;
  7242:C        6               info1 := gle_read_integer (current,addr(data[1]),cnt);
  7243:C        6               cnt := cnt + tcnt;
  7244:C        6               info3 := gle_read_integer (current,addr(data[cnt]),tcnt);
  7245:C        6               cnt := cnt + tcnt;
  7246:C        6               info2 := gle_read_integer (current,addr(data[cnt]),tcnt);
  7247:C        6               cnt := cnt + tcnt;
  7248:C        6               info4 := gle_read_integer (current,addr(data[cnt]),tcnt);
  7249:C        6               current := 0;
  7250:C        6             end
  7251:C        6           else
  7252:C        6             begin
***WARNING: (line 7253): 'ADDR' of a constant may not be supported on other implementations
  7253:C        6               if ( gle_match(4,addr(display_name),4,addr('7586')) ) or
***WARNING: (line 7254): 'ADDR' of a constant may not be supported on other implementations
  7254:C        7                  ( gle_match(4,addr(display_name),4,addr('7585')) ) then
  7255:C        7                 begin
  7256:C        7                   info1 := -23656;            { E size }
  7257:C        7                   info2 :=  23656;
  7258:C        7                   info3 := -17962;
  7259:C        7                   info4 :=  17962;
  7260:C        7                 end
  7261:C        7               else
  7262:C        7                 begin
  7263:C        7                   info1 := -16190;            { D size }
  7264:C        7                   info2 :=  16190;
  7265:C        7                   info3 := -10485;
  7266:C        7                   info4 :=  10485;
  7267:C        7                 end;
  7268:C        6             end
  7269:C        6         end
  7270:C        5       else
***WARNING: (line 7271): 'ADDR' of a constant may not be supported on other implementations
  7271:C        5         if (gle_match(4,addr(display_name),4,addr('7470'))) or
***WARNING: (line 7272): 'ADDR' of a constant may not be supported on other implementations
  7272:C        6          (gle_match(4,addr(display_name),4,addr('7440'))) then
  7273:C        6         begin
  7274:C        6           info1 := 0;
  7275:C        6           info2 := 10300;
  7276:C        6           info3 := 0;
  7277:C        6           info4 := 7650;
  7278:C        6         end
  7279:C        6       else
***WARNING: (line 7280): 'ADDR' of a constant may not be supported on other implementations
  7280:C        6         if (gle_match(4,addr(display_name),4,addr('7475'))) or
***WARNING: (line 7281): 'ADDR' of a constant may not be supported on other implementations
  7281:C        7          (gle_match(4,addr(display_name),4,addr('7090'))) then
  7282:C        7         begin
  7283:C        7           info1 := 0;
  7284:C        7           info2 := 16640;
  7285:C        7           info3 := 0;
  7286:C        7           info4 := 10365;
  7287:C        7         end
  7288:C        7       else
```

```
***WARNING: (line 7289): 'ADDR' of a constant may not be supported on other implementations
  7289:C        7      if gle_match(4,addr(display_name),4,addr('7550')) then
  7290:C        8        begin
  7291:C        8          info1 := 0;
  7292:C        8          info2 := 16450;
  7293:C        8          info3 := 0;
  7294:C        8          info4 := 10170;
  7295:C        8        end
  7296:C        8      else
***WARNING: (line 7297): 'ADDR' of a constant may not be supported on other implementations
  7297:C        8      if gle_match(4,addr(display_name),4,addr('9872')) or (spooling = 1) then
  7298:C        9        begin
  7299:C        9          info1 := 0;
  7300:C        9          info2 := 16000;
  7301:C        9          info3 := 0;
  7302:C        9          info4 := 11400;
  7303:C        9        end
  7304:C        9      else
  7305:C        9        begin  { initialize the device and use P1/P2 values }
  7306:C        9          hpgl_get_p1p2 ( gcb );
  7307:C        9          tp1x := info1; tp2x := info2;
  7308:C        9          tp1y := info3; tp2y := info4;
***WARNING: (line 7309): 'ADDR' of a constant may not be supported on other implementations
  7309:C        9          add_char_data ( gcb, 3, addr('IN'));
  7310:C        9          hpgl_flush_buffer ( gcb );                { BDS 3/28/84 }
  7311:C        9          hpgl_get_p1p2 ( gcb );
  7312:C        9          { restore p1, p2 }
***WARNING: (line 7313): 'ADDR' of a constant may not be supported on other implementations
  7313:C        9          add_char_data ( gcb, 2, addr('IP'));
  7314:C        9          add_parm_data ( gcb, tp1x );
***WARNING: (line 7315): 'ADDR' of a constant may not be supported on other implementations
  7315:C        9          add_char_data ( gcb, 1, addr(', '));
  7316:C        9          add_parm_data ( gcb, tp1y );
***WARNING: (line 7317): 'ADDR' of a constant may not be supported on other implementations
  7317:C        9          add_char_data ( gcb, 1, addr(', '));
  7318:C        9          add_parm_data ( gcb, tp2x );
***WARNING: (line 7319): 'ADDR' of a constant may not be supported on other implementations
  7319:C        9          add_char_data ( gcb, 1, addr(', '));
  7320:C        9          add_parm_data ( gcb, tp2y );
  7321:C        9          hpgl_flush_buffer ( gcb );
  7322:C        9        end;
  7323:C        4      recover
  7324:C        4       if escapecode = -20 then
  7325:C        5          begin
  7326:C        5            current := 0;
  7327:C        5            escape(-20);
  7328:C        5          end;
  7329:C        3      end;
  7330:C        2 end;
  7331:S
  7332:D        1 procedure hpgl_move (  gcb : graphics_control_block_ptr );
  7333:S
  7334:C        2 begin
  7335:C        2   with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ ,
  7336:C        3        ascii_buffer_ptr(device_buf)^ do
  7337:C        3     begin
  7338:C        3       try
  7339:C        4         if driver_state = moving then
***WARNING: (line 7340): 'ADDR' of a constant may not be supported on other implementations
  7340:C        5           add_char_data ( gcb, 1, addr(', '))
```



```
  7341:C        5         else
  7342:C        5           change_state ( gcb, moving );
  7343:C        4         add_parm_data ( gcb, end_x);
***WARNING: (line 7344): 'ADDR' of a constant may not be supported on other implementations
  7344:C        4         add_char_data ( gcb, 1, addr(', '));
  7345:C        4         add_parm_data ( gcb, end_y);
  7346:C        4         current_pos_x := end_x;
  7347:C        4         current_pos_y := end_y;
  7348:C        4         buffer_cleanup ( gcb );
  7349:C        4       recover
  7350:C        4         if escapecode = -20 then
  7351:C        5            begin
  7352:C        5              current := 0;
  7353:C        5              escape(-20);
  7354:C        5            end;
  7355:C        3     end;
  7356:C        2 end;
  7357:S
  7358:D        1 procedure hpgl_draw (  gcb : graphics_control_block_ptr );
  7359:S
  7360:C        2 begin
  7361:C        2   with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ ,
  7362:C        3        ascii_buffer_ptr(device_buf)^ do
  7363:C        3     begin
  7364:C        3       try
  7365:C        4         if driver_state = drawing then
***WARNING: (line 7366): 'ADDR' of a constant may not be supported on other implementations
  7366:C        5           add_char_data ( gcb, 1, addr(', '))
  7367:C        5         else
  7368:C        5           change_state ( gcb, drawing );
  7369:C        4         add_parm_data ( gcb, end_x);
***WARNING: (line 7370): 'ADDR' of a constant may not be supported on other implementations
  7370:C        4         add_char_data ( gcb, 1, addr(', '));
  7371:C        4         add_parm_data ( gcb, end_y);
  7372:C        4         current_pos_x := end_x;
  7373:C        4         current_pos_y := end_y;
  7374:C        4         buffer_cleanup ( gcb );
  7375:C        4       recover
  7376:C        4         if escapecode = -20 then
  7377:C        5            begin
  7378:C        5              current := 0;
  7379:C        5              escape(-20);
  7380:C        5            end;
  7381:C        3     end;
  7382:C        2 end;
  7383:S
  7384:D        1 procedure hpgl_buffer_mode(  gcb : graphics_control_block_ptr );
  7385:S
  7386:C        2 begin
  7387:C        2  with gcb^ ,ascii_buffer_ptr(device_buf)^ do
  7388:C        3  begin
  7389:C        3   try
  7390:C        4    hpgl_flush_buffer ( gcb );
  7391:C        4    current_buffer_mode := info1;
  7392:C        4   recover
  7393:C        4      if escapecode = -20 then
  7394:C        5         begin
  7395:C        5           current := 0;
  7396:C        5           escape(-20);
  7397:C        5         end;
```

```
  7398:C        3  end;
  7399:C        2 end;
  7400:S
  7401:D        1 procedure hpgl_set_color (  gcb : graphics_control_block_ptr );
  7402:S
  7403:C        2 begin
  7404:C        2   with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ ,
  7405:C        3        ascii_buffer_ptr(gcb^.device_buf)^ do
  7406:C        3     begin
  7407:C        3       try
  7408:C        4        change_state ( gcb, unknown );
***WARNING: (line 7409): 'ADDR' of a constant may not be supported on other implementations
  7409:C        4        add_char_data ( gcb, 5, addr('PU;SP') );
  7410:C        4        add_parm_data ( gcb, info1 );
  7411:C        4        buffer_cleanup ( gcb );
  7412:C        4        current_color_index := info1;
  7413:C        4       recover
  7414:C        4         if escapecode = -20 then
  7415:C        5             begin
  7416:C        5               current := 0;
  7417:C        5               escape(-20);
  7418:C        5             end;
  7419:C        3     end;
  7420:C        2 end;
  7421:S
  7422:S          (********************** this procedure is never used ************************
  7423:S          procedure hpgl_fill_index_color (  gcb : graphics_control_block_ptr );
  7424:S
  7425:S          begin
  7426:S            with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ do
  7427:S              current_polygon_color := info1;
  7428:D        1 end;}
  7429:S
  7430:D        1 procedure hpgl_linestyle (  gcb : graphics_control_block_ptr );
  7431:S
  7432:C        2 begin
  7433:C        2   with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ ,
  7434:C        3             ascii_buffer_ptr(gcb^.device_buf)^ do
  7435:C        3     begin
  7436:C        3       try
  7437:C        4        change_state ( gcb, unknown );
  7438:C        4        current_linestyle := info1;
  7439:C        4        current_pattern_length := info2;
  7440:C        4        current_linestyle_mode := info3;
  7441:C        4        current_linestyle_pattern := info4;
***WARNING: (line 7442): 'ADDR' of a constant may not be supported on other implementations
  7442:C        4        add_char_data(gcb,2,addr('LT'));
  7443:C        4        if info1 = 0 then
  7444:C        5          begin
  7445:C        5          end
  7446:C        5        else
  7447:C        5        if info1 = 7 then
  7448:C        6          add_parm_data(gcb,0)
  7449:C        6        else
  7450:C        6          begin
  7451:C        6            if info3 = 0 then
  7452:C        7              add_parm_data(gcb,info1)
  7453:C        7            else
  7454:C        7              add_parm_data(gcb,-info1);
***WARNING: (line 7455): 'ADDR' of a constant may not be supported on other implementations
```



```
  7455:C        6            add_char_data(gcb,1,addr(', '));
  7456:C        6            add_parm_data(gcb,info2);
  7457:C        6          end;
  7458:C        4        buffer_cleanup ( gcb );
  7459:C        4       recover
  7460:C        4         if escapecode = -20 then
  7461:C        5             begin
  7462:C        5               current := 0;
  7463:C        5               escape(-20);
  7464:C        5             end;
  7465:C        3     end;
  7466:C        2 end;
  7467:S
  7468:D        1 procedure hpgl_clear(  gcb : graphics_control_block_ptr );
  7469:S
  7470:C        2 begin
  7471:C        2   with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ ,
  7472:C        3             ascii_buffer_ptr(gcb^.device_buf)^do
  7473:C        3     begin
  7474:C        3       try
  7475:C        4        change_state ( gcb, unknown );
***WARNING: (line 7476): 'ADDR' of a constant may not be supported on other implementations
  7476:C        4        add_char_data(gcb,2,addr('PG'));
  7477:C        4        buffer_cleanup ( gcb );
  7478:C        4       recover
  7479:C        4         if escapecode = -20 then
  7480:C        5             begin
  7481:C        5               current := 0;
  7482:C        5               escape(-20);
  7483:C        5             end;
  7484:C        3     end;
  7485:C        2 end;
  7486:S
  7487:D        1 procedure hpgl_cursor (  gcb : graphics_control_block_ptr );
  7488:S
  7489:C        2 begin
  7490:C        2   with gcb^,hpgl_device_rec_ptr(dev_dep_stuff)^ ,
  7491:C        3             ascii_buffer_ptr(gcb^.device_buf)^do
  7492:C        3     begin
  7493:C        3       try
  7494:C        4        change_state  ( gcb, unknown );
***WARNING: (line 7495): 'ADDR' of a constant may not be supported on other implementations
  7495:C        4        add_char_data ( gcb, 5, addr('PU;PR'));
  7496:C        4        add_parm_data ( gcb, info1);
***WARNING: (line 7497): 'ADDR' of a constant may not be supported on other implementations
  7497:C        4        add_char_data ( gcb, 1, addr(', '));
  7498:C        4        add_parm_data ( gcb, info2);
  7499:C        4        current_cursor_state := info3;
  7500:C        4        hpgl_flush_buffer ( gcb );
  7501:C        4       recover
  7502:C        4         if escapecode = -20 then
  7503:C        5             begin
  7504:C        5               current := 0;
  7505:C        5               escape(-20);
  7506:C        5             end;
  7507:C        3     end;
  7508:C        2 end;
  7509:S
  7510:D        1 procedure gle_init_hpgl_output (  gcb : graphics_control_block_ptr);
  7511:S
```

```
  7512:D        2 var
  7513:D    -4  2   saved_timeout : integer;
  7514:D    -6  2   i : gle_shortint;
  7515:S
  7516:C        2 begin
  7517:C        2   with gcb^,
  7518:C        3        hpgl_device_rec_ptr(dev_dep_stuff)^,
  7519:C        3        ascii_buffer_ptr(device_buf)^ do
  7520:C        3     try
  7521:C        4       current := 0;
  7522:C        4       maximum := max_buffer;
  7523:S
  7524:C        4       driver_state          := start_of_buffer;
  7525:C        4       unclipped_move        := hpgl_move;
  7526:C        4       unclipped_draw        := hpgl_draw;
  7527:C        4       move                  := gle_soft_clip_move;
  7528:C        4       draw                  := gle_soft_clip_draw;
  7529:C        4       clear                 := hpgl_clear;
  7530:C        4       text                  := gle_soft_text (hpgl_text);
  7531:C        4       char_size             := gle_soft_char_size;
  7532:C        4       clip_limits           := gle_soft_clip_limits;
  7533:C        4       text_spacing          := gle_soft_text_spacing;
  7534:C        4       linestyle             := hpgl_linestyle;
  7535:C        4       text_dir              := gle_soft_text_dir;
  7536:C        4       text_just             := gle_soft_text_just;
  7537:C        4       marker                := gle_soft_marker;
  7538:C        4       marker_size           := gle_soft_marker_size;
  7539:C        4       set_marker            := gle_soft_set_marker;
  7540:C        4       index_color           := hpgl_set_color;
  7541:C        4       inq_pip2              := hpgl_get_pip2;
  7542:C        4       get_polygon_info      := dummy;
  7543:C        4       graphics_on_off       := dummy;
  7544:C        4       cursor                := hpgl_cursor;
  7545:C        4       calc_soft_text_xform  := gle_text_xform;
  7546:C        4       buffer_mode           := hpgl_buffer_mode;
  7547:C        4       output_escapeo        := hpgl_output_escapeo;
  7548:C        4       output_escapei        := hpgl_output_escapei;
  7549:C        4       define_drawing_mode   := dummy;
  7550:C        4       define_color_map      := dummy;
  7551:C        4       polygon               := dummy;
  7552:C        4       fill_index_color      := dummy;
  7553:C        4       linewidth             := dummy;
  7554:C        4       gload                 := dummy;
  7555:C        4       gstore                := dummy;
  7556:C        4       get_raster            := dummy;
  7557:C        4       get_color_map         := dummy;
  7558:C        4       await_blanking        := dummy;
  7559:C        4       flush_buffer          := hpgl_flush_buffer;
  7560:S
  7561:C        4       soft_font_ptr         := addr(font);
  7562:S
  7563:C        4       error_return := 0;
  7564:C        4       if spooling = 0 then
  7565:C        5         try
  7566:C        6           call (io_inq_timeout, iocb, saved_timeout );
  7567:C        6           call (io_set_timeout, iocb, SCO ( ms ) );
  7568:C        6           { send command that all HPGL plotters can respond to   }
  7569:C        6           { if the command fails then the address does not match }
  7570:C        6           { the device.                                          }
***WARNING: (line 7571): 'ADDR' of a constant may not be supported on other implementations
```



```
  7571:C        6           add_char_data ( gcb, 2, addr('OE') );
  7572:C        6           hpgl_flush_buffer ( gcb );
  7573:C        6           call (io_read, iocb,  device_buf);
  7574:C        6           current := 0;
  7575:S
  7576:C        6           { if this point is reached then a vaild HPGL device was found }
  7577:S
  7578:C        6           try  { perform an output identify seq.  Note a 9872A will fail }
***WARNING: (line 7579): 'ADDR' of a constant may not be supported on other implementations
  7579:C        7             add_char_data ( gcb, 2, addr('OI') );
  7580:C        7             hpgl_flush_buffer ( gcb );
  7581:C        7             call (io_read, iocb,  device_buf);
  7582:S
  7583:C        7             for i := 1 to current do display_name[i] := data[i];
  7584:C        7             for i := current+1 to 6 do display_name[i] := ' ';
  7585:C        7             display_name_char_count := current;
  7586:C        7             current := 0;
  7587:C        7           recover
  7588:C        7             if escapecode = -26 { io system } then
  7589:C        8               begin
  7590:C        8                 display_name := '9872A ';
  7591:C        8                 display_name_char_count := 5;
  7592:C        8                 current := 0;
  7593:C        8               end
  7594:C        8             else if escapecode = -20 {stop key} then
  7595:C        9                begin
  7596:C        9                  current := 0;
  7597:C        9                  escape(-20);
  7598:C        9                end
  7599:C        9             else escape(escapecode);
  7600:S
  7601:C        6          recover
  7602:C        6            if escapecode = -26 then error_return := 1
  7603:C        7            else if escapecode = -20 {stop key} then
  7604:C        8                 begin
  7605:C        8                   current := 0;
  7606:C        8                   escape(-20);
  7607:C        8                 end
  7608:C        8            else escape(escapecode); { ignor io errors }
  7609:S
  7610:C        4       call (io_set_timeout, iocb, saved_timeout );
  7611:S
  7612:C        4       if error_return = 0 then
  7613:C        5         begin
***WARNING: (line 7614): 'ADDR' of a constant may not be supported on other implementations
  7614:C        5           add_char_data ( gcb, 12, addr('DF;SP1;IM30;') );
  7615:C        5           hpgl_flush_buffer ( gcb );
***WARNING: (line 7616): 'ADDR' of a constant may not be supported on other implementations
  7616:C        5           if gle_match(4,addr(display_name),4,addr('9111')) then { 9111 is input only }
  7617:C        6             escape(-26) { force io error }
  7618:C        6           else
***WARNING: (line 7619): 'ADDR' of a constant may not be supported on other implementations
  7619:C        6             if gle_match(display_name_char_count,addr(display_name),5,addr('7470A')) or
***WARNING: (line 7620): 'ADDR' of a constant may not be supported on other implementations
  7620:C        7                gle_match(display_name_char_count,addr(display_name),4,addr('7470')) then
  7621:C        7               begin
  7622:C        7                 pallette := 2;
  7623:C        7                 cont_linestyles := 8;
  7624:C        7                 vect_linestyles := 0;
  7625:C        7               end
```

```
  7626:C         7              else
***WARNING: (line 7627): 'ADDR' of a constant may not be supported on other implementations
  7627:C         7                if gle_match(display_name_char_count,addr(display_name),4,addr('7440')) then
  7628:C         8                  begin
  7629:C         8                    pallette := 8;
  7630:C         8                    cont_linestyles := 8;
  7631:C         8                    vect_linestyles := 0;
  7632:C         8                  end
  7633:C         8                else
***WARNING: (line 7634): 'ADDR' of a constant may not be supported on other implementations
  7634:C         8                  if gle_match(display_name_char_count,addr(display_name),5,addr('7475A')) or
***WARNING: (line 7635): 'ADDR' of a constant may not be supported on other implementations
  7635:C         9                    gle_match(display_name_char_count,addr(display_name),4,addr('7475')) or
***WARNING: (line 7636): 'ADDR' of a constant may not be supported on other implementations
  7636:C         9                    gle_match(display_name_char_count,addr(display_name),4,addr('7090')) then
  7637:C         9                    begin
  7638:C         9                      pallette := 6;
  7639:C         9                      cont_linestyles := 8;
  7640:C         9                      vect_linestyles := 0;
  7641:C         9                    end
  7642:C         9                  else
***WARNING: (line 7643): 'ADDR' of a constant may not be supported on other implementations
  7643:C         9                    if (gle_match(display_name_char_count,addr(display_name),5,addr('7580A'))) or
***WARNING: (line 7644): 'ADDR' of a constant may not be supported on other implementations
  7644:C        10                     (gle_match(display_name_char_count,addr(display_name),5,addr('7580B'))) or
***WARNING: (line 7645): 'ADDR' of a constant may not be supported on other implementations
  7645:C        10                     (gle_match(display_name_char_count,addr(display_name),4,addr('7580'))) or
***WARNING: (line 7646): 'ADDR' of a constant may not be supported on other implementations
  7646:C        10                     (gle_match(display_name_char_count,addr(display_name),5,addr('7585A'))) or
***WARNING: (line 7647): 'ADDR' of a constant may not be supported on other implementations
  7647:C        10                     (gle_match(display_name_char_count,addr(display_name),5,addr('7585B'))) or
***WARNING: (line 7648): 'ADDR' of a constant may not be supported on other implementations
  7648:C        10                     (gle_match(display_name_char_count,addr(display_name),4,addr('7585'))) or
***WARNING: (line 7649): 'ADDR' of a constant may not be supported on other implementations
  7649:C        10                     (gle_match(display_name_char_count,addr(display_name),5,addr('7586B'))) or
***WARNING: (line 7650): 'ADDR' of a constant may not be supported on other implementations
  7650:C        10                     (gle_match(display_name_char_count,addr(display_name),5,addr('7586'))) or
***WARNING: (line 7651): 'ADDR' of a constant may not be supported on other implementations
  7651:C        10                     (gle_match(display_name_char_count,addr(display_name),5,addr('7550A'))) or
***WARNING: (line 7652): 'ADDR' of a constant may not be supported on other implementations
  7652:C        10                     (gle_match(display_name_char_count,addr(display_name),5,addr('7550'))) then
  7653:C        10                   begin
  7654:C        10                     pallette := 8;
  7655:C        10                     cont_linestyles := 8;
  7656:C        10                     vect_linestyles := 8;
  7657:C        10                   end
  7658:C        10                 else
  7659:C        10                   begin
***WARNING: (line 7660): 'ADDR' of a constant may not be supported on other implementations
  7660:C        10                     if (gle_match(display_name_char_count,addr(display_name),5,addr('9872A'))) or
***WARNING: (line 7661): 'ADDR' of a constant may not be supported on other implementations
  7661:C        11                       (gle_match(display_name_char_count,addr(display_name),5,addr('9872B'))) or
***WARNING: (line 7662): 'ADDR' of a constant may not be supported on other implementations
  7662:C        11                       (gle_match(display_name_char_count,addr(display_name),5,addr('9872S'))) or
***WARNING: (line 7663): 'ADDR' of a constant may not be supported on other implementations
  7663:C        11                       (gle_match(display_name_char_count,addr(display_name),4,addr('9872'))) then
  7664:C        11                     pallette   := 4
  7665:C        11                   else
***WARNING: (line 7666): 'ADDR' of a constant may not be supported on other implementations
  7666:C        11                     if (gle_match(display_name_char_count,addr(display_name),5,addr('9872C'))) or
```



```
***WARNING: (line 7667): 'ADDR' of a constant may not be supported on other implementations
  7667:C        12                       (gle_match(display_name_char_count,addr(display_name),5,addr('9872T'))) then
  7668:C        12                     pallette := 8
  7669:C        12                   else
  7670:C        12                     if spooling = 0 then pallette := 4   { assume 9872 like device }
  7671:C        13                   else
  7672:C        13                     escape(-26); { can't init device, force io error }
  7673:S
  7674:C        10                   cont_linestyles := 8;
  7675:C        10                   vect_linestyles := 0;
  7676:C        10                 end;
  7677:S
  7678:C         5             hpgl_get_hard_clip ( gcb );
  7679:C         5             display_min_x := info1;
  7680:C         5             display_min_y := info3;
  7681:C         5             display_max_x := info2;
  7682:C         5             display_max_y := info4;
  7683:S
  7684:C         5             gle_soft_clip_limits ( gcb );  { set default clipping limits }
  7685:S
  7686:C         5             gamut := pallette;
  7687:C         5             polygon_support := 0;              { polygon routine dummyed out }
  7688:C         5             display_handler_name := 'HPGL  ';
  7689:C         5             display_handler_char_count := 4;
  7690:S
  7691:C         5             display_res_x := 40;
  7692:C         5             display_res_y := 40;
  7693:S
  7694:C         5             linewidths :=  1;
  7695:C         5             char_sizes := -1;
  7696:C         5             background :=  0;
  7697:C         5             complement_support := 0;
  7698:C         5             non_dominant_support := 0;
  7699:C         5             erase_support := 0;
  7700:C         5             color_map_support := 0;
  7701:C         5             redef_background  := 0;
  7702:S
  7703:C         5             polygon_fill_factor := 16;
  7704:C         5             polygon_solid_fill  := 5;
  7705:C         5             dither_support      := 0;
  7706:S
  7707:C         5             current_pos_x := 0;
  7708:C         5             current_pos_y := 0;
  7709:C         5             current_cursor_state := 0; { off }
  7710:S
  7711:C         5             current_buffer_mode := 0;  { imed visb }
  7712:C         5           end;
  7713:C         4       recover
  7714:C         4         begin
  7715:C         4           if escapecode = -26 then error_return := 1
  7716:C         5           else if escapecode = -20 {stop key} then
  7717:C         6             begin
  7718:C         6               current := 0;
  7719:C         6               escape(-20);
  7720:C         6             end
  7721:C         6           else escape(escapecode); { ignor io errors }
  7722:C         4         end;
  7723:C         2 end;
  7724:S
  7725:C         1 end. { hpgl_output }
```

```
  7726:S
No errors. 57 warnings.
              ***** Nonstandard language features enabled *****
```

GLE_HPGLI

Description

GLE_HPGLI contains the driver code for HPGL input devices.

Requirements

GLE_TYPES, and GLE_UTLS.

```
     1:D        0 {                                                                                 }
     2:D        0 { Graphics Low End                                                                }
     3:D        0 {                                                                                 }
     4:D        0 { Module    = GLE_HPGL_IN                                                         }
     5:D        0 { Programer = BJS                                                                 }
     6:D        0 { Date      = 10-10-82                                                            }
     7:D        0 {                                                                                 }
     8:D        0 { Purpose: To provide HPGL input handler routines.                                }
     9:S
    10:D        0 { Rev history                                                                     }
    11:D        0 {  Created  - 10-10-82                                                            }
    12:D        0 {  Modified - 03-15-84 BDS Added support for 7586B, 7550A, 7440, 7090             }
    13:S
    14:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    15:S                All rights are reserved.  Copying or other
    16:S                reproduction of this program except for archival
    17:S                purposes is prohibited without the prior
    18:S                written consent of Hewlett-Packard Company.
    19:S
    20:S
    21:S                            RESTRICTED RIGHTS LEGEND
    22:S
    23:S                Use, duplication, or disclosure by the Government
    24:S                is subject to restrictions as set forth in
    25:S                paragraph (b) (3) (B) of the Rights in Technical
    26:S                Data and Computer Software clause in
    27:S                DAR 7-104.9(a).
    28:S
    29:S                HEWLETT-PACKARD COMPANY
    30:D        0       Fort Collins, Colorado                                                      }
    31:S
    32:S
    33:D        0 $SEARCH 'GLE_TYPES','GLE_UTLS','GEN_TOOLS'$
    34:D        0 $modcal$
    34:D        0 $INCLUDE 'OPTIONS'$  { ******************** COMPILER OPTIONS ******************** }
    35:S
    36:S          { This include file specifies range checking, debug and other compiler
    37:D        0   options for the graphics library                                                }
    38:S
    39:D        0 $debug OFF$
    40:D        0 $range OFF$
    41:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    42:D        0 $FLOAT_HDW TEST$
    43:S
    44:S
    45:S
    46:S
    47:D        0 $INCLUDE 'OPTIONS'$  { ******************** COMPILER OPTIONS ******************** }
 18000:D        0 $LINENUM 18000$
 18001:S
 18002:D        0 module gle_hpgl_in;
 18003:S
 18004:D        1 import gle_types;
 18005:S
 18006:D        1 export
 18007:S
 18008:D        1 const
 18009:D        1   max_buffer = 255;
 18010:D        1   buffer_fudge = 32;
 18011:S
```



```
 18012:D        1 type
 18013:D        1   ascii_buffer_ptr = ^ascii_buffer;
 18014:S
 18015:D        1   ascii_buffer = packed record
 18016:D        1                   maximum : integer;
 18017:D        1                   current : integer;
 18018:D        1                   data    : packed array [1..max_buffer] of char;
 18019:D        1                 end;
 18020:S
 18021:D        1 procedure gle_init_hpgl_input (  gcbi : graphics_input_control_block_ptr);
 18022:S
 18023:D        1 implement
 18024:S
 18025:S          import gle_utls(,
 18026:D        1        gen_tools);
 18027:S
 18028:D        1 procedure hpgl_flush_buffer ( gcbi : graphics_input_control_block_ptr );
 18029:S
 18030:C        2 begin
 18031:C        2   with gcbi^,
 18032:C        3        ascii_buffer_ptr(device_buf)^ do
 18033:C        3     begin
 18034:C        3       if current <> 0 then call (io_write,iocb,device_buf);
 18035:C        3     end;
 18036:C        2 end;
 18037:S
 18038:D        1 procedure add_char_data (  gcbi : graphics_input_control_block_ptr;
 18039:D        2                           count : gle_shortint;
 18040:D        2                           s : anychar_ptr );
 18041:D        2 var
 18042:D    -2  2   i : gle_shortint;
 18043:S
 18044:C        2 begin
 18045:C        2   with gcbi^,ascii_buffer_ptr(gcbi^.device_buf)^ do
 18046:C        3     begin
 18047:C        3       for i := 1 to count do
 18048:C        4         data[i+current] := s^[i];
 18049:C        3       current := current + count;
 18050:C        3     end;
 18051:C        2 end;
 18052:S
 18053:D        1 procedure add_parm_data ( gcbi : graphics_input_control_block_ptr;
 18054:D        2                           value : gle_shortint);
 18055:D        2 var
 18056:D    -2  2   count : gle_shortint;
 18057:S
 18058:C        2 begin
 18059:C        2   with gcbi^,ascii_buffer_ptr(gcbi^.device_buf)^ do
 18060:C        3     begin
 18061:C        3       gle_write_integer (value,count,addr(data[current+1]));
 18062:C        3       current := current + count;
 18063:C        3     end;
 18064:C        2 end;
 18065:S
 18066:D        1 procedure hpgl_input_escapeo ( gcbi : graphics_input_control_block_ptr );
 18067:S
 18068:C        2 begin
 18069:C        2   with gcbi^ do
 18070:C        3     begin
 18071:C        3       add_char_data(gcbi,info1,anyptr(info_ptr1));
```

```
 18072:C        3       hpgl_flush_buffer ( gcbi );
 18073:C        3     end;
 18074:C        2 end;
 18075:S
 18076:D        1 procedure hpgl_input_escapei ( gcbi : graphics_input_control_block_ptr );
 18077:S
 18078:D        2 var
 18079:D    -2  2   i : gle_shortint;
 18080:D    -6  2   sptr : anychar_ptr;
 18081:S
 18082:C        2 begin
 18083:C        2   with gcbi^,ascii_buffer_ptr(gcbi^.device_buf)^ do
 18084:C        3     begin
 18085:C        3       call (io_read, iocb,  device_buf);
 18086:C        3       sptr := anyptr(info_ptr1);
 18087:C        3       info1 := current;
 18088:C        3       for i := 1 to current do sptr^[i] := data[i];
 18089:C        3       current := 0;                                ( reset buffer counter )
 18090:C        3     end;
 18091:C        2 end;
 18092:S
 18093:D        1 procedure hpgl_get_input_p1p2 (  gcbi : graphics_input_control_block_ptr );
 18094:S
 18095:D        2 var
 18096:D    -2  2   cnt    : gle_shortint;
 18097:D    -4  2   tcnt   : gle_shortint;
 18098:D    -8  2   temp   : integer;
 18099:S
 18100:C        2 begin
 18101:C        2   with gcbi^,
 18102:C        3   ascii_buffer_ptr(gcbi^.device_buf)^ do
 18103:C        3     begin
***WARNING: (line18104): 'ADDR' of a constant may not be supported on other implementations
 18104:C        3       add_char_data ( gcbi, 2, addr('OP') );
 18105:C        3       hpgl_flush_buffer ( gcbi );
 18106:C        3       call (io_read, iocb,  device_buf);
 18107:C        3       tcnt := 1;
 18108:C        3       info1 := gle_read_integer (current,addr(data[1]),cnt);
 18109:C        3       cnt := cnt + tcnt;
 18110:C        3       info3 := gle_read_integer (current,addr(data[cnt]),tcnt);
 18111:C        3       cnt := cnt + tcnt;
 18112:C        3       info2 := gle_read_integer (current,addr(data[cnt]),tcnt);
 18113:C        3       cnt := cnt + tcnt;
 18114:C        3       info4 := gle_read_integer (current,addr(data[cnt]),tcnt);
 18115:C        3       current := 0;
 18116:C        3       if info1 > info2 then     { make xmin <= xmax }
 18117:C        4         begin
 18118:C        4           temp := info1;
 18119:C        4           info1 := info2;
 18120:C        4           info2 := temp;
 18121:C        4         end;
 18122:C        3       if info3 > info4 then     { make ymin <= ymax }
 18123:C        4         begin
 18124:C        4           temp := info3;
 18125:C        4           info3 := info4;
 18126:C        4           info4 := temp;
 18127:C        4         end;
 18128:C        3     end;
 18129:C        2 end;
 18130:S
```



```
 18131:D        1 procedure hpgl_get_input_hard_clip (  gcbi : graphics_input_control_block_ptr );
 18132:S
 18133:D        2 var
 18134:D    -4  2   tcnt,cnt    : gle_shortint;
 18135:D   -12  2   tp1x,tp1y,tp2x,tp2y : gle_shortint;
 18136:S
 18137:C        2 begin
 18138:C        2   with gcbi^,
 18139:C        3   ascii_buffer_ptr(gcbi^.device_buf)^ do
***WARNING: (line18140): 'ADDR' of a constant may not be supported on other implementations
 18140:C        3     if gle_match(4,addr(input_name),4,addr('7580')) or
***WARNING: (line18141): 'ADDR' of a constant may not be supported on other implementations
 18141:C        4        gle_match(4,addr(input_name),4,addr('7585')) or
***WARNING: (line18142): 'ADDR' of a constant may not be supported on other implementations
 18142:C        4        gle_match(4,addr(input_name),4,addr('7586')) then
 18143:C        4       begin
***WARNING: (line18144): 'ADDR' of a constant may not be supported on other implementations
 18144:C        4         add_char_data ( gcbi, 2, addr('OH') );
 18145:C        4         hpgl_flush_buffer ( gcbi );
 18146:C        4         call (io_read, iocb,  device_buf);
 18147:C        4         tcnt := 1;
 18148:C        4         info1 := gle_read_integer (current,addr(data[1]),cnt);
 18149:C        4         cnt := cnt + tcnt;
 18150:C        4         info3 := gle_read_integer (current,addr(data[cnt]),tcnt);
 18151:C        4         cnt := cnt + tcnt;
 18152:C        4         info2 := gle_read_integer (current,addr(data[cnt]),tcnt);
 18153:C        4         cnt := cnt + tcnt;
 18154:C        4         info4 := gle_read_integer (current,addr(data[cnt]),tcnt);
 18155:C        4         current := 0;
 18156:C        4       end
 18157:C        4     else
***WARNING: (line18158): 'ADDR' of a constant may not be supported on other implementations
 18158:C        4       if gle_match(4,addr(input_name),4,addr('7470')) or
***WARNING: (line18159): 'ADDR' of a constant may not be supported on other implementations
 18159:C        5        gle_match(4,addr(input_name),4,addr('7440')) then
 18160:C        5       begin
 18161:C        5         info1 := 0;
 18162:C        5         info2 := 10300;
 18163:C        5         info3 := 0;
 18164:C        5         info4 := 7650;                (changed from 7600 per 7550 doc)
 18165:C        5       end
 18166:C        5     else
***WARNING: (line18167): 'ADDR' of a constant may not be supported on other implementations
 18167:C        5       if gle_match(4,addr(input_name),4,addr('7475')) or
***WARNING: (line18168): 'ADDR' of a constant may not be supported on other implementations
 18168:C        6        gle_match(4,addr(input_name),4,addr('7090')) then
 18169:C        6       begin
 18170:C        6         info1 := 0;
 18171:C        6         info2 := 16640;
 18172:C        6         info3 := 0;
 18173:C        6         info4 := 10365;
 18174:C        6       end
 18175:C        6     else
***WARNING: (line18176): 'ADDR' of a constant may not be supported on other implementations
 18176:C        6       if gle_match(4,addr(input_name),4,addr('7550')) then
 18177:C        7       begin
 18178:C        7         info1 := 0;
 18179:C        7         info2 := 16450;
 18180:C        7         info3 := 0;
 18181:C        7         info4 := 10170;
```

```
 18182:C        7       end
 18183:C        7     else
***WARNING: (line18184): 'ADDR' of a constant may not be supported on other implementations
 18184:C        7     if gle_match(4,addr(input_name),4,addr('9872')) then
 18185:C        8       begin
 18186:C        8         info1 := 0;
 18187:C        8         info2 := 16000;
 18188:C        8         info3 := 0;
 18189:C        8         info4 := 11400;
 18190:C        8       end
 18191:C        8     else
***WARNING: (line18192): 'ADDR' of a constant may not be supported on other implementations
 18192:C        8     if gle_match(4,addr(input_name),4,addr('9111')) then
 18193:C        9       begin
 18194:C        9         info1 := 0;
 18195:C        9         info2 := 12032;
 18196:C        9         info3 := 0;
 18197:C        9         info4 := 8704;
 18198:C        9       end
 18199:C        9     else
 18200:C        9       begin  { initialize the device and use P1/P2 values }
 18201:C        9         hpgl_get_input_p1p2 ( gcbi );
 18202:C        9         tp1x := info1; tp2x := info2;
 18203:C        9         tp1y := info3; tp2y := info4;
***WARNING: (line18204): 'ADDR' of a constant may not be supported on other implementations
 18204:C        9         add_char_data ( gcbi, 2, addr('IN'));
 18205:C        9         hpgl_get_input_p1p2 ( gcbi );
 18206:C        9         { restore p1, p2 }
***WARNING: (line18207): 'ADDR' of a constant may not be supported on other implementations
 18207:C        9         add_char_data ( gcbi, 2, addr('IP'));
 18208:C        9         add_parm_data ( gcbi, tp1x );
***WARNING: (line18209): 'ADDR' of a constant may not be supported on other implementations
 18209:C        9         add_char_data ( gcbi, 1, addr(', '));
 18210:C        9         add_parm_data ( gcbi, tp1y );
***WARNING: (line18211): 'ADDR' of a constant may not be supported on other implementations
 18211:C        9         add_char_data ( gcbi, 1, addr(', '));
 18212:C        9         add_parm_data ( gcbi, tp2x );
***WARNING: (line18213): 'ADDR' of a constant may not be supported on other implementations
 18213:C        9         add_char_data ( gcbi, 1, addr(', '));
 18214:C        9         add_parm_data ( gcbi, tp2y );
 18215:C        9         hpgl_flush_buffer ( gcbi );
 18216:C        9       end;
 18217:C        2 end;
 18218:S
 18219:S
 18220:D        1 procedure hpgl_sample (  gcbi : graphics_input_control_block_ptr );
 18221:S
 18222:D        2 type
 18223:D        2   byte = 0..255;
 18224:S
 18225:D        2 var
 18226:D    -2  2   cnt    : gle_shortint;
 18227:D    -4  2   button : gle_shortint;
 18228:D    -4  2   status : packed record
 18229:D    -4  2              case byte of
 18230:D    -4  2                0 : (whole : gle_shortint);
 18231:D    -4  2                1 : (part  : packed record
 18232:D    -4  2                               bit15,bit14,bit13,bit12,bit11,
 18233:D    -4  2                               pen_down,
 18234:D    -4  2                               new_cursor,
```



```
 18235:D    -4  2                               proximity,
 18236:D    -4  2                               softkey,
 18237:D    -4  2                               srq,
 18238:D    -4  2                               error,
 18239:D    -4  2                               ready,
 18240:D    -4  2                               init,
 18241:D    -4  2                               point_ready,
 18242:D    -4  2                               bit1,
 18243:D    -4  2                               bit0 : boolean;
 18244:D    -4  2                             end);
 18245:D    -6  2            end;
 18246:S
 18247:C        2 begin
 18248:C        2   with gcbi^,
 18249:C        3   ascii_buffer_ptr(gcbi^.device_buf)^ do
 18250:C        3     begin
***WARNING: (line18251): 'ADDR' of a constant may not be supported on other implementations
 18251:C        3       add_char_data ( gcbi, 2, addr('OA') );
 18252:C        3       hpgl_flush_buffer ( gcbi );
 18253:C        3       call (io_read, iocb,  device_buf);
 18254:C        3       info1 := gle_read_integer (current,addr(data[1]),cnt);
 18255:C        3       info2 := gle_read_integer (current,addr(data[cnt+1]),cnt);
 18256:C        3       current := 0;
 18257:S
 18258:C        3       input_cpx := info1;
 18259:C        3       input_cpy := info2;
 18260:S
***WARNING: (line18261): 'ADDR' of a constant may not be supported on other implementations
 18261:C        3       add_char_data ( gcbi, 2, addr('OS') );
 18262:C        3       hpgl_flush_buffer ( gcbi );
 18263:C        3       call (io_read, iocb,  device_buf);
 18264:S
 18265:C        3       status.whole := gle_read_integer (current,addr(data[1]),cnt);
 18266:C        3       current := 0;
 18267:S
 18268:C        3       if status.part.point_ready then
 18269:C        4         info3 := -1
 18270:C        4       else
 18271:C        4       if status.part.pen_down then
 18272:C        5         info3 := 1
 18273:C        5       else
 18274:C        5         info3 := 0;
 18275:C        3     end;
 18276:C        2 end;
 18277:S
 18278:D        1 procedure hpgl_start_digitize (  gcbi : graphics_input_control_block_ptr );
 18279:S
 18280:C        2 begin
 18281:C        2   with gcbi^ do
 18282:C        3     begin
***WARNING: (line18283): 'ADDR' of a constant may not be supported on other implementations
 18283:C        3       add_char_data ( gcbi, 8, addr('PU;SG;DP'));
 18284:C        3       hpgl_flush_buffer ( gcbi );
 18285:C        3     end;
 18286:C        2 end;
 18287:S
 18288:D        1 procedure hpgl_get_digitize (  gcbi : graphics_input_control_block_ptr );
 18289:S
 18290:D        2 var
 18291:D    -4  2   i,cnt    : gle_shortint;
```

```
 18292:S
 18293:C        2 begin
 18294:C        2   with gcbi^,
 18295:C        3   ascii_buffer_ptr(gcbi^.device_buf)^ do
 18296:C        3     begin
***WARNING: (line18297): 'ADDR' of a constant may not be supported on other implementations
 18297:C        3       add_char_data ( gcbi, 2, addr('0D') );
 18298:C        3       hpgl_flush_buffer ( gcbi );
 18299:C        3       call (io_read, iocb,  device_buf);
 18300:C        3       tcnt := I;
 18301:C        3       info1 := gle_read_integer (current,addr(data[1]),cnt);
 18302:C        3       cnt := cnt + tcnt;
 18303:C        3       info2 := gle_read_integer (current,addr(data[cnt]),tcnt);
 18304:C        3       cnt := cnt + tcnt;
 18305:C        3       info3 := gle_read_integer (current,addr(data[cnt]),tcnt);
 18306:C        3       current := 0;
***WARNING: (line18307): 'ADDR' of a constant may not be supported on other implementations
 18307:C        3       add_char_data ( gcbi, 2, addr('DC') );
 18308:C        3       hpgl_flush_buffer ( gcbi );
 18309:C        3     end;
 18310:C        2 end;
 18311:S
 18312:D        1 procedure hpgl_input_echo (  gcbi : graphics_input_control_block_ptr );
 18313:S
 18314:C        2 begin
 18315:C        2   with gcbi^ do
 18316:C        3     begin
 18317:C        3       if info1 <> 0 then
 18318:C        4         begin
***WARNING: (line18319): 'ADDR' of a constant may not be supported on other implementations
 18319:C        4           add_char_data ( gcbi, 2, addr('BP'));
 18320:C        4           hpgl_flush_buffer ( gcbi );
 18321:C        4         end;
 18322:C        3     end;
 18323:C        2 end;
 18324:S
 18325:D        1 procedure gle_init_hpgl_input (  gcbi : graphics_input_control_block_ptr);
 18326:S
 18327:D        2 var
 18328:D    -4  2   saved_timeout : integer;
 18329:D    -8  2   i : integer;
 18330:S
 18331:C        2 begin
 18332:C        2   with gcbi^, ascii_buffer_ptr(device_buf)^ do
 18333:C        3     begin
 18334:C        3       try
 18335:C        4         maximum := max_buffer;
 18336:C        4         current := 0;
 18337:C        4         error_return := 0;
 18338:S
 18339:C        4         call (io_inq_timeout, iocb, saved_timeout );
 18340:C        4         call (io_set_timeout, iocb, 500 { ms } );
 18341:C        4         { send command that all HPGL plotters can respond to   }
 18342:C        4         { if the command fails then the address does not match }
 18343:C        4         { the device.                                          }
***WARNING: (line18344): 'ADDR' of a constant may not be supported on other implementations
 18344:C        4         add_char_data ( gcbi, 2, addr('OE') );
 18345:C        4         hpgl_flush_buffer ( gcbi );
 18346:C        4         call (io_read, iocb,  device_buf);
 18347:C        4         current := 0;
```



```
 18348:S
 18349:C        4         { if this point is reached then a vaild HPGL device was found }
 18350:S
 18351:C        4         try  { perform an output identify seq.  Note a 9872A will fail }
***WARNING: (line18352): 'ADDR' of a constant may not be supported on other implementations
 18352:C        5           add_char_data ( gcbi, 10, addr('IM30;DC;OI') );
 18353:C        5           hpgl_flush_buffer ( gcbi );
 18354:C        5           call (io_read, iocb,  device_buf);
 18355:S
 18356:C        5           for i := 1 to current do input_name[i] := data[i];
 18357:C        5           for i := current+1 to 6 do input_name[i] := ' ';
 18358:C        5           input_name_char_count := current;
 18359:C        5           current := 0;
 18360:C        5         recover
 18361:C        5           if escapecode = -26 then
 18362:C        6             begin
 18363:C        6               input_name := '9872A ';
 18364:C        6               input_name_char_count := 5;
 18365:C        6               current := 0;
 18366:C        6             end
 18367:C        6           else
 18368:C        6             escape(escapecode);
 18369:S
 18370:C        4         sample         := hpgl_sample;
 18371:C        4         start_digitize := hpgl_start_digitize;
 18372:C        4         get_digitize   := hpgl_get_digitize;
 18373:C        4         inq_p1p2       := hpgl_get_input_p1p2;
 18374:C        4         input_echo     := hpgl_input_echo;
 18375:C        4         input_escapei  := hpgl_input_escapei;
 18376:C        4         input_escapeo  := hpgl_input_escapeo;
 18377:S
 18378:C        4         input_res_x := 40;
 18379:C        4         input_res_y := 40;
 18380:S
 18381:C        4         input_handler_name := 'HPGL  ';
 18382:C        4         input_handler_char_count := 4;
 18383:S
 18384:C        4         hpgl_get_input_hard_clip ( gcbi );
 18385:C        4         input_min_x := info1;
 18386:C        4         input_max_x := info2;
 18387:C        4         input_min_y := info3;
 18388:C        4         input_max_y := info4;
 18389:S
 18390:C        4         input_cpx := input_min_x;
 18391:C        4         input_cpy := input_min_y;
 18392:S
 18393:C        4       recover
 18394:C        4         if escapecode = -26 then error_return := 1
 18395:C        5         else escape(escapecode); { ignor io errors }
 18396:S
 18397:C        3     try
 18398:C        4       call (io_set_timeout, iocb, saved_timeout );
 18399:C        4     recover
 18400:C        4       if escapecode = -26 then error_return := 1
 18401:C        5       else escape(escapecode); { ignor io errors }
 18402:C        3   end;
 18403:C        2 end;
 18404:S
 18405:S
 18406:C        1 end. { hpgl_input }
```

```
 18407:S
No errors. 25 warnings.
             ***** Nonstandard language features enabled *****
```

GLE_HPIB

Description

GLE_HPIB contains I/O routines which interface the graphics library to the system HP-IB I/O drivers.

Usage

Used with HPGL displays or locators connected via HP-IB.

Requirements

GLE_TYPES, GENERAL_0, IODECLARATIONS, SYSGLOBALS, IOCOMASM, and GLE_UTLS.

```
Pascal [Rev 3.0M  6/ 4/84] GLE_HPIB.TEXT          26-Dec-84 22:36:48 Page 1   *** File name: GLE_HPIB ***

     1:S
     2:D         0 {                                                                    }
     3:D         0 { Graphics Low End                                                   }
     4:D         0 {                                                                    }
     5:D         0 { Module    = GLE_HPIB                                               }
     6:D         0 { Programer = BJS                                                    }
     7:D         0 { Date      = 10-10-82                                               }
     8:D         0 {                                                                    }
     9:D         0 { Purpose: To provide IO routines for ascii device handlers.         }
    10:S
    11:D         0 { Rev history                                                        }
    12:D         0 {  Created  - 10-10-82                                               }
    13:D         0 {  Modified - 12-12-83 BDS -- Brought needed general_1 and hpib_1    }
    14:D         0 {                             routines in-line.                      }
    15:S
    16:S           {     (c) Copyright Hewlett-Packard Company, 1983.
    17:S                 All rights are reserved.  Copying or other
    18:S                 reproduction of this program except for archival
    19:S                 purposes is prohibited without the prior
    20:S                 written consent of Hewlett-Packard Company.
    21:S
    22:S
    23:S                             RESTRICTED RIGHTS LEGEND
    24:S
    25:S                 Use, duplication, or disclosure by the Government
    26:S                 is subject to restrictions as set forth in
    27:S                 paragraph (b) (3) (B) of the Rights in Technical
    28:S                 Data and Computer Software clause in
    29:S                 DAR 7-104.9(a).
    30:S
    31:S                 HEWLETT-PACKARD COMPANY
    32:D         0       Fort Collins, Colorado                                 }
    33:S
    34:D         0 $SEARCH 'GLE_TYPES',
    35:D         0         'GLE_UTLS'$
    36:D         0 $model$
    36:D         0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS *************** }
    37:S
    38:S            { This include file specifies range checking, debug and other compiler
    39:D         0    options for the graphics library                                  }
    40:S
    41:D         0 $debug OFF$
    42:D         0 $range OFF$
    43:D         0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    44:D         0 $FLOAT_HDW TEST$
    45:S
    46:S
    47:S
    48:S
    49:D         0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS *************** }
 10000:D         0 $LINENUM 10000$
 10001:S
 10002:D         0 module gle_hpib_io;
 10003:S
 10004:D         1 import gle_types,
 10005:D         1        general_0,
 10006:D         1        iocomasm;
 10007:S
 10008:D         1 export
 10009:D         1 type
```

```
Pascal [Rev 3.0M  6/ 4/84] GLE_HPIB.TEXT          26-Dec-84 22:36:48 Page 2   *** File name: GLE_HPIB ***

 10010:D         1    hpib_iocb_ptr = ^ hpib_iocb;
 10011:D         1    hpib_iocb = record
 10012:D         1                  device_addr : anyptr;
 10013:D         1                  name_size   : gle_shortint;
 10014:D         1                  address     : integer;
 10015:D         1                  select_code : integer;
 10016:D         1                  error       : integer;
 10017:D         1                end;
 10018:D         1    timeoutrec = record                          {tttt JS 8/3/83}
 10019:D         1                  counter: integer;              {tttt JS 8/3/83}
 10020:D         1                  firsttime: boolean;            {tttt JS 8/3/83}
 10021:D         1                end;                             {tttt JS 8/3/83}
 10022:S
 10023:D         1 procedure hpib_init  ( anyvar iocb_ptr : anyptr );
 10024:D         1 procedure hpib_inq_timeout ( anyvar iocb_ptr : anyptr; var value : integer );
 10025:D         1 procedure hpib_set_timeout ( anyvar iocb_ptr : anyptr; value : integer );
 10026:D         1 procedure hpib_write ( anyvar iocb_ptr, data_ptr : anyptr );
 10027:D         1 procedure hpib_read  ( anyvar iocb_ptr, data_ptr : anyptr );
 10028:D         1 procedure hpib_term  ( anyvar iocb_ptr : anyptr );
 10029:S
 10030:D         1 implement
 10031:S
 10032:D         1 import
 10033:D         1    iodeclarations,
 10034:D         1    (general_1,)
 10035:D         1    (hpib_1,)
 10036:D         1    gle_utls;
 10037:S
 10038:D         1 { The following types must match types declared in GLE_HPGL, and GLE_HPGLI }
 10039:D         1 type
 10040:D         1   ascii_buffer_ptr = ^ascii_buffer;
 10041:S
 10042:D         1   ascii_buffer = packed record
 10043:D         1                     maximum : integer;
 10044:D         1                     current : integer;
 10045:D         1                     data    : packed array [1..32767] of char;
 10046:D         1                   end;
 10047:S
 10048:D         1 {|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||}
 10049:S
 10050:D         1                          {*** general_1 ***}
 10051:S
 10052:S           {The following procedures have been brought in-line to avoid importing
 10053:S            general_1 and hpib_1. NOTE: These routines must be duplicates of those
 10054:S            found in general_1 and hpib_1 and therefore must reflect any modifications
 10055:D         1  to those modules.                                                    }
 10056:S
 10057:S
 10058:S
 10059:D         1   FUNCTION timerexists: boolean;  external;          { tttt JS 8/3/83 }
 10060:S
 10061:D         1   FUNCTION timed_out(var rec: timeoutrec): boolean; external; {tttt JS 8/3/83}
 10062:S
 10063:S
 10064:D         1   PROCEDURE ioreset     ( select_code : type_isc);
 10065:C         2   BEGIN
 10066:C         2     WITH isc_table[select_code] DO
 10067:C         3     CALL(io_drv_ptr^.iod_init,
 10068:C         3          io_tmp_ptr);
 10069:C         2   END; { of ioreset }
```

```
10070:S
10071:S
10072:D        1   PROCEDURE writechar   ( select_code : type_isc ;
10073:D        2                           value       : CHAR );
10074:C        2   BEGIN
10075:C        2     WITH isc_table[select_code] DO
10076:C        3     CALL(io_drv_ptr^.iod_wtb,
10077:C        3          io_tmp_ptr,
10078:C        3          value);
10079:C        2   END; { of writechar }
10080:S
10081:D        1   PROCEDURE set_timeout ( select_code : type_isc ;
10082:D    -8  2                           time        : REAL          { in seconds ) );
10083:C        2   BEGIN
10084:C        2     IF time>8191                { 4 byte timeout  - 1 byte left for shifts }
10085:C        3       THEN BEGIN
10086:C        3         { error }
10087:C        3         io_escape(ioe_bad_tmo,select_code);
10088:C        3       END; { of IF }
10089:C        2     IF (time>0) AND (time<0.001)
10090:C        3       THEN BEGIN
10091:C        3         { error }
10092:C        3         io_escape(ioe_bad_tmo,select_code);
10093:C        3       END; { of IF }
10094:S
10095:C        2     WITH isc_table[select_code] DO BEGIN
10096:S
10097:C        3       { the table entry used by drivers is in milliseconds }
10098:C        3       user_time:=ROUND(time*1000);
10099:S
10100:C        3       IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout := user_time;
10101:S
10102:C        3     END; { of WITH DO BEGIN }
10103:S
10104:C        2   END; { of set_timeout }
10105:D        1
10106:D        1                           {*** hpib_1 ***}
10107:S
10108:D        1   PROCEDURE send_command( select_code : type_isc ;
10109:D        2                           command     : CHAR );
10110:C        2   BEGIN
10111:C        2     WITH isc_table[select_code] DO
10112:C        3       CALL (io_drv_ptr^.iod_send,
10113:C        3             io_tmp_ptr,
10114:C        3             command);
10115:C        2   END; { of send_command }
10116:S
10117:S
10118:S
10119:D        1   FUNCTION  my_address  ( select_code : type_isc)
10120:D        2           : type_hpib_addr ;
10121:C        2   BEGIN
10122:C        2     IF isc_table[select_code].io_tmp_ptr <> NIL
10123:C        3       THEN BEGIN
10124:C        3         WITH isc_table[select_code].io_tmp_ptr^ DO
10125:C        4           IF addressed <> -1
10126:C        5             THEN BEGIN
10127:C        5               my_address:=addressed;
10128:C        5             END
10129:C        5             ELSE BEGIN
```



```
10130:C        5               { error }
10131:C        5               io_escape(ioe_not_hpib,select_code);
10132:C        5             END; { of IF addressed }
10133:C        3       END
10134:C        3       ELSE BEGIN
10135:C        3         { error }
10136:C        3         io_escape(ioe_not_hpib,select_code);
10137:C        3       END; { of IF io_tmp_ptr }
10138:C        2   END; { of my_address }
10139:S
10140:S
10141:S
10142:D        1   FUNCTION  active_controller
10143:D        2                           ( select_code : type_isc)
10144:D        2                           : BOOLEAN;
10145:C        2   BEGIN
10146:C        2     IF isc_table[select_code].card_type=hpib_card
10147:C        3       THEN BEGIN
10148:C        3         active_controller:=bit_set(iostatus(select_code,3),6);
10149:C        3       END
10150:C        3       ELSE BEGIN
10151:C        3         active_controller := TRUE;
10152:C        3       END; { of IF }
10153:C        2   END; { of active_controller }
10154:S
10155:S
10156:S
10157:D        1   {************************* this function is not used  ********************}
10158:S            {FUNCTION  system_controller
10159:S                                    ( select_code : type_isc)
10160:S                                    : BOOLEAN;
10161:S            BEGIN
10162:S              IF isc_table[select_code].card_type=hpib_card
10163:S                THEN BEGIN
10164:S                  system_controller:=bit_set(iostatus(select_code,3),7);
10165:S                END
10166:S                ELSE BEGIN
10167:S                  system_controller := TRUE;
10168:S                END;  of IF
10169:D        1   END;  of system_controller }
10170:S
10171:S
10172:S
10173:D        1   {************************* this functio is not used ********************}
10174:S            {FUNCTION  end_set      ( select_code : type_isc )
10175:S                                    : BOOLEAN ;
10176:S            VAR mybool : BOOLEAN;
10177:S            BEGIN
10178:S              WITH isc_table[select_code] DO
10179:S                CALL (io_drv_ptr^.iod_end,
10180:S                      io_tmp_ptr,
10181:S                      mybool);
10182:S              end_set := mybool;
10183:D        1   END;  of send_command }
10184:S
10185:S
```

```
10186:D        1   $PAGE$
10187:S
10188:D        1   FUNCTION  addr_to_talk( device      : type_device)
10189:D        2                          : type_isc;
10190:D    -2  2   VAR io_isc : type_isc;
10191:D    -6  2       timer  : INTEGER;
10192:D   -12  2       hpibrec: timeoutrec;                                    {tttt JS 8/3/83}
10193:S
10194:C        2   BEGIN
10195:S
10196:C        2     IF device>iomaxisc
10197:C        3       THEN BEGIN
10198:C        3         io_isc:=device DIV 100;
10199:S
10200:C        3         WITH isc_table[io_isc] DO BEGIN
10201:S
10202:C        4           IF io_tmp_ptr <> NIL
10203:C        5             THEN BEGIN
10204:S
10205:C        5               { set up user timeout - in case system drivers changed it }
10206:C        5               io_tmp_ptr^.timeout:=user_time;
10207:S
10208:C        5               IF io_tmp_ptr^.addressed <> -1
10209:C        6                 THEN BEGIN
10210:C        6                   IF ( card_type <> hpib_card ) AND
10211:C        7                      ( device MOD 100 > 31 )
10212:C        7                     THEN io_escape(ioe_misc,io_isc);
10213:C        6                   send_command(io_isc,CHR(talk_constant+(device MOD 100)));
10214:C        6                   send_command(io_isc,'?');
10215:C        6                   send_command(io_isc,CHR(my_address(io_isc)+listen_constant));
10216:C        6                 END
10217:C        6                 ELSE BEGIN
10218:C        6                   { error }
10219:C        6                   io_escape(ioe_not_hpib,io_isc);
10220:C        6                 END; { of IF }
10221:C        5             END
10222:C        5             ELSE BEGIN
10223:C        5             END; { of IF }
10224:C        4           END; { of WITH DO BEGIN }
10225:C        3       END
10226:C        3       ELSE BEGIN
10227:C        3         io_isc:=device;
10228:C        3
10229:C        3         WITH isc_table[io_isc] DO BEGIN
10230:S
10231:C        4           { set up user timeout - in case system drivers changed it }
10232:C        4           IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout:=user_time;
10233:S
10234:C        4           IF card_type=hpib_card THEN
10235:C        5             BEGIN
10236:C        5               IF NOT active_controller(io_isc)
10237:C        6                 THEN BEGIN
10238:C        6                   { if non controller wait until listener }
10239:C        6                   IF user_time = 0
10240:C        7                     THEN BEGIN
10241:C        7                       REPEAT
10242:C        8                         { wait forever }
10243:C        8                       UNTIL bit_set(iostatus(io_isc,6),10);
10244:C        7                     END
10245:C        7                     ELSE BEGIN
```



```
10246:C        7                       { wait for timeout value }
10247:C        7                       IF timerexists THEN BEGIN                {tttt JS 8/3/83}
10248:C        8                         hpibrec.firsttime:=true;               {tttt JS 8/3/83}
10249:C        8                         hpibrec.counter:=user_time;            {tttt JS 8/3/83}
10250:C        8                         REPEAT                                 {tttt JS 8/3/83}
10251:C        9                         UNTIL timed_out(hpibrec) OR            {tttt JS 8/3/83}
10252:C        9                           bit_set(iostatus(io_isc,6),10);      {tttt JS 8/3/83}
10253:C        8                       END                                      {tttt JS 8/3/83}
10254:C        8                       ELSE BEGIN                               {tttt JS 8/3/83}
10255:C        8                         timer:=user_time*3;
10256:C        8                         REPEAT
10257:C        9                           timer:=timer-1;
10258:C        9                         UNTIL ( timer = 0 ) OR
10259:C        9                               ( bit_set(iostatus(io_isc,6),10) ) ;
10260:C        8                       END;                                     {tttt JS 8/3/83}
10261:C        7                       IF NOT bit_set(iostatus(io_isc,6),10)
10262:C        8                         THEN io_escape(ioe_timeout,io_isc);
10263:C        7                     END; { of IF user_time=0 }
10264:C        6                 END; { of IF }
10265:C        5             END; { of IF card_type = hpib_card }
10266:C        4           END; { of WITH DO BEGIN }
10267:C        3       END; { of IF }
10268:S
10269:C        2     addr_to_talk:=io_isc; { return select code }
10270:S
10271:C        2   END; { of addr_to_talk }
10272:S
```

```
10273:D        1   $PAGE$
10274:S
10275:D        1   FUNCTION  addr_to_listen
10276:D        2                        ( device      : type_device)
10277:D        2                        : type_isc;
10278:D    -2  2   VAR io_isc : type_isc;
10279:D    -6  2       timer  : INTEGER;
10280:D   -12  2       hpibrec: timeoutrec;                                    {tttt JS 8/3/83}
10281:S
10282:C        2   BEGIN
10283:S
10284:C        2     IF device>iomaxisc
10285:C        3       THEN BEGIN
10286:C        3         io_isc:=device DIV 100;
10287:S
10288:C        3         WITH isc_table[io_isc] DO BEGIN
10289:S
10290:C        4           IF io_tmp_ptr <> NIL
10291:C        5             THEN BEGIN
10292:S
10293:C        5               { set up user timeout - in case system drivers changed it }
10294:C        5               io_tmp_ptr^.timeout:=user_time;
10295:S
10296:C        5               IF io_tmp_ptr^.addressed <> -1
10297:C        6                 THEN BEGIN
10298:C        6                   IF ( card_type <> hpib_card ) AND
10299:C        7                      ( device MOD 100 > 31 )
10300:C        7                     THEN io_escape(ioe_misc,io_isc);
10301:C        6                   send_command(io_isc,CHR(my_address(io_isc)+talk_constant));
10302:C        6                   send_command(io_isc,'?');
10303:C        6                   send_command(io_isc,CHR(listen_constant+(device MOD 100)));
10304:C        6                 END
10305:C        6                 ELSE BEGIN
10306:C        6                   { error }
10307:C        6                   io_escape(ioe_not_hpib,io_isc);
10308:C        6                 END; { of IF }
10309:C        5             END
10310:C        5             ELSE BEGIN
10311:C        5             END; { of IF }
10312:C        4         END; { of WITH DO BEGIN }
10313:C        3       END
10314:C        3       ELSE BEGIN
10315:C        3         io_isc:=device;
10316:C        3
10317:C        3         WITH isc_table[io_isc] DO BEGIN
10318:S
10319:C        4           { set up user timeout - in case system drivers changed it }
10320:C        4           IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout:=user_time;
10321:S
10322:C        4           IF card_type=hpib_card THEN
10323:C        5             BEGIN
10324:C        5               IF NOT active_controller(io_isc)
10325:C        6                 THEN BEGIN
10326:C        6                   { if non controller wait until talker }
10327:C        6                   IF user_time = 0
10328:C        7                     THEN BEGIN
10329:C        7                       REPEAT
10330:C        8                         { wait forever }
10331:C        8                       UNTIL bit_set(iostatus(io_isc,6),9);
10332:C        7                     END
```



```
10333:C        7                     ELSE BEGIN
10334:C        7                       { wait for timeout value }
10335:C        7                       IF timerexists THEN BEGIN              {tttt JS 8/3/83}
10336:C        8                         hpibrec.firsttime:=true;              {tttt JS 8/3/83}
10337:C        8                         hpibrec.counter:=user_time;           {tttt JS 8/3/83}
10338:C        8                         REPEAT                                {tttt JS 8/3/83}
10339:C        9                         UNTIL timed_out(hpibrec) OR           {tttt JS 8/3/83}
10340:C        9                           bit_set(iostatus(io_isc,6), 9);     {tttt JS 8/3/83}
10341:C        8                       END                                     {tttt JS 8/3/83}
10342:C        8                       ELSE BEGIN                              {tttt JS 8/3/83}
10343:C        8                         timer:=user_time*3;
10344:C        8                         REPEAT
10345:C        9                           timer:=timer-1;
10346:C        9                         UNTIL ( timer = 0 ) OR
10347:C        9                               ( bit_set(iostatus(io_isc,6),9) ) ;
10348:C        8                       END;                                    {tttt JS 8/3/83}
10349:C        7                       IF NOT bit_set(iostatus(io_isc,6),9)
10350:C        8                         THEN io_escape(ioe_timeout,io_isc);
10351:C        7                     END; { of IF user_time=0 }
10352:C        6                 END; { of IF }
10353:C        5             END; { of IF card_type = hpib_card }
10354:C        4         END; { of WITH DO BEGIN }
10355:C        3       END; { of IF }
10356:S
10357:C        2     addr_to_listen:=io_isc;
10358:S
10359:C        2   END; { of addr_to_listen }
10360:S
```

```
10361:D       1   $PAGE$
10362:S
10363:S
10364:S           { set to talk exists because of HPIB_2/HPIB_3 -
10365:S             those routines are intended to be the controller
10366:S             ( active ) and should not wait for the card to be
10367:S             addressed as talker.  addr_to_talk is used by
10368:S             data transfer routines.  set_to_talk is used by
10369:D       1     bus control routines.                          }
10370:S
10371:S
10372:S
10373:S
10374:D       1   {************************ this function is never used ******************}
10375:S           {FUNCTION  set_to_talk ( device      : type_device)
10376:S                                  : type_isc;
10377:S           VAR io_isc : type_isc;
10378:S           BEGIN
10379:S
10380:S             IF device>iomaxisc
10381:S               THEN BEGIN
10382:S                 io_isc:=addr_to_talk(device);
10383:S               END
10384:S               ELSE BEGIN
10385:S                 io_isc:=device;
10386:S
10387:S                 WITH isc_table[io_isc] DO BEGIN
10388:S
10389:S                    set up user timeout - in case system drivers changed it
10390:S                   IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout:=user_time;
10391:S
10392:S                   IF card_type=hpib_card THEN
10393:S                     BEGIN
10394:S                       IF NOT active_controller(io_isc)
10395:S                         THEN BEGIN
10396:S
10397:S                           io_escape(ioe_not_act,io_isc);
10398:S
10399:S                         END;  of IF
10400:S                     END;  of IF card_type = hpib_card
10401:S                 END;  of WITH DO BEGIN
10402:S               END;  of IF
10403:S
10404:S             set_to_talk:=io_isc;  return select code
10405:S
10406:D       1   END;  of set_to_talk }
10407:S
```



```
10408:D       1   $PAGE$
10409:S
10410:S
10411:S           { set to listen exists because of HPIB_2/HPIB_3 -
10412:S             those routines are intended to be the controller
10413:S             ( active ) and should not wait for the card to be
10414:S             addressed as listener.  addr_to_listen is used by
10415:S             data transfer routines.  set_to_listen is used by
10416:D       1     bus control routines.                          }
10417:S
10418:S
10419:S
10420:S
10421:S
10422:D       1   {******************* this function is never used **********************}
10423:S           {FUNCTION  set_to_listen
10424:S                                ( device      : type_device)
10425:S                                : type_isc;
10426:S           VAR io_isc : type_isc;
10427:S               timer  : INTEGER;
10428:S           BEGIN
10429:S
10430:S             IF device>iomaxisc
10431:S               THEN BEGIN
10432:S                 io_isc:=addr_to_listen(device);
10433:S
10434:S               END
10435:S               ELSE BEGIN
10436:S                 io_isc:=device;
10437:S
10438:S                 WITH isc_table[io_isc] DO BEGIN
10439:S
10440:S                    set up user timeout - in case system drivers changed it
10441:S                   IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout:=user_time;
10442:S
10443:S                   IF card_type=hpib_card THEN
10444:S                     BEGIN
10445:S                       IF NOT active_controller(io_isc)
10446:S                         THEN BEGIN
10447:S
10448:S                           io_escape(ioe_not_act,io_isc);
10449:S
10450:S                         END;  of IF
10451:S                     END;  of IF card_type = hpib_card
10452:S                 END;  of WITH DO BEGIN
10453:S               END;  of IF
10454:S
10455:S             set_to_listen:=io_isc;
10456:S
10457:D       1   END;  of set_to_listen }
10458:S
10459:S
10460:S
10461:S
10462:S
10463:S
10464:D       1 {|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||}
10465:S
10466:S
10467:D       1 procedure hpib_inq_timeout ( anyvar iocb_ptr : anyptr; var value : integer );
```

```
10468:S
10469:C         2 begin
10470:C         2   with hpib_iocb_ptr(iocb_ptr)^ do
10471:C         3     begin
10472:C         3       value := isc_table[select_code].user_time;
10473:C         3     end;
10474:C         2 end;
10475:S
10476:D         1 procedure hpib_set_timeout ( anyvar iocb_ptr : anyptr; value : integer );
10477:S
10478:C         2 begin
10479:C         2   with hpib_iocb_ptr(iocb_ptr)^ do
10480:C         3     set_timeout (select_code,value/1000);
10481:C         2 end;
10482:S
10483:D         1 procedure hpib_init  ( anyvar iocb_ptr : anyptr );
10484:S
10485:D         2 var
10486:D    -2   2   cnt : gle_shortint;
10487:S
10488:C         2 begin
10489:C         2   with hpib_iocb_ptr(iocb_ptr)^ do
10490:C         3     begin
10491:C         3       error := 1;
10492:C         3       try
10493:C         4         address := gle_read_integer ( name_size, device_addr, cnt );
10494:C         4         select_code := address div 100;
10495:C         4         if (select_code >= minrealisc) and
10496:C         5            (select_code <= maxrealisc) then
10497:C         5            begin
10498:C         5              ioreset (select_code);
10499:C         5              error := 0;
10500:C         5            end;
10501:C         4       recover
10502:C         4         { error is set, ignore range and io escapes }
10503:C         4         if (escapecode <> -8) and (escapecode <> -26) then escape(escapecode);
10504:C         3     end;
10505:C         2 end;
10506:S
10507:D         1 procedure hpib_write ( anyvar iocb_ptr, data_ptr : anyptr );
10508:S
10509:D         2 var
10510:D    -4   2   i : integer;
10511:D    -6   2   io_isc : type_isc;
10512:S
10513:C         2 begin
10514:C         2   with hpib_iocb_ptr(iocb_ptr)^,ascii_buffer_ptr(data_ptr)^ do
10515:C         3     begin
10516:C         3       io_isc := addr_to_listen(address);
10517:C         3       with isc_table[io_isc].io_drv_ptr^, isc_table[io_isc] do
10518:C         4         begin
10519:C         4           for i := 1 to current do
10520:C         5             call (iod_wtb, io_tmp_ptr, data[i] );
10521:C         4           writechar(io_isc,io_carriage_rtn);
10522:C         4           writechar(io_isc,io_line_feed);
10523:C         4         end;
10524:C         3       current := 0;
10525:C         3     end;
10526:C         2 end;
10527:S
```



```
10528:D         1 procedure hpib_read  ( anyvar iocb_ptr, data_ptr : anyptr );
10529:S
10530:D         2 var
10531:D    -4   2   i : integer;
10532:D    -6   2   io_isc : type_isc;
10533:S
10534:C         2 begin
10535:C         2   with hpib_iocb_ptr(iocb_ptr)^,ascii_buffer_ptr(data_ptr)^ do
10536:C         3     begin
10537:C         3       io_isc := addr_to_talk ( address );
10538:C         3       with isc_table[io_isc].io_drv_ptr^, isc_table[io_isc] do
10539:C         4         begin
10540:C         4           i := 0;
10541:C         4           repeat
10542:C         5             i := i + 1;
10543:C         5             call (iod_rdb, io_tmp_ptr, data[i]);
10544:C         5           until ( ( i >= maximum ) or ( data[i] = io_line_feed ) );
10545:C         4           if data[i] = io_line_feed then i := i - 1;
10546:C         4           if i <> 0 then if data[i] = io_carriage_rtn then i := i - 1;
10547:C         4           current := i;
10548:C         4         end;
10549:C         3     end;
10550:C         2 end;
10551:S
10552:D         1 procedure hpib_term  ( anyvar iocb_ptr : anyptr );
10553:S
10554:C         2 begin
10555:C         2 end;
10556:S
10557:C         1 end. { of module gle_hpib_io }
10558:D         1
10559:D         1 $LIST ON$

No errors. No warnings.
          ***** Nonstandard language features enabled *****
```

GLE_KNOB

Description

GLE_KNOB provides a device handler for the built-in knob input device.

Requirements

GLE_TYPES, GLE_UTLS, and SYSGLOBALS.

```
     1:S
     2:D        0 {                                                                    }
     3:D        0 { Graphics Low End                                                   }
     4:D        0 {                                                                    }
     5:D        0 { Module    = GLE_KNOB_IN                                            }
     6:D        0 { Programer = BJS                                                    }
     7:D        0 { Date      = 11-05-82                                               }
     8:D        0 {                                                                    }
     9:D        0 { Purpose: To provide a device handler for the knob input device.    }
    10:S
    11:D        0 { Rev history                                                        }
    12:D        0 {  Created  - 11- 5-82                                               }
    13:D        0 {  Modified -  4-11-84 BY JWS -- remove unitio dependecies           }
    14:S
    15:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    16:S                All rights are reserved.  Copying or other
    17:S                reproduction of this program except for archival
    18:S                purposes is prohibited without the prior
    19:S                written consent of Hewlett-Packard Company.
    20:S
    21:S
    22:S                            RESTRICTED RIGHTS LEGEND
    23:S
    24:S                Use, duplication, or disclosure by the Government
    25:S                is subject to restrictions as set forth in
    26:S                paragraph (b) (3) (B) of the Rights in Technical
    27:S                Data and Computer Software clause in
    28:S                DAR 7-104.9(a).
    29:S
    30:S                HEWLETT-PACKARD COMPANY
    31:D        0       Fort Collins, Colorado                                 }
    32:S
    33:S
    34:D        0 $UCSD$
    35:D        0 $SEARCH 'GLE_TYPES','GLE_UTLS'$
    36:D        0 $modcal$
    36:D        0 $INCLUDE 'OPTIONS'$  { ***************** COMPILER OPTIONS ***************** }
    37:S
    38:S          { This include file specifies range checking, debug and other compiler
    39:D        0   options for the graphics library                                   }
    40:S
    41:D        0 $debug OFF$
    42:D        0 $range OFF$
    43:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    44:D        0 $FLOAT_HDW TEST$
    45:S
    46:S
    47:S
    48:S
    49:D        0 $INCLUDE 'OPTIONS'$  { ***************** COMPILER OPTIONS ***************** }
 18000:D        0 $LINENUM 18000$
 18001:S
 18002:D        0 module gle_knob_in;
 18003:S
 18004:D        1 import gle_types;
 18005:S
 18006:D        1 export
 18007:D        1
 18008:D        1 type
 18009:D        1   knob_device_rec_ptr = ^knob_device_rec;
```



```
 18010:D        1   knob_device_rec = packed record
 18011:D        1     knob_type : gle_shortint;       { 0 - unknown device }
 18012:D        1                                     { 1 - 9816           }
 18013:D        1                                     { 2 - 9826           }
 18014:D        1                                     { 3 - 9836           }
 18015:D        1                                     { 4 - 9836c          }
 18016:D        1     echo_rate : gle_shortint;
 18017:D        1     echo_mult : gle_shortint;
 18018:D        1     last_but  : integer;
 18019:D        1     digitize_enabled : boolean;
 18020:D        1     last_x    : integer;
 18021:D        1     last_y    : integer;
 18022:D        1   end;
 18023:S
 18024:D        1 procedure gle_init_knob_input ( gcbi : graphics_input_control_block_ptr );
 18025:D        1
 18026:D        1 implement
 18027:S
 18028:D        1 import gle_utls,sysglobals;
 18029:S
 18030:S
 18031:C        1  procedure eot(fp: fibp); begin end;                    {do nothing}
 18032:S
 18033:D        1 procedure openfib(anyvar F: fib; unum: unitnum);
 18034:C        2 begin
 18035:C        2 if (unum<=0) or (unum>maxunit) then ioresult := ord(ibadunit)
 18036:C        3 else with F do
 18037:C        4   begin
 18038:C        4   fistextvar := false;
 18039:C        4   funit := unum;
 18040:C        4   feot  := eot;
 18041:C        4   call(unitable^[unum].dam, f, unum, openunit);
 18042:C        4   end;
 18043:C        2 end;
 18044:S
 18045:D        1 procedure gunitread ( u: integer; buf: charptr; len: integer);
 18046:D  -270  2 var f:    file;
 18047:D  -272  2     r:    amrequesttype;
 18048:C        2 begin with unitable^[u] do
 18049:C        3   begin
 18050:C        3   openfib(f, u);
 18051:C        3   if ioresult = ord(inoerror) then
 18052:C        4     begin
 18053:C        4     r := readbytes;
 18054:C        4     call(tm, addr(f), r, buf^, len, 0);
 18055:C        4     end;
 18056:C        3   end;
 18057:C        2 end
 18058:S
 18059:D        1 function gunitbusy ( u: integer): boolean;
 18060:D  -270  2 var f:    file;
 18061:C        2 begin with unitable^[u] do
 18062:C        3   begin
 18063:C        3   gunitbusy := true;
 18064:C        3   openfib(f, u);
 18065:C        3   if ioresult = ord(inoerror) then
 18066:C        4     begin
 18067:C        4     call(tm, addr(f), unitstatus, f, 0, 0);
 18068:C        4     gunitbusy := fibp(addr(f))^.fbusy;
 18069:C        4     end;
```

```
18070:C        3   end;
18071:C        2 end;
18072:S
18073:S
18074:S
18075:D        1 procedure knob_dummy (  gcbi : graphics_input_control_block_ptr );
18076:S
18077:C        2 begin
18078:C        2 end;
18079:S
18080:D        1 procedure knob_get_input_p1p2 (  gcbi : graphics_input_control_block_ptr );
18081:S
18082:C        2 begin
18083:C        2   with gcbi^,knob_device_rec_ptr(dev_dep_stuff)^ do
18084:C        3     case knob_type of
18085:C        4       0 : begin                    { input_xxxx must be set by config }
18086:C        4             info1 := input_min_x;
18087:C        4             info2 := input_max_x;
18088:C        4             info3 := input_min_y;
18089:C        4             info4 := input_max_y;
18090:C        4           end;
18091:C        4     1,2 : begin
18092:C        4             info1 := 0;
18093:C        4             info2 := 399;
18094:C        4             info3 := 0;
18095:C        4             info4 := 299;
18096:C        4           end;
18097:C        4     3,4 : begin
18098:C        4             info1 := 0;
18099:C        4             info2 := 511;
18100:C        4             info3 := 0;
18101:C        4             info4 := 389;
18102:C        4           end;
18103:C        4     end; { case }
18104:C        2 end;
18105:S
18106:D        1 procedure knob_sample (  gcbi : graphics_input_control_block_ptr );
18107:S
18108:D        2 var
18109:D    -1  2   ch : char;
18110:D    -6  2   mycharptr: charptr;
18111:D    -6  2   x_adj,
18112:D    -6  2   y_adj,
18113:D   -18  2   rate : integer;
18114:D   -19  2   commandinprogress: char;
18115:D   -24  2   interruptlevel: integer;
18116:D   -24  2
18117:C        2 begin
18118:C        2   mycharptr:=addr(ch);
18119:C        2   with gcbi^,knob_device_rec_ptr(dev_dep_stuff)^ do
18120:C        3     begin
18121:C        3       if last_but < 0 then info3 := -1
18122:C        4       else                 info3 := 0;
18123:C        3       if gunitbusy(2) then
18124:C        4       { if no keys in type ahead buffer }
18125:C        4         begin
18126:C        4           { just return last position }
18127:C        4           info1 := input_cpx;
18128:C        4           info2 := input_cpy;
18129:C        4         end
```



```
18130:C        4       else
18131:C        4         begin
18132:C        4           gunitread(2,mycharptr,1);
18133:C        4           x_adj := 0;
18134:C        4           y_adj := 0;
18135:C        4           rate  := 0;
18136:C        4           case ord(ch) of
18137:C        5             8 : x_adj := -echo_rate;      { left arrow }
18138:C        5            28 : x_adj := +echo_rate;      { right arrow }
18139:C        5            10 : y_adj := -echo_rate;      { down arrow }
18140:C        5            31 : y_adj := +echo_rate;      { up arrow }
18141:C        5            49,33 : rate := 1;             { number 1 }
18142:C        5            50,64 : rate := 2;             {   ...  2 }
18143:C        5            51,35 : rate := 3;
18144:C        5            52,36 : rate := 4;
18145:C        5            53,37 : rate := 5;
18146:C        5            54,94 : rate := 6;
18147:C        5            55,38 : rate := 7;
18148:C        5            56,42 : rate := 8;
18149:C        5            57,40 : rate := 9;             { number 9 }
18150:C        5            otherwise
18151:C        5              begin
18152:C        5                if ch = chr(13) then ch := ' ';
18153:C        5                if digitize_enabled then
18154:C        6                  begin
18155:C        6                    if (last_but = 0) then
18156:C        7                      begin
18157:C        7                        last_but := -ord(ch);
18158:C        7                        info3  := -1;
18159:C        7                        last_x := input_cpx;
18160:C        7                        last_y := input_cpy;
18161:C        7                      end
18162:C        7                  end
18163:C        6                else
18164:C        6                  info3 := ord(ch);
18165:C        5              end;
18166:S
18167:C        5           end;  { of case }
18168:S
18169:C        4           { ck for new rate }
18170:C        4           if rate > 0 then
18171:C        5              echo_rate := (echo_mult * (rate-1) + 1);
18172:S
18173:C        4           { calc new x and y }
18174:C        4           input_cpx := gle_shortint_max(input_min_x,gle_shortint_min(input_max_x,input_cpx+x_adj));
18175:C        4           input_cpy := gle_shortint_max(input_min_y,gle_shortint_min(input_max_y,input_cpy+y_adj));
18176:S
18177:C        4           info1 := input_cpx;
18178:C        4           info2 := input_cpy;
18179:C        4         end;
18180:C        3     end;
18181:C        2 end;
18182:S
18183:D        1 procedure knob_start_digitize (  gcbi : graphics_input_control_block_ptr );
18184:S
18185:C        2 begin
18186:C        2   with gcbi^,knob_device_rec_ptr(dev_dep_stuff)^ do
18187:C        3     begin
18188:C        3       digitize_enabled := true;
18189:C        3       echo_mult := info2;
```

```
18190:C        3       last_but := 0;
18191:C        3     end;
18192:C        2 end;
18193:S
18194:D        1 procedure knob_get_digitize (  gcbi : graphics_input_control_block_ptr );
18195:S
18196:C        2 begin
18197:C        2   with gcbi^,knob_device_rec_ptr(dev_dep_stuff)^ do
18198:C        3     begin
18199:C        3       digitize_enabled := false;
18200:C        3       info1 := last_x;
18201:C        3       info2 := last_y;
18202:C        3       info3 := abs(last_but);
18203:C        3       last_but := 0;
18204:C        3     end
18205:C        3 end;
18206:S
18207:D        1 procedure knob_input_echo (  gcbi : graphics_input_control_block_ptr );
18208:S
18209:C        2 begin
18210:C        2   if gcbi^.info1 <> 0 then write(#G);
18211:C        2 end;
18212:S
18213:D        1 procedure gle_init_knob_input (  gcbi : graphics_input_control_block_ptr);
18214:S
18215:C        2 begin
18216:C        2   with gcbi^,knob_device_rec_ptr(dev_dep_stuff)^ do
18217:C        3     begin
18218:C        3       input_handler_name := 'KNOB  ';
18219:C        3       input_handler_char_count := 4;
18220:C        3
18221:C        3       sample          := knob_sample;
18222:C        3       start_digitize  := knob_start_digitize;
18223:C        3       get_digitize    := knob_get_digitize;
18224:C        3       input_echo      := knob_input_echo;
18225:C        3       input_escapei   := knob_dummy;
18226:C        3       input_escapeo   := knob_dummy;
18227:C        3       inq_p1p2        := knob_get_input_p1p2;
18228:C        3
18229:C        3       case knob_type of
18230:C        4         1 : begin
18231:C        4               input_name := '9816A ';
18232:C        4               input_name_char_count := 5;
18233:C        4               input_res_x := 2.375;              { 168mm X 126mm }
18234:C        4               input_res_y := 2.37301587301587;
18235:C        4             end;
18236:C        4         2 : begin
18237:C        4               input_name := '9826A ' ;
18238:C        4               input_name_char_count := 5;
18239:C        4               input_res_x := 3.325;              { 120mm X 90mm }
18240:C        4               input_res_y := 3.32222222222222;
18241:C        4             end;
18242:C        4         3 : begin
18243:C        4               input_name := '9836A ';
18244:C        4               input_name_char_count := 5;
18245:C        4               input_res_x := 2.43333333333333;   { 210mm X 160mm }
18246:C        4               input_res_y := 2.43125;
18247:C        4             end;
18248:C        4         4 : begin
18249:C        4               input_name := '9836C ';
```



```
18250:C        4               input_name_char_count := 5;
18251:C        4               input_res_x := 2.35483870967742;   { 217mm X 163mm }
18252:C        4               input_res_y := 2.38650306748466;
18253:C        4             end;
18254:C        4         otherwise ;
18255:C        4       end;
18256:C        3
18257:C        3       input_cpx := info1;             { setup init input values }
18258:C        3       input_cpy := info2;
18259:C        3       last_x    := info1;
18260:C        3       last_y    := info2;
18261:C        3       last_but  := 0;
18262:S
18263:C        3       knob_get_input_p1p2 ( gcbi );
18264:C        3       input_min_x := info1;
18265:C        3       input_max_x := info2;
18266:C        3       input_min_y := info3;
18267:C        3       input_max_y := info4;
18268:C        3
18269:C        3       digitize_enabled := false;
18270:C        3       echo_rate := 1;
18271:C        3       error_return := 0;
18272:C        3     end;
18273:C        2 end;
18274:S
18275:C        1 end. { knob_input }
18276:S
18277:D        1 $LIST ON$

No errors. No warnings.
             ***** Nonstandard language features enabled *****
```

Description

GLE_RGL provides high-level device handler routines for raster devices.

Requirements

GLE_TYPES, SYSGLOBALS, SYSDEVS, GLE_ARAS_OUT, GLE_STEXT, GLE_ASTEXT, GLE_SCLIP, GLE_ASCLIP, GLE_SMARK, and GLE_AUTL.

```
     1:D        0 {                                                                                 }
     2:D        0 { Graphics Low End                                                                }
     3:D        0 {                                                                                 }
     4:D        0 { Module    = GLE_RAS_OUT                                                         }
     5:D        0 { Programer = BJS                                                                 }
     6:D        0 { Date      = 11-05-82                                                            }
     7:D        0 {                                                                                 }
     8:D        0 { Purpose: To provide device handler routines for raster devices.                 }
     9:S
    10:D        0 { Rev history                                                                     }
    11:D        0 {  Created  - 11-05-82                                                            }
    12:D        0 {  Modified - 02-14-84  BDS  (added code for gator black/white)                   }
    13:S
    14:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    15:S                All rights are reserved.  Copying or other
    16:S                reproduction of this program except for archival
    17:S                purposes is prohibited without the prior
    18:S                written consent of Hewlett-Packard Company.
    19:S
    20:S
    21:S                           RESTRICTED RIGHTS LEGEND
    22:S
    23:S                Use, duplication, or disclosure by the Government
    24:S                is subject to restrictions as set forth in
    25:S                paragraph (b) (3) (B) of the Rights in Technical
    26:S                Data and Computer Software clause in
    27:S                DAR 7-104.9(a).
    28:S
    29:S                HEWLETT-PACKARD COMPANY
    30:D        0       Fort Collins, Colorado                                  }
    31:S
    32:D        0 $search 'GLE_TYPES',
    33:D        0         'GLE_STEXT',
    34:D        0         'ASM_STEXT',
    35:D        0         'GLE_SCLIP',
    36:D        0         'ASM_SCLIP',
    37:D        0         'GLE_SMARK',
    38:D        0         'GLE_AUTL',
    39:D        0         'RGL'$
    40:D        0 $modcal$
    40:D        0 $include 'OPTIONS'$       { ****************** COMPILER OPTIONS ********************** }
    41:S
    42:S           { This include file specifies range checking, debug and other compiler
    43:D        0    options for the graphics library                                          }
    44:S
    45:D        0 $debug OFF$
    46:D        0 $range OFF$
    47:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    48:D        0 $FLOAT_HDW TEST$
    49:S
    50:S
    51:S
    52:S
    53:D        0 $include 'OPTIONS'$       { ****************** COMPILER OPTIONS ********************** }
  8000:D        0 $linenum 8000$
  8001:S
  8002:D        0 module gle_ras_out;
  8003:S
  8004:D        1 import gle_types,sysglobals,sysdevs,gle_aras_out;
  8005:S
```



```
  8006:D        1 export
  8007:S
  8008:D        1 procedure gle_init_raster_output (  gcb : graphics_control_block_ptr);
  8009:D        1 procedure gator_clear ( gcb : graphics_control_block_ptr );
  8010:S
  8011:D        1 type
  8012:D        1   raster_byte = 0..255;
  8013:D        1   raster_code_space = packed array[1..240] of raster_byte;
  8014:D        1   dither_type = packed array [0..15] of raster_byte;
  8015:D        1   cmap_def = packed record
  8016:D        1                map_red : gle_shortint;
  8017:D        1                map_grn : gle_shortint;
  8018:D        1                map_blu : gle_shortint;
  8019:D        1              end;
  8020:S
  8021:D        1   system_cmap_def = packed array [0..15] of cmap_def;
  8022:S
  8023:S
  8024:D        1   raster_device_rec_ptr = ^ raster_device_rec;
  8025:D        1   raster_device_rec =
  8026:D        1     record
  8027:D        1       addr1           : anyptr   ;
  8028:D        1       addr2           : anyptr   ;
  8029:D        1       addr3           : anyptr   ;
  8030:D        1       n3              : gle_shortint ;
  8031:D        1       devicetype      : gle_shortint ;
  8032:D        1       deviceaddress   : integer  ;
  8033:D        1       monitortype     : gle_shortint ;
  8034:D        1       plane1_addr     : anyptr   ;
  8035:D        1       plane1_offset   : integer  ;
  8036:D        1       plane2_offset   : integer  ;
  8037:D        1       plane3_offset   : integer  ;
  8038:D        1       n_glines        : gle_shortint ;
  8039:D        1       gspacing        : gle_shortint ;
  8040:D        1       bytesperline    : gle_shortint ;
  8041:D        1       hard_xmax       : gle_shortint ;
  8042:D        1       hard_ymax       : gle_shortint ;
  8043:D        1       red_intensity   : gle_shortint ;
  8044:D        1       grn_intensity   : gle_shortint ;
  8045:D        1       blu_intensity   : gle_shortint ;
  8046:D        1       dither_pattern  : dither_type;
  8047:D        1       cursor_x        : gle_shortint ;
  8048:D        1       cursor_y        : gle_shortint ;
  8049:D        1       area_draw_mode  : gle_shortint ;
  8050:D        1       pen_draw_mode   : gle_shortint ;
  8051:D        1       linepattern     : gle_shortint ;
  8052:D        1       pen_number      : gle_shortint ;
  8053:D        1       cpen            : gle_shortint ;
  8054:D        1       oldpattern      : gle_shortint ;
  8055:D        1       rgltemp1        : integer  ;
  8056:D        1       rgltemp2        : integer  ;
  8057:D        1       rgltemp3        : integer  ;
  8058:D        1       rgltemp4        : integer  ;
  8059:D        1       rgltemp5        : integer  ;
  8060:D        1       repeatrate      : gle_shortint ;
  8061:D        1       repeatcount     : gle_shortint ;
  8062:D        1       index           : integer  ;
  8063:D        1       softvec         : raster_code_space  ;
  8064:D        1       system_cmap       : system_cmap_def;
  8065:D        1       brightness_sequence : packed array [0..15] of gle_shortint;
```

```
8066:D        1       count               : packed array [0..15] of gle_shortint;
8067:D        1       cmap_address    : integer;
8068:D        1     end;
8069:S
8070:D        1 const
8071:D        1   packed_pixel_odd_byte_display  = 0;
8072:D        1   packed_pixel_display           = 1;
8073:D        1   packed_pixel_3_plane_display   = 2;
8074:D        1   byte_per_pixel_display         = 3;
8075:D        1   gator_display                  = 4;
8076:S
8077:D        1   dominant                       = 3;
8078:D        1   erase                          = 0;
8079:D        1   non_dominant                  = 7;
8080:D        1   compliment                    = 10;
8081:D        1   clr_with_LM                   = 128;
8082:D        1   one_with_LM                   = 255;
8083:S
8084:D        1 implement
8085:S
8086:D        1 import gle_stext,
8087:D        1        gle_astext,
8088:D        1        gle_sclip,
8089:D        1        gle_asclip,
8090:D        1        gle_smark,
8091:D        1        gle_autl;
8092:S
8093:D        1 type
8094:D        1   word_array = packed array[1..maxint] of gle_shortint;
8095:D        1   fixed_word_array = array [0..13] of gle_shortint;
8096:D        1   map_array = array[1..maxint] of integer;
8097:D        1   wd_ptr = ^word_array;
8098:D        1   byte_array = packed array [0..maxint] of raster_byte;
8099:S
8100:D        1 var
8101:D    -4  1   fb_ptr : ^byte_array;
8102:D    -8  1   fb_ptr_ptr : ^integer;
8103:D    -8  1
8104:S
8105:S
8106:D    -8  1 {const}
8107:D    -8  1   {init_crt = fixed_word_array [22,16,21,01,48,11,48,48,0,15,0,0,0,0 ];{17")
8108:D    -8  1   {init_crt = fixed_word_array [20,16,17,2,48,11,48,48,0,15,0,0,0,0 ];{19")
8109:D    -8  1 {init_crt = fixed_word_array [20,16,16,4,48,11,48,48,0,15,0,0,32,0 ];{17"-NEW}
8110:S
8111:S
8112:S
8113:D        1 PROCEDURE wait_ready;
8114:S
8115:C        2 BEGIN
8116:C        2   REPEAT
8117:C        3   UNTIL status^.notbusy
8118:C        3 END;
8119:S
8120:D        1 PROCEDURE set_rule ( rule : INTEGER );
8121:S
8122:C        2 BEGIN
8123:C        2   replregcopy := rule;
8124:C        2   wait_ready;
8125:C        2   rule_reg^ := rule;
```



```
8126:C        2 END;
8127:S
8128:S          {PROCEDURE set_ww ( rule : INTEGER );
8129:S
8130:S          BEGIN
8131:S            writeregcopy := rule;
8132:S            wait_ready;
8133:S            ww_reg^ := rule;
8134:D    -8  1 END;}
8135:S
8136:D        1 PROCEDURE set_width ( width : INTEGER );
8137:S
8138:C        2 BEGIN
8139:C        2   windowregcopy := width;
8140:C        2   wait_ready;
8141:C        2   width_reg^ := width;
8142:C        2 END;
8143:S
8144:S
8145:S
8146:D        1 procedure gator_fill_index_color ( gcb : graphics_control_block_ptr );
8147:S
8148:D        2 type
8149:D        2      dpt1 = packed array [0..15] of boolean;
8150:D        2      dpt2 = packed array [0..16] of dpt1;
8151:D        2 var
8152:D    -4  2      i : integer;
8153:D    -8  2      j : integer;
8154:S
8155:D   -12  2      r : integer;
8156:D   -16  2      g : integer;
8157:D   -20  2      b : integer;
8158:S
8159:D   -20  2 const
8160:D   -20  2      t = true;
8161:D   -20  2      f = false;
8162:D   -20  2      dp = dpt2[dpt1[f,f,f,f, f,f,f,f, f,f,f,f, f,f,f,f],
8163:D   -20  2                dpt1[t,f,f,f, f,f,f,f, f,f,f,f, f,f,f,f],
8164:D   -20  2                dpt1[t,f,f,f, f,f,f,f, f,f,t,f, f,f,f,f],
8165:D   -20  2                dpt1[t,f,f,f, f,f,f,f, t,f,t,f, f,f,f,f],
8166:D   -20  2                dpt1[t,f,t,f, f,f,f,f, t,f,t,f, f,f,f,f],
8167:D   -20  2                dpt1[t,f,t,f, f,t,f,f, t,f,t,f, f,f,f,f],
8168:D   -20  2                dpt1[t,f,t,f, f,t,f,f, t,f,t,f, f,f,f,t],
8169:D   -20  2                dpt1[t,f,t,f, f,t,f,t, t,f,t,f, f,f,f,t],
8170:D   -20  2                dpt1[t,f,t,f, f,t,f,t, t,f,t,f, f,t,f,t],
8171:D   -20  2                dpt1[t,f,t,f, t,t,f,t, t,f,t,f, f,t,f,t],
8172:D   -20  2                dpt1[t,f,t,f, t,t,f,t, t,f,t,f, f,t,t,t],
8173:D   -20  2                dpt1[t,f,t,f, t,t,f,t, t,f,t,f, t,t,t,t],
8174:D   -20  2                dpt1[t,f,t,f, t,t,t,t, t,f,t,f, t,t,t,t],
8175:D   -20  2                dpt1[t,t,t,f, t,t,t,t, t,f,t,f, t,t,t,t],
8176:D   -20  2                dpt1[t,t,t,f, t,t,t,t, t,f,t,t, t,t,t,t],
8177:D   -20  2                dpt1[t,t,t,f, t,t,t,t, t,t,t,t, t,t,t,t],
8178:D   -20  2                dpt1[t,t,t,t, t,t,t,t, t,t,t,t, t,t,t,t]];
8179:S
8180:C        2 begin
8181:C        2   with gcb^,raster_device_rec_ptr(dev_dep_stuff)^ do
8182:C        3     begin
8183:C        3       if info1 = 0 then                    (set up dither pattern)
8184:C        4         begin
8185:C        4           r := info2;
```

```
8186:C        4           g := info3;
8187:C        4           b := info4;
8188:S
8189:C        4           current_polygon_red := r;
8190:C        4           current_polygon_green := g;
8191:C        4           current_polygon_blue := b;
8192:S
8193:C        4           r := (r+32) div 64;
8194:C        4           g := (g+32) div 64;
8195:C        4           b := (b+32) div 64;
8196:S
8197:C        4           red_intensity := r;
8198:C        4           grn_intensity := g;
8199:C        4           blu_intensity := b;
8200:S
8201:C        4           i:=r;
8202:C        4           if i<g then i:=g;
8203:C        4           if i<b then i:=b;
8204:S
8205:C        4           for j:=0 to 15 do dither_pattern[j]:=0;
8206:C        4           for j:=0 to 15 do if dp[i,j] then dither_pattern[j]:=1;
8207:S
8208:C        4           store_dit(gcb);
8209:C        4           end
8210:C        4       else
8211:C        4       if info1 = 1 then
8212:C        5         begin
8213:C        5           for i := 0 to 15 do
8214:C        6              dither_pattern[i] := info2;
8215:C        5           store_dit(gcb);
8216:C        5         end;
8217:S
8218:C        3     end;
8219:C        2 end;
8220:S
8221:S
8222:S
8223:D        1 procedure gator_clear ( gcb : graphics_control_block_ptr );
8224:S
8225:D    -4  2 var i : integer;
8226:S
8227:C        2 begin
8228:C        2   if gcb^.info2 = 0 then set_rule(clr_with_LM)
8229:C        3   else                   set_rule(one_with_LM);
8230:C        2   set_width(-1024);
8231:C        2   with gcb^,raster_device_rec_ptr(dev_dep_stuff)^ do
8232:C        3     begin
8233:C        3       for i := 0 to hard_ymax do          (fix to allow for changing ymax)
8234:C        4         begin
8235:C        4           wait_ready;
8236:C        4           fb_ptr^ [i*1024] := 0;
8237:C        4         end;
8238:C        3     end;
8239:C        2   set_rule(dominant);
8240:C        2 end;
8241:S
8242:S
8243:S
8244:D        1 procedure setup_gator(gcb : graphics_control_block_ptr);
8245:S
```



```
8246:D        2 var
8247:D   -96  2   color_list : packed array [0..47] of gle_shortint;
8248:D  -100  2   i : integer;
8249:S
8250:C        2 begin
8251:C        2   {with gcb^ do)
8252:S             {if info1 = mgator_c then begin
8253:C        2       set_ww(0);}
8254:C        2                              { I moved this as a fix }
8255:S             {blink1_reg := 255;
8256:S              blink2_reg := 255;
8257:C        2    end;}
8258:C        2   set_6845s (gcb);
8259:S             {for i := 0 to 13 do
8260:S               begin
8261:S                 crt^[1] := i;
8262:S                 crt^[2] := init_crt[I];  SETUP 6845
8263:C        2      end;}
8264:C        2   status^.bit0 := write_to_status;
8265:S             {color_list[0] := 0;
8266:S             color_list[1] := 0;
8267:C        2    color_list[2] := 0;}
8268:C        2   i := 0;
8269:S             {repeat
8270:S               i := i + 3;
8271:S               color_list[i] := 1023;
8272:S               color_list[i+1] := 0;
8273:S               color_list[i+2] := 0;
8274:C        2    until i = 45;}
8275:C        2    with gcb^ do
8276:C        3      begin
8277:C        3        info1 := 0;
8278:C        3        info2 := 1;
8279:C        3        info_ptr1 := addr(color_list);
8280:C        3        call (define_color_map,gcb);
8281:C        3      end;
8282:C        2   set_rule(dominant);
8283:C        2 end;
8284:S
8285:S           {procedure rdummy (  anyvar a,b : anyptr);
8286:S
8287:S           begin
8288:D    -8  1 end;}
8289:S
8290:D        1 procedure rdummy_proc (  gcb : graphics_control_block_ptr );
8291:S
8292:D        2 VAR
8293:D    -2  2   I : GLE_SHORTINT;
8294:S
8295:C        2 begin
8296:C        2   with gcb^,raster_device_rec_ptr(dev_dep_stuff)^ do
8297:S             {for I := 0 to 15 do
8298:C        3     WRITELN(LISTING,'DIT[',I:0,'] = ',DITHER_PATTERN[I]);}
8299:C        3 end;
8300:S
8301:D        1 procedure rget_polygon_info (  gcb : graphics_control_block_ptr );
8302:S
8303:C        2 begin             { only solid fill supported }
8304:C        2   with gcb^ do
8305:C        3     if info2 = polygon_solid_fill then error_return := 0
```

```
8306:C        4     else                                  error_return := 1;
8307:C        2 end;
8308:S
8309:D        1 procedure rdefine_color_map (  gcb : graphics_control_block_ptr );
8310:S
8311:D        2 type
8312:D        2   color_map_def = packed array [0..15] of gle_shortint;
8313:D        2   color_map_ptr = ^color_map_def;
8314:D        2   color_data_def = packed array [0..47] of gle_shortint;
8315:D        2   color_data_ptr_def = ^color_data_def;
8316:S
8317:D        2 var
8318:D    -4  2   color_map : color_map_ptr;
8319:D   -36  2   temp_color_map : color_map_def;
8320:D   -40  2   n,i : gle_shortint;
8321:D   -44  2   temp : integer;
8322:D  -108  2   brt : array [0..15] of integer;
8323:D  -112  2   color_data_ptr : color_data_ptr_def;
8324:S
8325:C        2 begin
8326:C        2   with gcb^,raster_device_rec_ptr(dev_dep_stuff)^ do
8327:C        3     begin
8328:C        3       color_data_ptr := info_ptr1;
8329:C        3       for i := info1 to info2 do
8330:C        4         with system_cmap[i] do
8331:C        5           begin
8332:C        5             n := (i - info1)*3;
8333:C        5             map_red := color_data_ptr^[n];
8334:C        5             map_grn := color_data_ptr^[n+1];
8335:C        5             map_blu := color_data_ptr^[n+2];
8336:C        5             temp_color_map[i] := (15-map_red div 64)*256 +
8337:C        5                                  (15-map_grn div 64) * 16 +
8338:C        5                                  (15-map_blu div 64);
8339:C        5           end;
8340:S
8341:C        3       if color_map_support = yes then
8342:C        4         begin
8343:C        4           color_map := anyptr(cmap_address);
8344:C        4           rawait_blanking(gcb);
8345:C        4           for i := info1 to info2 do
8346:C        5             color_map^[i] := temp_color_map[i];
8347:C        4         end;
8348:S
8349:C        3       for i := 0 to 15 do
8350:C        4         with system_cmap[i] do
8351:C        5           begin
8352:C        5             brt[i] := map_red*23+map_grn*69+map_blu*8;
8353:C        5             brightness_sequence[i] := i;
8354:C        5           end;
8355:C        3       for n := 0 to 14 do
8356:C        4         for i := n+1 to 15 do
8357:C        5           if brt[i]>brt[n] then
8358:C        6             begin
8359:C        6               temp := brt[i];
8360:C        6               brt[i] := brt[n];
8361:C        6               brt[n] := temp;
8362:C        6               temp := brightness_sequence[i];
8363:C        6               brightness_sequence[i] := brightness_sequence[n];
8364:C        6               brightness_sequence[n] := temp;
8365:C        6             end;
```



```
8366:S              {FOR N := 0 TO 15 DO
8367:S                WRITELN(LISTING,'N = ',N,'BRI[N]= ',BRIGHTNESS_SEQUENCE[N]);
8368:S              FOR N := 0 TO 15 DO
8369:S                WITH SYSTEM_CMAP[N] DO
8370:C        3         WRITELN(LISTING,'N = ',N,'CMAP[N]= ',MAP_RED:8,MAP_GRN:8,MAP_BLU:8);}
8371:C        3     end;
8372:C        2 end;
8373:S
8374:D        1 procedure gle_init_raster_output (  gcb : graphics_control_block_ptr);
8375:S
8376:D        2 var
8377:D    -2  2   cnt : gle_shortint;
8378:D   -10  2   I,id : integer;
8379:S
8380:C        2 begin
8381:C        2   with gcb^,raster_device_rec_ptr(dev_dep_stuff)^ do
8382:C        3     begin
8383:C        3       if (info1 = m9837a)  then                  {or (info1 = mgator_c)}
8384:C        4         begin
8385:C        4           fb_ptr_ptr := info_ptr1;
8386:C        4           fb_ptr := anyptr(fb_ptr_ptr^);
8387:C        4           rule_reg := anyptr(info2 + (hex('4008')));
8388:C        4           width_reg := anyptr(info2 + (hex('400c')));
8389:C        4           status := anyptr(info2 + (hex('4000')));
8390:C        4           crt := anyptr(info2 + (hex('6000')));
8391:C        4           {if info1 = mgator_c then ww_reg := anyptr(info2 + (hex('4108')));}
8392:C        4         end;
8393:C        3       error_return := 0;
8394:C        3       rgcbinit ( gcb );
8395:S
8396:C        3       move                  := rmove;
8397:C        3       draw                  := rdraw;
8398:C        3       clear                 := rclear;
8399:C        3       text                  := gle_soft_text;
8400:C        3       clip_limits           := gle_soft_clip_limits;
8401:C        3       char_size             := gle_soft_char_size;
8402:C        3       text_spacing          := gle_soft_text_spacing;
8403:C        3       text_dir              := gle_soft_text_dir;
8404:C        3       text_just             := gle_soft_text_just;
8405:C        3       marker                := gle_soft_marker;
8406:C        3       marker_size           := gle_soft_marker_size;
8407:C        3       set_marker            := gle_soft_set_marker;
8408:C        3       index_color           := rset_color;
8409:C        3       linestyle             := rlinestyle;
8410:C        3       await_blanking        := rawait_blanking;
8411:C        3       linewidth             := rdummy_proc;
8412:C        3       inq_p1p2              := rget_p1p2;
8413:C        3       get_polygon_info      := rget_polygon_info;
8414:C        3       calc_soft_text_xform  := gle_text_xform;
8415:C        3       fill_index_color      := rfill_index_color;
8416:C        3       graphics_on_off       := rgraphics_on_off;
8417:C        3       cursor                := rcursor;
8418:C        3       define_drawing_mode   := rdefine_drawing_mode;
8419:C        3       polygon               := rpolygon;
8420:C        3       define_color_map      := rdefine_color_map;
8421:C        3       buffer_mode           := rdummy_proc;
8422:C        3       output_escapei        := rdummy_proc;
8423:C        3       output_escapeo        := rdummy_proc;
8424:C        3       flush_buffer          := rdummy_proc;
8425:C        3       get_raster            := rget_raster;
```

```
8426:S
8427:C        3       soft_font_ptr        := addr(font);
8428:S
8429:C        3       case info1 of
8430:C        4         munknown : { reserved } ;
8431:C        4         m9816a   : { 9816 }
8432:C        4           begin
8433:C        4             display_name := '9816A ';
8434:C        4             display_name_char_count := 5;
8435:C        4             display_res_x := 2.375;                { 168mm X 126mm }
8436:C        4             display_res_y := 2.373015873015873;
8437:C        4           end;
8438:C        4         m9826a :   { 9826 }
8439:C        4           begin
8440:C        4             display_name := '9826A ';
8441:C        4             display_name_char_count := 5;
8442:C        4             display_res_x := 3.325;                { 120mm X 90mm }
8443:C        4             display_res_y := 3.322222222222222;
8444:C        4           end;
8445:C        4         m9836a : { 9836 }
8446:C        4           begin
8447:C        4             display_name := '9836A ';
8448:C        4             display_name_char_count := 5;
8449:C        4             display_res_x := 2.433333333333333;     { 210mm X 160mm }
8450:C        4             display_res_y := 2.43125;
8451:C        4           end;
8452:C        4         m9836c : { 9836C }
8453:C        4           begin
8454:C        4             display_name := '9836C ';
8455:C        4             display_name_char_count := 5;
8456:C        4             display_res_x := 2.35483870967742;      { 217mm X 163mm }
8457:C        4             display_res_y := 2.38650306748466;
8458:C        4           end;
8459:C        4         m98627a : { 98627 }
8460:C        4           begin
8461:C        4             display_name := '98627A';
8462:C        4             display_name_char_count := 6;
8463:C        4             deviceaddress := info2;
8464:C        4             display_res_x := 3.333333333333333;
8465:C        4             display_res_y := 3.333333333333333;
8466:C        4           end;
8467:C        4         m9837a : { Gator Black/White }
8468:C        4           begin
8469:C        4             display_name := '9837a ';
8470:C        4             display_name_char_count := 5;
8471:C        4             display_res_x := 3.291666666666667;     { 312mm X  234mm }
8472:C        4             display_res_y := 3.282051282051282;
8473:C        4             if info3 = 1 then                       {added to detect current}
8474:C        5               begin                                  {ymax.                  }
8475:C        5                 n_glines := 768;
8476:C        5                 hard_ymax := 767;
8477:C        5               end;
8478:C        4             await_blanking := rdummy_proc;
8479:C        4             setup_gator(gcb);
8480:C        4             clear := gator_clear;
8481:C        4             fill_index_color := gator_fill_index_color;
8482:C        4             define_color_map := rdummy_proc;
8483:C        4           end;
8484:S                  {mgator_c :  Gator Color
8485:S                    begin
```



```
8486:S                      display_name := 'Gator ';
8487:S                      display_name_char_count := 5;
8488:S                      display_res_x := 3.291666666666667;      312mm X  234mm
8489:S                      display_res_y := 3.282051282051282;
8490:S                      await_blanking := rdummy_proc;
8491:S                      setup_gator(gcb);
8492:S                      clear := gator_clear;
8493:S                      define_color_map := rdummy_proc;
8494:S                  INFO2 := 0;                                  WHY IS THIS HERE ??
8495:S                  GATOR_CLEAR(GCB);
8496:C        4             end;}
8497:C        4           end; {of case }
8498:S
8499:S
8500:C        3       display_handler_name := 'RASTER';
8501:C        3       display_handler_char_count := 6;
8502:S
8503:C        3       display_min_x := 0;
8504:C        3       display_min_y := 0;
8505:C        3       display_max_x := hard_xmax;
8506:C        3       display_max_y := hard_ymax;
8507:S
8508:S
8509:C        3       info1 := 0; info2 := display_max_x;
8510:C        3       info3 := 0; info4 := display_max_y;
8511:C        3       gle_soft_clip_limits ( gcb );  { set default clipping limits }
8512:S
8513:C        3       pen_draw_mode := 0;
8514:C        3       area_draw_mode := 0;
8515:C        3       pen_number := 1;
8516:C        3       info1 := 1;
8517:C        3       rset_color(gcb);
8518:C        3     end;
8519:C        2 end;
8520:S
8521:C        1 end. { output }
8522:S

No errors. No warnings.
          ***** Nonstandard language features enabled *****
```

GLE_SCLIP

Description

GLE_SCLIP provides a routine which sets GLE clip limits.

Requirements

GLE_TYPES.

```
Pascal [Rev 3.0M  6/ 4/84] GLE_SCLIP.TEXT          26-Dec-84 22:31:03 Page 1   *** File name: GLE_SCLIP ***

     1:D        0 {                                                                    }
     2:D        0 { Graphics Low End                                                   }
     3:D        0 {                                                                    }
     4:D        0 { Module    = GLE_SCLIP                                              }
     5:D        0 { Programer = BJS                                                    }
     6:D        0 { Date      = 10-10-82                                               }
     7:D        0 {                                                                    }
     8:D        0 { Purpose: To provide software clipping routines.                    }
     9:S
    10:D        0 { Rev history                                                        }
    11:D        0 {  Created  - 10-10-82                                               }
    12:D        0 {  Modified - XX-XX-XX                                               }
    13:S
    14:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    15:S                All rights are reserved.  Copying or other
    16:S                reproduction of this program except for archival
    17:S                purposes is prohibited without the prior
    18:S                written consent of Hewlett-Packard Company.
    19:S
    20:S
    21:S                            RESTRICTED RIGHTS LEGEND
    22:S
    23:S                Use, duplication, or disclosure by the Government
    24:S                is subject to restrictions as set forth in
    25:S                paragraph (b) (3) (B) of the Rights in Technical
    26:S                Data and Computer Software clause in
    27:S                DAR 7-104.9(a).
    28:S
    29:S                HEWLETT-PACKARD COMPANY
    30:D        0       Fort Collins, Colorado                                 }
    31:S
    32:S
    33:D        0 $modcal$
    34:D        0 $search 'GLE_TYPES'$
    34:D        0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS ****************** }
    35:S
    36:S           { This include file specifies range checking, debug and other compiler
    37:D        0    options for the graphics library                                  }
    38:S
    39:D        0 $debug OFF$
    40:D        0 $range OFF$
    41:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    42:D        0 $FLOAT_HDW TEST$
    43:S
    44:S
    45:S
    46:S
    47:D        0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS ****************** }
  6000:D        0 $linenum 6000$
  6001:S
  6002:D        0 module gle_sclip;
  6003:S
  6004:D        1 import gle_types;
  6005:S
  6006:D        1 export
  6007:S
  6008:D        1 procedure gle_soft_clip_limits  ( gcb : graphics_control_block_ptr);
  6009:S
  6010:D        1 implement
  6011:S
```

```
Pascal [Rev 3.0M  6/ 4/84] GLE_SCLIP.TEXT          26-Dec-84 22:31:03 Page 2   *** File name: GLE_SCLIP ***

  6012:D        1 procedure gle_soft_clip_limits ( gcb : graphics_control_block_ptr );
  6013:S
  6014:C        2 begin
  6015:C        2   with gcb^ do
  6016:C        3     begin
  6017:C        3       clip_limits_xmin := info1;
  6018:C        3       clip_limits_xmax := info2;
  6019:C        3       clip_limits_ymin := info3;
  6020:C        3       clip_limits_ymax := info4;
  6021:C        3     end;
  6022:C        2 end;
  6023:S
  6024:C        1 end. { module gle_sclip }
  6025:S

No errors. No warnings.
          ***** Nonstandard language features enabled *****
```

GLE_SMARK

Description

GLE_SMARK provides software marker routines.

Requirements

GLE_TYPES, and GLE_AUTL.

```
    1:S
    2:D        0 {                                                                    }
    3:D        0 { Graphics Low End                                                   }
    4:D        0 {                                                                    }
    5:D        0 { Module    = GLE_SMARK                                              }
    6:D        0 { Programer = BJS                                                    }
    7:D        0 { Date      = 10-15-82                                               }
    8:D        0 {                                                                    }
    9:D        0 { Purpose: To provide software marker routines.                      }
   10:S
   11:D        0 { Rev history                                                        }
   12:D        0 {  Created  - 10-15-82  BJS                                          }
   13:D        0 {  Modified - XX-XX-XX                                               }
   14:S
   15:S          {     (c) Copyright Hewlett-Packard Company, 1983.
   16:S                All rights are reserved.  Copying or other
   17:S                reproduction of this program except for archival
   18:S                purposes is prohibited without the prior
   19:S                written consent of Hewlett-Packard Company.
   20:S
   21:S
   22:S                            RESTRICTED RIGHTS LEGEND
   23:S
   24:S                Use, duplication, or disclosure by the Government
   25:S                is subject to restrictions as set forth in
   26:S                paragraph (b) (3) (B) of the Rights in Technical
   27:S                Data and Computer Software clause in
   28:S                DAR 7-104.9(a).
   29:S
   30:S                HEWLETT-PACKARD COMPANY
   31:D        0       Fort Collins, Colorado                                 }
   32:S
   33:D        0 $modcal$
   34:D        0 $search 'GLE_AUTL',
   35:D        0         'GLE_TYPES'$
   36:S
   36:D        0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS ******************** }
   37:S
   38:S           { This include file specifies range checking, debug and other compiler
   39:D        0    options for the graphics library                                  }
   40:S
   41:D        0 $debug OFF$
   42:D        0 $range OFF$
   43:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
   44:D        0 $FLOAT_HDW TEST$
   45:S
   46:S
   47:S
   48:S
   49:D        0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS ******************** }
 5000:D        0 $linenum 5000$
 5001:S
 5002:D        0 module gle_smark;
 5003:S
 5004:D        1 import gle_types;
 5005:S
 5006:D        1 export
 5007:S
 5008:D        1 procedure gle_soft_marker       ( gcb : graphics_control_block_ptr);
 5009:D        1 procedure gle_soft_set_marker   ( gcb : graphics_control_block_ptr );
```



```
 5010:D        1 procedure gle_soft_marker_size  ( gcb : graphics_control_block_ptr );
 5011:S
 5012:D        1 implement
 5013:S
 5014:D        1 import gle_autl;
 5015:S
 5016:D        1 procedure gle_soft_marker ( gcb : graphics_control_block_ptr);
 5017:S
 5018:D        2 const
 5019:D        2   marker_cell_shift   = -2; { part of marker viewing transformation }
 5020:S
 5021:D        2 type marker_data      = -2..2;
 5022:D        2      index_def       = 0..50;
 5023:D        2      marker_record   = packed record
 5024:D        2                          x_pos : marker_data;
 5025:D        2                          y_pos : marker_data;
 5026:D        2                          penup : boolean;
 5027:D        2                        end;
 5028:D        2      mark_def        = packed array [1..47] of marker_record;
 5029:D        2      marker_index_def = packed array [1..10] of index_def;
 5030:S
 5031:D        2 const
 5032:D        2   markers =
 5033:D        2   mark_def [ marker_record [x_pos : 0, y_pos : 0, penup : true ], {1}
 5034:D        2                marker_record [x_pos : 0, y_pos : 0, penup : false], {2}
 5035:D        2                marker_record [x_pos : 0, y_pos : 2, penup : true], {3}
 5036:D        2                marker_record [x_pos : 0, y_pos :-2, penup : false], {4}
 5037:D        2                marker_record [x_pos : 2, y_pos : 0, penup : true], {5}
 5038:D        2                marker_record [x_pos :-2, y_pos : 0, penup : false], {6}
 5039:D        2                marker_record [x_pos : 2, y_pos : 0, penup : true], {7}
 5040:D        2                marker_record [x_pos :-2, y_pos : 0, penup : false], {8}
 5041:D        2                marker_record [x_pos :-2, y_pos : 2, penup : true], {9}
 5042:D        2                marker_record [x_pos : 2, y_pos :-2, penup : false], {10}
 5043:D        2                marker_record [x_pos : 2, y_pos : 2, penup : true], {11}
 5044:D        2                marker_record [x_pos :-2, y_pos :-2, penup : false], {12}
 5045:D        2                marker_record [x_pos :-1, y_pos :-2, penup : true], {13}
 5046:D        2                marker_record [x_pos : 1, y_pos :-2, penup : false], {14}
 5047:D        2                marker_record [x_pos : 2, y_pos :-1, penup : false], {15}
 5048:D        2                marker_record [x_pos : 2, y_pos : 1, penup : false], {16}
 5049:D        2                marker_record [x_pos : 1, y_pos : 2, penup : false], {17}
 5050:D        2                marker_record [x_pos :-1, y_pos : 2, penup : false], {18}
 5051:D        2                marker_record [x_pos :-2, y_pos : 1, penup : false], {19}
 5052:D        2                marker_record [x_pos :-2, y_pos :-1, penup : false], {20}
 5053:D        2                marker_record [x_pos :-1, y_pos :-2, penup : false], {21}
 5054:D        2                marker_record [x_pos :-2, y_pos :-2, penup : true], {22}
 5055:D        2                marker_record [x_pos : 2, y_pos : 2, penup : false], {23}
 5056:D        2                marker_record [x_pos :-2, y_pos : 2, penup : true], {24}
 5057:D        2                marker_record [x_pos : 2, y_pos :-2, penup : false], {25}
 5058:D        2                marker_record [x_pos : 0, y_pos : 2, penup : true], {26}
 5059:D        2                marker_record [x_pos : 2, y_pos :-2, penup : false], {27}
 5060:D        2                marker_record [x_pos :-2, y_pos :-2, penup : false], {28}
 5061:D        2                marker_record [x_pos : 0, y_pos : 2, penup : false], {29}
 5062:D        2                marker_record [x_pos :-2, y_pos :-2, penup : true], {30}
 5063:D        2                marker_record [x_pos : 2, y_pos :-2, penup : false], {31}
 5064:D        2                marker_record [x_pos : 2, y_pos : 2, penup : false], {32}
 5065:D        2                marker_record [x_pos :-2, y_pos : 2, penup : false], {33}
 5066:D        2                marker_record [x_pos :-2, y_pos :-2, penup : false], {34}
 5067:D        2                marker_record [x_pos : 0, y_pos : 2, penup : true], {35}
 5068:D        2                marker_record [x_pos : 2, y_pos : 0, penup : false], {36}
 5069:D        2                marker_record [x_pos : 0, y_pos :-2, penup : false], {37}
```

```
5070:D        2              marker_record [x_pos :-2, y_pos : 0, penup : false], {38}
5071:D        2              marker_record [x_pos : 0, y_pos : 2, penup : false], {39}
5072:D        2              marker_record [x_pos :-2, y_pos :-2, penup : true],  {40}
5073:D        2              marker_record [x_pos : 2, y_pos :-2, penup : false], {41}
5074:D        2              marker_record [x_pos : 2, y_pos : 2, penup : false], {42}
5075:D        2              marker_record [x_pos :-2, y_pos : 2, penup : false], {43}
5076:D        2              marker_record [x_pos :-2, y_pos :-2, penup : false], {44}
5077:D        2              marker_record [x_pos : 2, y_pos : 2, penup : false], {45}
5078:D        2              marker_record [x_pos : 2, y_pos :-2, penup : true],  {46}
5079:D        2              marker_record [x_pos :-2, y_pos : 2, penup : false]];
5080:S
5081:D        2   marker_index =
5082:D        2     marker_index_def [ {1} 1,
5083:D        2                        {2} 3,
5084:D        2                        {3} 7,
5085:D        2                        {4} 13,
5086:D        2                        {5} 22,
5087:D        2                        {6} 26,
5088:D        2                        {7} 30,
5089:D        2                        {8} 35,
5090:D        2                        {9} 40,
5091:D        2                    {last} 48];
5092:S
5093:D        2 var
5094:D    -4  2   i : integer;
5095:D    -6  2   s : packed array [1..1] of char;
5096:D    -8  2   cpx : gle_shortint;
5097:D   -10  2   cpy : gle_shortint;
5098:D   -12  2   dx : gle_shortint;
5099:D   -14  2   dy : gle_shortint;
5100:D   -16  2   local_x_pos : gle_shortint;
5101:D   -18  2   local_y_pos : gle_shortint;
5102:D   -22  2   saved_char_width   : integer;
5103:D   -26  2   saved_char_height  : integer;
5104:D   -30  2   saved_text_cos_dir : integer;
5105:D   -34  2   saved_text_sin_dir : integer;
5106:D   -38  2   saved_line_pattern : integer;
5107:D   -42  2   saved_linestyle    : integer;
5108:D   -46  2   saved_length       : integer;
5109:D   -50  2   saved_mode         : integer;
5110:S
5111:C        2 begin
5112:C        2   with gcb^ do
5113:C        3     begin
5114:C        3       saved_mode := current_linestyle_mode;
5115:C        3       saved_length := current_pattern_length;
5116:C        3       saved_linestyle := current_linestyle;
5117:C        3       saved_line_pattern := current_linestyle_pattern;
5118:C        3       info1 := 0; info2 := 1;
5119:C        3       info3 := 0; info4 := -1;
5120:C        3       call ( linestyle, gcb );
5121:C        3       cpx := current_pos_x;
5122:C        3       cpy := current_pos_y;
5123:C        3       if marker_type > 9 then
5124:C        4         begin
5125:C        4           saved_char_width   := char_width;
5126:C        4           saved_char_height  := char_height;
5127:C        4           saved_text_cos_dir := text_cos_dir;
5128:C        4           saved_text_sin_dir := text_sin_dir;
5129:C        4           info1 := marker_width * 8;
```



```
5130:C        4           info2 := marker_height * 8;
5131:C        4           call ( char_size, gcb );
5132:C        4           info1 := 32768;
5133:C        4           info2 := 0;
5134:C        4           call ( text_dir, gcb );
5135:C        4           s[1] := chr(38+marker_type);
5136:C        4           end_x := cpx - (marker_width div 2);
5137:C        4           end_y := cpy - (marker_height div 2);
5138:C        4           call ( move, gcb );
5139:C        4           info_ptr1 := addr(s);
5140:C        4           info1 := 1;
5141:C        4           call ( text, gcb );
5142:S
5143:C        4           { restore char state }
5144:S
5145:C        4           info1 := saved_char_width;
5146:C        4           info2 := saved_char_height;
5147:C        4           call ( char_size, gcb );
5148:C        4           info1 := saved_text_cos_dir;
5149:C        4           info2 := saved_text_sin_dir;
5150:C        4           call ( text_dir, gcb );
5151:C        4         end
5152:C        4       else
5153:C        4         for i := marker_index[marker_type] to marker_index[marker_type+1]-1 do
5154:C        5           with markers[i] do
5155:C        6             begin
5156:C        6               local_x_pos := x_pos; { move packed field to temp to force type }
5157:C        6               local_y_pos := y_pos; { to gle_shortint }
5158:C        6               dx := gle_ishift((local_x_pos * marker_width),marker_cell_shift) + cpx;
5159:C        6               dy := gle_ishift((local_y_pos * marker_height),marker_cell_shift) + cpy;
5160:C        6               end_x := dx;
5161:C        6               end_y := dy;
5162:C        6               if penup then
5163:C        7                 call ( move, gcb )
5164:C        7               else
5165:C        7                 call ( draw, gcb );
5166:C        6             end;
5167:C        3       info1 := saved_linestyle;
5168:C        3       info2 := saved_length;
5169:C        3       info3 := saved_mode;
5170:C        3       info4 := saved_line_pattern;
5171:C        3       call ( linestyle, gcb );
5172:C        3       end_x := cpx;           { restore cp }
5173:C        3       end_y := cpy;
5174:C        3       call ( move, gcb );
5175:C        3     end;
5176:C        2 end;
5177:S
5178:D        1 procedure gle_soft_set_marker ( gcb : graphics_control_block_ptr );
5179:S
5180:C        2 begin
5181:C        2   with gcb^ do
5182:C        3     marker_type := info1;
5183:C        2 end;
5184:S
5185:D        1 procedure gle_soft_marker_size ( gcb : graphics_control_block_ptr );
5186:S
5187:C        2 begin
5188:C        2   with gcb^ do
5189:C        3     begin
```

```
  5190:C        3       marker_width  := info1;
  5191:C        3       marker_height := info2;
  5192:C        3     end;
  5193:C        2 end;
  5194:S
  5195:C        1 end. { module gle_smark }
  5196:S

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

GLE_STEXT

Description

GLE_STEXT provides software text setup routines and the text size and rotation transformation.

Requirements

GLE_TYPES.

```
     1:D        0 {                                                                    }
     2:D        0 { Graphics Low End                                                   }
     3:D        0 {                                                                    }
     4:D        0 { Module     = GLE_STEXT                                             }
     5:D        0 { Programer = BJS                                                    }
     6:D        0 { Date       = 10-10-82                                              }
     7:D        0 {                                                                    }
     8:D        0 { Purpose: To provide software text routines.                        }
     9:S
    10:D        0 { Rev history                                                        }
    11:D        0 {  Created  - 10-10-82                                               }
    12:D        0 {  Modified -  6-28-83  BJS  Removed soft_text from export text.  This }
    13:D        0 {                            procedure is imported from gle_astext.  }
    14:D        0 {              12-08-83  BDS  Put Range Check on around calculation of }
    15:D        0 {                            dx and dy to eliminate random vectors.  }
    16:S
    17:S
    18:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    19:S                All rights are reserved.  Copying or other
    20:S                reproduction of this program except for archival
    21:S                purposes is prohibited without the prior
    22:S                written consent of Hewlett-Packard Company.
    23:S
    24:S
    25:S                            RESTRICTED RIGHTS LEGEND
    26:S
    27:S                Use, duplication, or disclosure by the Government
    28:S                is subject to restrictions as set forth in
    29:S                paragraph (b) (3) (B) of the Rights in Technical
    30:S                Data and Computer Software clause in
    31:S                DAR 7-104.9(a).
    32:S
    33:S                HEWLETT-PACKARD COMPANY
    34:D        0       Fort Collins, Colorado                                 }
    35:S
    36:D        0 $medcal$
    37:D        0 $search 'GLE_TYPES'$
    37:D        0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS ******************** }
    38:S
    39:S           { This include file specifies range checking, debug and other compiler
    40:D        0    options for the graphics library                                  }
    41:S
    42:D        0 $debug OFF$
    43:D        0 $range OFF$
    44:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    45:D        0 $FLOAT_HDW TEST$
    46:S
    47:S
    48:S
    49:S
    50:D        0 $include 'OPTIONS'$  { ******************** COMPILER OPTIONS ******************** }
  4000:D        0 $linenum 4000$
  4001:S
  4002:D        0 module gle_stext;
  4003:S
  4004:D        1 import gle_types.
  4005:S
  4006:D        1 export
  4007:S
  4008:D        1 var font ['GLE_GLE_STROKE_TABLE'] : array [0..maxint] of gle_shortint;
```



```
  4009:S
  4010:D        1 procedure gle_soft_char_size    ( gcb : graphics_control_block_ptr );
  4011:D        1 procedure gle_soft_text_spacing ( gcb : graphics_control_block_ptr );
  4012:D        1 procedure gle_soft_text_dir     ( gcb : graphics_control_block_ptr );
  4013:D        1 procedure gle_soft_text_just    ( gcb : graphics_control_block_ptr );
  4014:D        1 procedure gle_text_xform        ( gcb : graphics_control_block_ptr );
  4015:S
  4016:D        1 implement
  4017:S
  4018:D        1 procedure gle_text_xform  ( gcb : graphics_control_block_ptr);
  4019:S
  4020:D        2 const
  4021:D        2   fraction_adjust      = 32768;
  4022:D        2   res_adjust           = 8;
  4023:D        2   adjusted_cell_width  = 7 * fraction_adjust;
  4024:D        2   adjusted_cell_height = 9 * fraction_adjust;
  4025:S
  4026:D    -4  2 var i : integer;
  4027:D   -12  2     dx,dy : integer;
  4028:S
  4029:C        2 begin
  4030:C        2   with gcb^ do
  4031:C        3     begin
  4032:C        3       $RANGE ON$
  4033:C        3       for i := 0 to 15 do
  4034:C        4         begin
  4035:C        4           if i < 8 then
  4036:C        5             begin
  4037:C        5               dx := ((i-1) * char_width) div res_adjust;
  4038:C        5               cosx_table[i] := (dx * text_cos_dir)  div adjusted_cell_width;
  4039:C        5               sinx_table[i] := (dx * text_sin_dir)  div adjusted_cell_width;
  4040:C        5             end;
  4041:C        4           dy := ((i-4) * char_height) div res_adjust;
  4042:C        4           siny_table[i] := (-dy * text_sin_dir) div adjusted_cell_height;
  4043:C        4           cosy_table[i] := (dy * text_cos_dir)  div adjusted_cell_height;
  4044:C        4         end;
  4045:S
  4046:C        3       dx := (char_width + char_space);
  4047:C        3       text_space_x := ( dx * text_cos_dir ) div fraction_adjust;
  4048:C        3       text_space_y := ( dx * text_sin_dir ) div fraction_adjust;
  4049:S
  4050:C        3       dy := -(char_height + line_space);
  4051:C        3       text_line_x := ( -dy * text_sin_dir ) div fraction_adjust;
  4052:C        3       text_line_y := ( dy * text_cos_dir )  div fraction_adjust;
  4053:C        3       $RANGE OFF$                  { I added to avoid random vectors }
  4054:C        3       calc_text_xform := 0;  { transformation is not needed }
  4055:C        3     end;
  4056:C        2 end;
  4057:S
  4058:D        1 procedure gle_soft_char_size ( gcb : graphics_control_block_ptr);
  4059:S
  4060:C        2 begin
  4061:C        2   with gcb^ do
  4062:C        3     if (char_width <> info1) or (char_height <> info2) then
  4063:C        4       begin
  4064:C        4         calc_text_xform := 1;  { calc new transform }
  4065:C        4         char_width  := info1;
  4066:C        4         char_height := info2;
  4067:C        4       end;
  4068:C        2 end;
```

```
  4069:S
  4070:D        1 procedure gle_soft_text_spacing ( gcb : graphics_control_block_ptr);
  4071:S
  4072:C        2 begin
  4073:C        2   with gcb^ do
  4074:C        3     if (char_space <> info1) or (line_space <> info2) then
  4075:C        4       begin
  4076:C        4         calc_text_xform := 1;  { calc new transform }
  4077:C        4         char_space := info1;
  4078:C        4         line_space := info2;
  4079:C        4       end;
  4080:C        2 end;
  4081:S
  4082:D        1 procedure gle_soft_text_dir     ( gcb : graphics_control_block_ptr);
  4083:S
  4084:C        2 begin
  4085:C        2   with gcb^ do
  4086:C        3     if (text_cos_dir <> info1) or (text_sin_dir <> info2) then
  4087:C        4       begin
  4088:C        4         calc_text_xform := 1;  { calc new transform }
  4089:C        4         text_cos_dir := info1;
  4090:C        4         text_sin_dir := info2;
  4091:C        4       end;
  4092:C        2 end;
  4093:S
  4094:D        1 procedure gle_soft_text_just    ( gcb : graphics_control_block_ptr);
  4095:S
  4096:C        2 begin
  4097:C        2   with gcb^ do
  4098:C        3     if (char_just_x <> info1) or (char_just_y <> info2) then
  4099:C        4       begin
  4100:C        4         calc_text_xform := 1;  { calc new transform }
  4101:C        4         char_just_x := info1;
  4102:C        4         char_just_y := info2;
  4103:C        4       end;
  4104:C        2 end;
  4105:S
  4106:C        1 end. { module gle_stext }
  4107:S
  4108:S

No errors. No warnings.
          ***** Nonstandard language features enabled *****
```

GLE_TYPES

Description

GLE_TYPES defines the input and output graphics control blocks (GCBs) used by the GLE routines as well as the machine-type identifier tokens, and some globals for bit-mapped display use.

```
     1:S
     2:D        0 {                                                                    }
     3:D        0 { Graphics Low End                                                   }
     4:D        0 {                                                                    }
     5:D        0 { Module    = GLE_TYPES                                              }
     6:D        0 { Programer = BJS                                                    }
     7:D        0 { Date      = 10- 5-82                                               }
     8:D        0 {                                                                    }
     9:D        0 { Purpose: To define the graphics control blocks used by GLE.        }
    10:S
    11:D        0 { Rev history                                                        }
    12:D        0 {  Created  - 10- 5-82  BJS                                          }
    13:D        0 {  Modified - 11-29-82  BJS  Clean up and changed booleans to shortints }
    14:S
    15:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    16:S                All rights are reserved.  Copying or other
    17:S                reproduction of this program except for archival
    18:S                purposes is prohibited without the prior
    19:S                written consent of Hewlett-Packard Company.
    20:S
    21:S
    22:S                            RESTRICTED RIGHTS LEGEND
    23:S
    24:S                Use, duplication, or disclosure by the Government
    25:S                is subject to restrictions as set forth in
    26:S                paragraph (b) (3) (B) of the Rights in Technical
    27:S                Data and Computer Software clause in
    28:S                DAR 7-104.9(a).
    29:S
    30:S                HEWLETT-PACKARD COMPANY
    31:D        0       Fort Collins, Colorado                                 }
    32:S
    33:D        0 $tables on$
    34:D        0 $module$
    35:D        0 $include 'OPTIONS'$ { compiler options }
    36:S
    37:D          { This include file specifies range checking, debug and other compiler
    38:S        0   options for the graphics library                                   }
    39:D        0 $debug OFF$
    40:D        0 $range OFF$
    41:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    42:D        0 $FLOAT_HDW TEST$
    43:S
    44:S
    45:S
    46:S
    47:D        0 $include 'OPTIONS'$ { compiler options }
  1000:D        0 $linenum 1000$
  1001:S
  1002:D        0 module gle_types;
  1003:S
  1004:D        1 export
  1005:S
  1006:D        1 type gle_byte = -128 .. 127;
  1007:D        1      gle_shortint = -32768 .. 32767;
  1008:S
  1009:D        1      anychar = packed array [1..maxint] of char;
  1010:D        1      anychar_ptr = ^anychar;
  1011:S
```



```
  1012:D        1      gle_char6 = packed array [1..6] of char;
  1013:S
  1014:D        1      graphics_control_block_ptr = ^graphics_control_block;
  1015:D        1      graphics_input_control_block_ptr = ^graphics_input_control_block;
```

```
1016:D        1 $page$
1017:D        1 (****************************** OUTPUT ***************************************)
1018:S
1019:D        1       graphics_control_block =
1020:D        1         packed record
1021:D        1           info1 : integer;      { holds info passed to and from GLE }
1022:D        1           info2 : integer;
1023:D        1           info3 : integer;
1024:D        1           info4 : integer;
1025:S
1026:D        1           info_ptr1 : anyptr;
1027:D        1           info_ptr2 : anyptr;
1028:S
1029:D        1           await_blanking       : procedure ( gcb : graphics_control_block_ptr );
1030:D        1           buffer_mode          : procedure ( gcb : graphics_control_block_ptr );
1031:D        1           char_size            : procedure ( gcb : graphics_control_block_ptr );
1032:D        1           clear                : procedure ( gcb : graphics_control_block_ptr );
1033:D        1           clip_limits          : procedure ( gcb : graphics_control_block_ptr );
1034:D        1           cursor               : procedure ( gcb : graphics_control_block_ptr );
1035:D        1           define_color_map     : procedure ( gcb : graphics_control_block_ptr );
1036:D        1           define_drawing_mode  : procedure ( gcb : graphics_control_block_ptr );
1037:D        1           draw                 : procedure ( gcb : graphics_control_block_ptr );
1038:D        1           fill_index_color     : procedure ( gcb : graphics_control_block_ptr );
1039:D        1           flush_buffer         : procedure ( gcb : graphics_control_block_ptr );
1040:D        1           get_color_map        : procedure ( gcb : graphics_control_block_ptr );
1041:D        1           get_raster           : procedure ( gcb : graphics_control_block_ptr );
1042:D        1           get_polygon_info     : procedure ( gcb : graphics_control_block_ptr );
1043:D        1           gload                : procedure ( gcb : graphics_control_block_ptr );
1044:D        1           graphics_on_off      : procedure ( gcb : graphics_control_block_ptr );
1045:D        1           gstore               : procedure ( gcb : graphics_control_block_ptr );
1046:D        1           index_color          : procedure ( gcb : graphics_control_block_ptr );
1047:D        1           inq_plp2             : procedure ( gcb : graphics_control_block_ptr );
1048:D        1           linewidth            : procedure ( gcb : graphics_control_block_ptr );
1049:D        1           linestyle            : procedure ( gcb : graphics_control_block_ptr );
1050:D        1           marker               : procedure ( gcb : graphics_control_block_ptr );
1051:D        1           marker_size          : procedure ( gcb : graphics_control_block_ptr );
1052:D        1           move                 : procedure ( gcb : graphics_control_block_ptr );
1053:D        1           output_escapei       : procedure ( gcb : graphics_control_block_ptr );
1054:D        1           output_escapeo       : procedure ( gcb : graphics_control_block_ptr );
1055:D        1           polygon              : procedure ( gcb : graphics_control_block_ptr );
1056:D        1           set_marker           : procedure ( gcb : graphics_control_block_ptr );
1057:D        1           text                 : procedure ( gcb : graphics_control_block_ptr );
1058:D        1           text_dir             : procedure ( gcb : graphics_control_block_ptr );
1059:D        1           text_just            : procedure ( gcb : graphics_control_block_ptr );
1060:D        1           text_spacing         : procedure ( gcb : graphics_control_block_ptr );
1061:S
1062:D        1           dummy_xxx            : procedure ( gcb : graphics_control_block_ptr );
1063:S
1064:D        1           io_inq_timeout       : procedure ( anyvar iocb_ptr : anyptr; var value : integer);
1065:D        1           io_read              : procedure ( anyvar iocb_ptr, dev_buf_ptr : anyptr );
1066:D        1           io_set_timeout       : procedure ( anyvar iocb_ptr : anyptr; value : integer);
1067:D        1           io_term              : procedure ( anyvar iocb_ptr : anyptr );
1068:D        1           io_write             : procedure ( anyvar iocb_ptr, dev_buf_ptr : anyptr );
1069:S
1070:D        1           iocb          : anyptr;       { ptr to io system dependent info }
1071:D        1           device_buf    : anyptr;       { ptr to device dependent buffer  }
1072:D        1           dev_dep_stuff : anyptr;       { ptr to device dependent information }
1073:D        1           device_info   : anyptr;       { pointer to device identifier (e.g. 3) }
1074:D        1           device_info_char_count : gle_shortint; { # char in device_info }
1075:D        1           error_return  : gle_shortint; { used by some procedures }
```



```
1076:D        1           spooling      : gle_shortint; { '1' if output goes to a file }
1077:S
1078:D        1           display_name              : gle_char6;    { name of display device }
1079:D        1           display_name_char_count   : gle_shortint; { # chars used in display_name }
1080:D        1           display_handler_name      : gle_char6;    { 'HPGL', 'RASTER', 'GPIST' }
1081:D        1           display_handler_char_count : gle_shortint; { # chars used in display_handler_name }
1082:S
1083:D        1           display_res_x : real;         { points per mm in x direction }
1084:D        1           display_res_y : real;         { points per mm in y direction }
1085:D        1           display_min_x : integer;      { minimum x device coordinate }
1086:D        1           display_min_y : integer;      { minimum y device coordinate }
1087:D        1           display_max_x : integer;      { maximum x device coordinate }
1088:D        1           display_max_y : integer;      { maximum y device coordinate }
1089:S
1090:D        1           { general info stuff }
1091:D        1           background           : gle_shortint;  { '1' if drawing in color 0 supported, '0' if not }
1092:D        1           complement_support   : gle_shortint;  { '1' if complement drawing supported }
1093:D        1           non_dominant_support : gle_shortint;  { '1' if non-dominant drawing supported }
1094:D        1           erase_support        : gle_shortint;  { '1' if erase drawing supported }
1095:D        1           color_map_support    : gle_shortint;  { '1' if color map supported }
1096:D        1           polygon_support      : gle_shortint;  { '1' if polygons are supported }
1097:D        1           redef_background     : gle_shortint;  { '1' if color 0 can be changed }
1098:D        1           polygon_fill_factor  : gle_shortint;  { aprox size of a line in device units }
1099:D        1           polygon_solid_fill   : gle_shortint;  { fill line spacing which is used for doild fill }
1100:D        1           dither_support       : gle_shortint;  { '1' if polygons use dither }
1101:D        1           pallette             : integer;   { number of distinct colors supported  }
1102:D        1           gamut                : integer;   { number of colors which can appear at same time}
1103:D        1           cont_linestyles : gle_shortint;   { number of continuous linestyles supported }
1104:D        1           vect_linestyles : gle_shortint;   { number of vector adjusted linestyles supported }
1105:D        1           linewidths : gle_shortint;   { number of linewidths supported }
1106:S                    char_sizes : gle_shortint;   { number of character sizes supported,
1107:D        1                                          -1 for continuously varying sizes }
1108:S
1109:D        1           { current value information }
1110:D        1           current_pos_x : integer;      { last point used }
1111:D        1           current_pos_y : integer;
1112:D        1           end_x         : integer;
1113:D        1           end_y         : integer;
1114:S
1115:D        1           marker_type   : gle_shortint;
1116:D        1           marker_width  : integer;
1117:D        1           marker_height : integer;
1118:S
1119:D        1           kata          : gle_shortint; { '1' if kata char set to be used }
1120:S
1121:D        1           char_width    : integer;      { current character info }
1122:D        1           char_height   : integer;
1123:D        1           char_space    : integer;
1124:D        1           line_space    : integer;
1125:D        1           text_sin_dir  : integer;
1126:D        1           text_cos_dir  : integer;
1127:D        1           char_just_x   : integer;
1128:D        1           char_just_y   : integer;
1129:S
1130:D        1           clip_limits_xmin : integer;
1131:D        1           clip_limits_xmax : integer;
1132:D        1           clip_limits_ymin : integer;
1133:D        1           clip_limits_ymax : integer;
1134:S
1135:D        1           current_cursor_state : gle_shortint;  { off = 0; on = 1 }
```

```
1136:D  1          current_cursor_x     : integer;
1137:D  1          current_cursor_y     : integer;
1138:S
1139:D  1          current_buffer_mode  : gle_shortint;  { buffered = 1; unbuffered = 0 }
1140:S
1141:D  1          current_linestyle         : gle_shortint;
1142:D  1          current_linestyle_pattern : gle_shortint;
1143:D  1          current_pattern_length    : gle_shortint;
1144:D  1          current_linestyle_mode    : gle_shortint;
1145:S
1146:D  1          current_color_index       : gle_shortint;
1147:D  1          current_fill_index        : gle_shortint;
1148:D  1          current_drawing_mode      : gle_shortint;
1149:D  1          current_linewidth         : gle_shortint;
1150:S
1151:D  1          current_polygon_color     : gle_shortint;   { index, -1 if rgb }
1152:D  1          current_polygon_red       : gle_shortint;
1153:D  1          current_polygon_green     : gle_shortint;
1154:D  1          current_polygon_blue      : gle_shortint;
1155:S
1156:D  1          { temp storage of some regs }
1157:D  1          old_a5 : integer;
1158:D  1          old_a6 : integer;
1159:S
1160:D  1          { software text stuff }
1161:D  1          text_space_x : integer;  { spacing x&y after characters }
1162:D  1          text_space_y : integer;
1163:S
1164:D  1          text_line_x : integer;   { linefeed spacing }
1165:D  1          text_line_y : integer;
1166:D  1          cosx_table  : packed array [0..7]  of gle_shortint;
1167:D  1          cosy_table  : packed array [0..15] of gle_shortint;
1168:D  1          sinx_table  : packed array [0..7]  of gle_shortint;
1169:D  1          siny_table  : packed array [0..15] of gle_shortint;
1170:S
1171:D  1          calc_text_xform      : gle_shortint; { '1' if text xform needs to be recalculated }
1172:D  1          calc_soft_text_xform : procedure ( gcb : graphics_control_block_ptr );
1173:S
1174:D  1          soft_font_ptr   : anyptr;  { points to font table }
1175:S
1176:D  1          soft_text_temp1 : integer;
1177:D  1          soft_text_temp2 : integer;
1178:S
1179:D  1          { software clipping stuff }
1180:D  1          unclipped_move : procedure ( gcb : graphics_control_block_ptr );
1181:D  1          unclipped_draw : procedure ( gcb : graphics_control_block_ptr );
1182:S
1183:D  1          soft_clip_savex0 : integer;
1184:D  1          soft_clip_savex1 : integer;
1185:D  1          soft_clip_savey0 : integer;
1186:D  1          soft_clip_savey1 : integer;
1187:D  1          soft_clip_switch : gle_shortint;
1188:S
1189:D  1          soft_clip_cpx    : integer;
1190:D  1          soft_clip_cpy    : integer;
1191:D  1        end;
```



```
1192:D  1 $page$
1193:D  1 (***************************** INPUT ***************************************)
1194:S
1195:D  1      graphics_input_control_block =
1196:D  1        packed record
1197:D  1          info1 : integer;   { holds info passed to and from GLE }
1198:D  1          info2 : integer;
1199:D  1          info3 : integer;
1200:D  1          info4 : integer;
1201:S
1202:D  1          info_ptr1 : anyptr;
1203:S
1204:D  1          get_digitize      : procedure ( gcb : graphics_input_control_block_ptr );
1205:D  1          inq_pip2          : procedure ( gcb : graphics_input_control_block_ptr );
1206:D  1          input_echo        : procedure ( gcb : graphics_input_control_block_ptr );
1207:D  1          input_escapei     : procedure ( gcb : graphics_input_control_block_ptr );
1208:D  1          input_escapeo     : procedure ( gcb : graphics_input_control_block_ptr );
1209:D  1          sample            : procedure ( gcb : graphics_input_control_block_ptr );
1210:D  1          start_digitize    : procedure ( gcb : graphics_input_control_block_ptr );
1211:S
1212:D  1          dummy_xxx         : procedure ( gcb : graphics_input_control_block_ptr );
1213:S
1214:D  1          io_inq_timeout      : procedure ( anyvar iocb_ptr : anyptr; var value : integer);
1215:D  1          io_read             : procedure ( anyvar iocb_ptr, dev_buf_ptr : anyptr );
1216:D  1          io_set_timeout      : procedure ( anyvar iocb_ptr : anyptr; value : integer);
1217:D  1          io_term             : procedure ( anyvar iocb_ptr : anyptr );
1218:D  1          io_write            : procedure ( anyvar iocb_ptr, dev_buf_ptr : anyptr );
1219:S
1220:D  1          iocb          : anyptr;       { ptr to io system dependent info }
1221:D  1          device_buf    : anyptr;       { ptr to device dependent buffer  }
1222:D  1          dev_dep_stuff: anyptr;        { ptr to device dependent information }
1223:D  1          device_info   : anyptr;       { pointer to device identifier (e.g. 701) }
1224:D  1          device_info_char_count : gle_shortint; { # char in device_info }
1225:D  1          error_return : gle_shortint; { used by some procedures }
1226:S
1227:D  1          input_name              : gle_char6;    { name of display device }
1228:D  1          input_name_char_count   : gle_shortint; { # chars used in display_name }
1229:D  1          input_handler_name      : gle_char6;    { 'HPGL', 'KN0B', 'GPIS' }
1230:D  1          input_handler_char_count : gle_shortint; { # chars used in display_handler_name }
1231:S
1232:D  1          input_res_x : real;         { points per mm in x direction }
1233:D  1          input_res_y : real;         { points per mm in y direction }
1234:D  1          input_min_x : integer;      { input, min and max device coordinates }
1235:D  1          input_max_x : integer;
1236:D  1          input_min_y : integer;
1237:D  1          input_max_y : integer;
1238:S
1239:D  1          input_cpx : integer;        { Where device dependent echoes are started }
1240:D  1          input_cpy : integer;
1241:D  1        end;
1242:D  1
1243:D  1        status_def =
1244:D  1      packed record
1245:D  1        bit15,bit14,bit13,bit12,bit11,bit10,bit9,bit8,
1246:D  1        notbusy,bit6,vblank,bit4,hz50,monochrome,bit1,bit0 : boolean;
1247:D  1      end;
1248:S
1249:D  1        w_array     = packed array[1..maxint] of gle_shortint;
1250:S
1251:S
```

```
  1252:D        1 const
  1253:D        1   munknown  = 0;
  1254:D        1   m9816a    = 1;
  1255:D        1   m9826a    = 2;
  1256:D        1   m9836a    = 3;
  1257:D        1   m9836c    = 4;
  1258:D        1   m98627a   = 5;
  1259:D        1   m9837a    = 6;
  1260:D        1   mgator_c  = 7;
  1261:S
  1262:S
  1263:D        1   no  = 0;
  1264:D        1   yes = 1;
  1265:S
  1266:D        1 var
  1267:S
  1268:D    -4  1   rule_reg : ^gle_shortint;    { [HEX('564008')] : gle_shortint;}
  1269:D    -8  1   width_reg : ^gle_shortint;   { [HEX('56400C')] : gle_shortint;}
  1270:S            inq_buffer: ^gle_shortint;  { [HEX('564100')] : packed array [0..maxint] of
                                                        gle_shortint;}
  1271:D   -12  1
  1272:D   -16  1   crt  : ^w_array;            { [ HEX('566000') ] : word_array;}
  1273:D   -20  1   status : ^status_def;       { [ hex('564000') ] : status_def;}
  1274:D   -24  1   ww_reg : ^gle_shortint;     { [ hex('564108') ] : gle_shortint;}
  1275:D   -24  1   blink1_reg  [ hex('564100') ] : gle_shortint;
  1276:D   -24  1   blink2_reg [ hex('564104') ] : gle_shortint;
  1277:D   -25  1   write_to_status : boolean;
  1278:D   -30  1   control_space : integer;
  1279:S
  1280:D   -30  1 implement
  1281:S
```



```
  1282:D   -30  1 $page$

Dump of GLE_TYPES
Imported:

Exported:
 ANYCHAR          type
  array elbitsize=8 unpacksize=2147483647 align=2
 ANYCHAR_PTR      type
  pointer unpacksize=4 align=2
 BLINK1_REG       var lev= 1 addr=131072 long
 BLINK2_REG       var lev= 1 addr= 65578 long
 CONTROL_SPACE    var lev= 1 addr=   -30 globalbase = GLE_TYPES
 CRT              var lev= 1 addr=   -16 globalbase = GLE_TYPES
 GLE_BYTE         type
  subrange min=-128 max=127 unpacksize=2 align=2 bitsize=8 signed
 GLE_CHAR6        type
  array elbitsize=8 unpacksize=6 align=2
 GLE_SHORTINT     type
  subrange min=-32768 max=32767 unpacksize=2 align=2 bitsize=16 signed
 GLE_TYPES        module lev=1
 GRAPHICS_CONTROL_BLOCK type
  record unpacksize=730 align=2
    AWAIT_BLANKING   field offset=24
    BACKGROUND       field offset=398
    BUFFER_MODE      field offset=32
    CALC_SOFT_TEXT_XFORM field offset=668
    CALC_TEXT_XFORM  field offset=666
    CHAR_HEIGHT      field offset=466
    CHAR_JUST_X      field offset=486
    CHAR_JUST_Y      field offset=490
    CHAR_SIZE        field offset=40
    CHAR_SIZES       field offset=432
    CHAR_SPACE       field offset=470
    CHAR_WIDTH       field offset=462
    CLEAR            field offset=48
    CLIP_LIMITS      field offset=56
    CLIP_LIMITS_XMAX field offset=498
    CLIP_LIMITS_XMIN field offset=494
    CLIP_LIMITS_YMAX field offset=506
    CLIP_LIMITS_YMIN field offset=502
    COLOR_MAP_SUPPORT field offset=406
    COMPLEMENT_SUPPORT field offset=400
    CONT_LINESTYLES  field offset=426
    COSX_TABLE       field offset=570
    COSY_TABLE       field offset=586
    CURRENT_BUFFER_MODE field offset=520
    CURRENT_COLOR_INDEX field offset=530
    CURRENT_CURSOR_STATE field offset=510
    CURRENT_CURSOR_X field offset=512
    CURRENT_CURSOR_Y field offset=516
    CURRENT_DRAWING_MODE field offset=534
    CURRENT_FILL_INDEX field offset=532
    CURRENT_LINESTYLE field offset=522
    CURRENT_LINESTYLE_MODE field offset=528
    CURRENT_LINESTYLE_PATTERN field offset=524
    CURRENT_LINEWIDTH field offset=536
    CURRENT_PATTERN_LENGTH field offset=526
    CURRENT_POLYGON_BLUE field offset=544
    CURRENT_POLYGON_COLOR field offset=538
```

```
    CURRENT_POLYGON_GREEN field offset=542
    CURRENT_POLYGON_RED field offset=540
    CURRENT_POS_X       field offset=434
    CURRENT_POS_Y       field offset=438
    CURSOR              field offset=64
    DEFINE_COLOR_MAP field offset=72
    DEFINE_DRAWING_MODE field offset=80
    DEVICE_BUF          field offset=332
    DEVICE_INFO         field offset=340
    DEVICE_INFO_CHAR_COUNT field offset=344
    DEV_DEP_STUFF       field offset=336
    DISPLAY_HANDLER_CHAR_COUNT field offset=364
    DISPLAY_HANDLER_NAME field offset=358
    DISPLAY_MAX_X       field offset=390
    DISPLAY_MAX_Y       field offset=394
    DISPLAY_MIN_X       field offset=382
    DISPLAY_MIN_Y       field offset=386
    DISPLAY_NAME        field offset=350
    DISPLAY_NAME_CHAR_COUNT field offset=356
    DISPLAY_RES_X       field offset=366
    DISPLAY_RES_Y       field offset=374
    DITHER_SUPPORT      field offset=416
    DRAW                field offset=88
    DUMMY_XXX           field offset=280
    END_X               field offset=442
    END_Y               field offset=446
    ERASE_SUPPORT       field offset=404
    ERROR_RETURN        field offset=346
    FILL_INDEX_COLOR field offset=96
    FLUSH_BUFFER        field offset=104
    GAMUT               field offset=422
    GET_COLOR_MAP       field offset=112
    GET_POLYGON_INFO field offset=128
    GET_RASTER          field offset=120
    GLOAD               field offset=136
    GRAPHICS_ON_OFF     field offset=144
    GSTORE              field offset=152
    INDEX_COLOR         field offset=160
    INFO1               field offset=0
    INFO2               field offset=4
    INFO3               field offset=8
    INFO4               field offset=12
    INFO_PTR1           field offset=16
    INFO_PTR2           field offset=20
    INQ_M1P2            field offset=168
    IOCB                field offset=328
    IO_INQ_TIMEOUT      field offset=288
    IO_READ             field offset=296
    IO_SET_TIMEOUT      field offset=304
    IO_TERM             field offset=312
    IO_WRITE            field offset=320
    KATA                field offset=460
    LINESTYLE           field offset=184
    LINEWIDTH           field offset=176
    LINEWIDTHS          field offset=430
    LINE_SPACE          field offset=474
    MARKER              field offset=192
    MARKER_HEIGHT       field offset=456
    MARKER_SIZE         field offset=200
    MARKER_TYPE         field offset=450
```



```
    MARKER_WIDTH        field offset=452
    MOVE                field offset=208
    NON_DOMINANT_SUPPORT field offset=402
    OLD_A5              field offset=546
    OLD_A6              field offset=550
    OUTPUT_ESCAPEI      field offset=216
    OUTPUT_ESCAPEO      field offset=224
    PALETTE             field offset=418
    POLYGON             field offset=232
    POLYGON_FILL_FACTOR field offset=412
    POLYGON_SOLID_FILL field offset=414
    POLYGON_SUPPORT     field offset=408
    REDEF_BACKGROUND field offset=410
    SET_MARKER          field offset=240
    SINX_TABLE          field offset=618
    SINY_TABLE          field offset=634
    SOFT_CLIP_CPX       field offset=722
    SOFT_CLIP_CPY       field offset=726
    SOFT_CLIP_SAVEX0 field offset=704
    SOFT_CLIP_SAVEX1 field offset=708
    SOFT_CLIP_SAVEY0 field offset=712
    SOFT_CLIP_SAVEY1 field offset=716
    SOFT_CLIP_SWITCH field offset=720
    SOFT_FONT_PTR       field offset=676
    SOFT_TEXT_TEMP1     field offset=680
    SOFT_TEXT_TEMP2     field offset=684
    SPOOLING            field offset=348
    TEXT                field offset=248
    TEXT_COS_DIR        field offset=482
    TEXT_DIR            field offset=256
    TEXT_JUST           field offset=264
    TEXT_LINE_X         field offset=562
    TEXT_LINE_Y         field offset=566
    TEXT_SIN_DIR        field offset=478
    TEXT_SPACE_X        field offset=554
    TEXT_SPACE_Y        field offset=558
    TEXT_SPACING        field offset=272
    UNCLIPPED_DRAW      field offset=696
    UNCLIPPED_MOVE      field offset=688
    VECT_LINESTYLES     field offset=428
 GRAPHICS_CONTROL_BLOCK_PTR type
  pointer unpacksize=4 align=2
 GRAPHICS_INPUT_CONTROL_BLOCK type
  record unpacksize=200 align=2
    DEVICE_BUF          field offset=128
    DEVICE_INFO         field offset=136
    DEVICE_INFO_CHAR_COUNT field offset=140
    DEV_DEP_STUFF       field offset=132
    DUMMY_XXX           field offset=76
    ERROR_RETURN        field offset=142
    GET_DIGITIZE        field offset=20
    INFO1               field offset=0
    INFO2               field offset=4
    INFO3               field offset=8
    INFO4               field offset=12
    INFO_PTR1           field offset=16
    INPUT_CPX           field offset=192
    INPUT_CPY           field offset=196
    INPUT_ECHO          field offset=36
    INPUT_ESCAPEI       field offset=44
```

```
    INPUT_ESCAPEO     field offset=52
    INPUT_HANDLER_CHAR_COUNT field offset=158
    INPUT_HANDLER_NAME field offset=152
    INPUT_MAX_X       field offset=180
    INPUT_MAX_Y       field offset=188
    INPUT_MIN_X       field offset=176
    INPUT_MIN_Y       field offset=184
    INPUT_NAME        field offset=144
    INPUT_NAME_CHAR_COUNT field offset=150
    INPUT_RES_X       field offset=160
    INPUT_RES_Y       field offset=168
    INQ_PIP2          field offset=28
    IOCB              field offset=124
    IO_INQ_TIMEOUT    field offset=84
    IO_READ           field offset=92
    IO_SET_TIMEOUT    field offset=100
    IO_TERM           field offset=108
    IO_WRITE          field offset=116
    SAMPLE            field offset=60
    START_DIGITIZE    field offset=68
 GRAPHICS_INPUT_CONTROL_BLOCK_PTR type
  pointer unpacksize=4 align=2
 INQ_BUFFER        var lev= 1 addr=   -12 globalbase = GLE_TYPES
 M98I6A            konst 1
 M9826A            konst 2
 M9836A            konst 3
 M9836C            konst 4
 M9837A            konst 6
 M98627A           konst 5
 MGATOR_C          konst 7
 MUNKNOWN          konst 0
 NO                konst 0
 RULE_REG          var lev= 1 addr=    -4 globalbase = GLE_TYPES
 STATUS            var lev= 1 addr=   -20 globalbase = GLE_TYPES
 STATUS_DEF        type
  record unpacksize=2 align=2
    BIT0              field offset=0 bitoffset=15
    BIT1              field offset=0 bitoffset=14
    BIT10             field offset=0 bitoffset=5
    BIT11             field offset=0 bitoffset=4
    BIT12             field offset=0 bitoffset=3
    BIT13             field offset=0 bitoffset=2
    BIT14             field offset=0 bitoffset=1
    BIT15             field offset=0 bitoffset=0
    BIT4              field offset=0 bitoffset=11
    BIT6              field offset=0 bitoffset=9
    BIT8              field offset=0 bitoffset=7
    BIT9              field offset=0 bitoffset=6
    HZ50              field offset=0 bitoffset=12
    MONOCHROME        field offset=0 bitoffset=13
    NOTBUSY           field offset=0 bitoffset=8
    VBLANK            field offset=0 bitoffset=10
 WIDTH_REG         var lev= 1 addr=    -8 globalbase = GLE_TYPES
 WRITE_TO_STATUS   var lev= 1 addr=   -25 globalbase = GLE_TYPES
 WW_REG            var lev= 1 addr=   -24 globalbase = GLE_TYPES
 W_ARRAY           type
  array elbitsize=16 unpacksize=4 (OFLO) align=2
 YES               konst 1
```



```
GLE_TYPES dump complete

  1283:C        1 end. { GLE_TYPES }
  1284:D        1 $LIST ON$
  1285:S

No errors. No warnings.
              ***** Nonstandard language features enabled *****
```

GLE_UTLS

Description

GLE_UTLS provides general tools for GLE routines.

Requirements

GLE_TYPES.

```
     1:D        0 {                                                                                 }
     2:D        0 { Graphics Low End                                                                }
     3:D        0 {                                                                                 }
     4:D        0 { Module     = GLE_UTLS                                                           }
     5:D        0 { Programer  = BJS                                                                }
     6:D        0 { Date       = 11-15-82                                                           }
     7:D        0 {                                                                                 }
     8:D        0 { Purpose: To provide general GLE tools                                           }
     9:S
    10:D        0 { Rev history                                                                     }
    11:D        0 {  Created  - 11-15-82                                                            }
    12:D        0 {  Modified - XX-XX-XX                                                            }
    13:S
    14:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    15:S                All rights are reserved.  Copying or other
    16:S                reproduction of this program except for archival
    17:S                purposes is prohibited without the prior
    18:S                written consent of Hewlett-Packard Company.
    19:S
    20:S
    21:S                           RESTRICTED RIGHTS LEGEND
    22:S
    23:S                Use, duplication, or disclosure by the Government
    24:S                is subject to restrictions as set forth in
    25:S                paragraph (b) (3) (B) of the Rights in Technical
    26:S                Data and Computer Software clause in
    27:S                DAR 7-104.9(a).
    28:S
    29:S                HEWLETT-PACKARD COMPANY
    30:D        0       Fort Collins, Colorado                                  }
    31:S
    32:D        0 $search 'GLE_TYPES'$
    33:D        0 $sysprog$
    33:D        0 $include 'OPTIONS'$ { ****************** COMPILER OPTIONS ********************** }
    34:S
    35:S           { This include file specifies range checking, debug and other compiler
    36:D        0    options for the graphics library                                               }
    37:S
    38:D        0 $debug OFF$
    39:D        0 $range OFF$
    40:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    41:D        0 $FLOAT_HDW TEST$
    42:S
    43:S
    44:S
    45:S
    46:D        0 $include 'OPTIONS'$ { ****************** COMPILER OPTIONS ********************** }
 21000:D        0 $linenum 21000$
 21001:S
 21002:D        0 module gle_utls;
 21003:S
 21004:D        1 import gle_types;
 21005:S
 21006:D        1 export
 21007:S
 21008:D        1 function  gle_read_integer ( size : gle_shortint; s : anychar_ptr; var count : gle_shortint ) : gle_
shortint;
 21009:D        1 procedure gle_write_integer ( value : gle_shortint; var size : gle_shortint; s : anychar_ptr );
 21010:D        1 procedure gle_copy_to_string ( s : anychar_ptr; sl : gle_shortint; var d : string );
 21011:D        1 function  gle_match ( size1 : gle_shortint; s1 : anychar_ptr;
```



```
 21012:D        2                       size2 : gle_shortint; s2 : anychar_ptr ) : boolean;
 21013:D        1 function  gle_shortint_min ( p1,p2 : gle_shortint ) : gle_shortint;
 21014:D        1 function  gle_shortint_max ( p1,p2 : gle_shortint ) : gle_shortint;
 21015:S
```

```
 21016:D        1 $PAGE$
 21017:D        1 implement
 21018:S
 21019:D        1 function gle_shortint_min ( p1,p2 : gle_shortint ) : gle_shortint;
 21020:S
 21021:D        2 { Purpose : To return the minimum 16 bit integer                          }
 21022:S
 21023:C        2 begin
 21024:C        2   if p1 < p2 then gle_shortint_min := p1
 21025:C        3   else gle_shortint_min := p2;
 21026:C        2 end;  { gle_shortint_min }
 21027:S
 21028:D        1 function gle_shortint_max ( p1,p2 : gle_shortint ) : gle_shortint;
 21029:S
 21030:D        2 { Purpose : To return the maximum 16 bit integer                          }
 21031:S
 21032:C        2 begin
 21033:C        2   if p1 > p2 then gle_shortint_max := p1
 21034:C        3   else gle_shortint_max := p2;
 21035:C        2 end; { gle_shortint_max }
 21036:S
 21037:D        1 procedure gle_write_integer ( value : gle_shortint;
 21038:D        2                               var  size : gle_shortint;
 21039:D        2                                    s : anychar_ptr );
 21040:S
 21041:D        2 { Purpose : To convert a 16 bit signed integer to ASCII                   }
 21042:S
 21043:D        2 var
 21044:D   -20  2   t : packed array [1..20] of char;
 21045:D   -22  2   i : gle_shortint;
 21046:D   -24  2   temp_value : gle_shortint;
 21047:D   -26  2   digit : gle_shortint;
 21048:S
 21049:C        2 begin
 21050:C        2   temp_value := abs(value);
 21051:C        2   i := 0;
 21052:C        2   repeat
 21053:C        3     i := i + 1;
 21054:C        3     digit := temp_value mod 10;
 21055:C        3     t[i] := chr((digit) + 48);
 21056:C        3     temp_value := temp_value div 10;
 21057:C        3   until temp_value = 0;
 21058:C        2   if value < 0 then
 21059:C        3     begin
 21060:C        3       i := i + 1;
 21061:C        3       t[i] := '-';
 21062:C        3     end;
 21063:S
 21064:C        2   size := i;
 21065:C        2   for i := i downto 1 do
 21066:C        3     s^[size-i+1] := t[i];
 21067:C        2 end; { write_integer }
 21068:S
 21069:D        1 function gle_read_integer( size : gle_shortint; s : anychar_ptr; var count : gle_shortint ) : gle_sh
ortint;
 21070:S
 21071:D        2 { Purpose : To convert from ASCII to a 16 bit signed integer              }
 21072:S
 21073:D        2 var
 21074:D    -2  2   value : gle_shortint;
```



```
 21075:D    -3  2   neg   : boolean;
 21076:D    -6  2   digit : gle_shortint;
 21077:D    -8  2   i     : gle_shortint;
 21078:D   -10  2   start : gle_shortint;
 21079:S
 21080:C        2 begin
 21081:C        2   i := 1;
 21082:C        2   while (s^[i] = ' ') and ( i <= size  ) do i := i + 1;
 21083:C        2   if i > size then escape(-8);
 21084:S
 21085:C        2   if s^[i] = '-' then
 21086:C        3     begin
 21087:C        3       i := i + 1;
 21088:C        3       neg := true;
 21089:C        3     end
 21090:C        3   else
 21091:C        3     neg := false;
 21092:C        2   value := 0;
 21093:C        2   start := i;
 21094:C        2   while (s^[i] >= '0') and (s^[i] <= '9') and (i <= size) do
 21095:C        3     begin
 21096:C        3       digit := ord(s^[i]) - 48;
 21097:C        3       value := value * 10 + digit;
 21098:C        3       i := i + 1;
 21099:C        3     end;
 21100:C        2   if (i > size+1) or (i = start) then escape(-8);
 21101:C        2   count := i;  { next free byte }
 21102:C        2   if neg then value := -value;
 21103:C        2   gle_read_integer := value;
 21104:C        2 end; { read_integer }
 21105:S
 21106:D        1 procedure gle_copy_to_string ( s : anychar_ptr; sl : gle_shortint; var d : string );
 21107:S
 21108:D        2 { Purpose : To convert from a packed array of char to string format       }
 21109:S
 21110:D    -2  2 var i : gle_shortint;
 21111:S
 21112:C        2 begin
 21113:C        2   setstrlen(d,sl);
 21114:C        2   for i := 1 to sl do
 21115:C        3     d[i] := s^[i];
 21116:C        2 end; { copy_to_string }
 21117:S
 21118:D        1 function  gle_match ( size1 : gle_shortint; s1 : anychar_ptr;
 21119:D        2                       size2 : gle_shortint; s2 : anychar_ptr ) : boolean;
 21120:S
 21121:D        2 { Purpose : To return true if the two packed array of char match          }
 21122:S
 21123:D    -2  2 var i : gle_shortint;
 21124:S
 21125:C        2 begin
 21126:C        2   if size1 = size2 then
 21127:C        3     begin
 21128:C        3       i := 1;
 21129:C        3     $partial_eval on$
 21130:C        3       while (i <= size1) and (s1^[i] = s2^[i]) do i := i + 1;
 21131:C        3       gle_match := i > size1;
 21132:C        3     end
 21133:C        3   else gle_match := false;
 21134:C        2 end; { match }
```

```
21135:S
21136:C         1 end.
21137:S

No errors. No warnings.
              ***** Nonstandard language features enabled *****
```

Description

H_DRV provides low-level driver support.

Usage

H_DRV is used by the 98624 and built-in HP-IB interfaces.

Requirements

SYSGLOBALS and IODECLARATIONS.

```
1:S                                                     (*
2:S
3:S           (c) Copyright Hewlett-Packard Company, 1983.
4:S          All rights are reserved.  Copying or other
5:S          reproduction of this program except for archival
6:S          purposes is prohibited without the prior
7:S          written consent of Hewlett-Packard Company.
8:S
9:S
10:S                       RESTRICTED RIGHTS LEGEND
11:S
12:S         Use, duplication, or disclosure by the Government
13:S         is subject to restrictions as set forth in
14:S         paragraph (b) (3) (B) of the Rights in Technical
15:S         Data and Computer Software clause in
16:S         DAR 7-104.9(a).
17:S
18:S         HEWLETT-PACKARD COMPANY
19:D       0 Fort Collins, Colorado                         *)
20:S
21:S
22:D       0 $MODCAL ON$
23:D       0 $PARTIAL_EVAL ON$
24:D       0 $STACKCHECK ON$
25:D       0 $RANGE OFF$
26:D       0 $DEBUG OFF$
27:D       0 $OVFLCHECK OFF$
```



```
28:D       0 $PAGE$
29:S
30:D       0 (*****************************************************************************)
31:D       0 (*                                                                           *)
32:D       0 (*      not RELEASED        VERSION         3.0                              *)
33:D       0 (*                                                                           *)
34:D       0 (*****************************************************************************)
35:D       0 (*                                                                           *)
36:D       0 (*                                                                           *)
37:D       0 (*      IOLIB           HPIB_DRIVERS                                         *)
38:D       0 (*                                                                           *)
39:D       0 (*                                                                           *)
40:D       0 (*****************************************************************************)
41:D       0 (*                                                                           *)
42:D       0 (*                                                                           *)
43:D       0 (*      library       -  IOLIB                                               *)
44:D       0 (*      name          -  HPIB_DRIVERS                                        *)
45:D       0 (*      module(s)     -  h_drv                                               *)
46:D       0 (*                    -  hpib                                                *)
47:D       0 (*                                                                           *)
48:D       0 (*      author        -                                                      *)
49:D       0 (*      phone         -                                                      *)
50:D       0 (*                                                                           *)
51:D       0 (*      date          -  June  1 , 1981                                      *)
52:D       0 (*      update        -  Aug   1 , 1983                                      *)
53:D       0 (*      release       -  ????????????                                        *)
54:D       0 (*                                                                           *)
55:D       0 (*      source        -  IOLIB:H_DRV.TEXT                                    *)
56:D       0 (*      object        -  IOLIB:H_DRV.CODE                                    *)
57:D       0 (*                                                                           *)
58:D       0 (*****************************************************************************)
```

```
    59:D        0 $PAGE$
    60:D        0 (*************************************************************************)
    61:D        0 (*                                                                       *)
    62:D        0 (*                                                                       *)
    63:D        0 (*      BUG FIX HISTORY         - after release                          *)
    64:D        0 (*                                                                       *)
    65:D        0 (*                                                                       *)
    66:D        0 (*      BUG #   BY  / ON        LOC             DESCRIPTION              *)
    67:D        0 (*      -----   ----------      --------------  ----------------------   *)
    68:D        0 (*                                                                       *)
    69:D        0 (*      367     ----------      hpib_initialize Allow eXecute of driver  *)
    70:D        0 (*              09/22/82                        and have it install      *)
    71:D        0 (*                                              itself in the system.    *)
    72:D        0 (*                                                                       *)
    73:D        0 (*************************************************************************)
    74:S
```



```
    75:D        0 $PAGE$
    76:D        0 (*************************************************************************)
    77:D        0 (*                                                                       *)
    78:D        0 (*                                                                       *)
    79:D        0 (*      This is the source code for an external procedures library       *)
    80:D        0 (*      to be used for general purpose interfacing on the HP 9826.       *)
    81:D        0 (*                                                                       *)
    82:D        0 (*      The library consists of 3 primary sets of modules -              *)
    83:D        0 (*                                                                       *)
    84:D        0 (*              1.      KERNEL modules                                   *)
    85:D        0 (*              2.      driver modules                                   *)
    86:D        0 (*              3.      IOLIB  modules                                   *)
    87:D        0 (*                                                                       *)
    88:D        0 (*      The KERNEL modules consist of the following modules -            *)
    89:D        0 (*                                                                       *)
    90:D        0 (*              1.      iodeclarations  ( contains static r/w space )    *)
    91:D        0 (*              2.      iocomasm                                         *)
    92:D        0 (*              3.      general_0       ( initialization & low level    *)
    93:D        0 (*                                        routines like ioread/iowrite) *)
    94:D        0 (*      The KERNEL modules also have an executable program segement      *)
    95:D        0 (*      that gets executed at the time it is loaded.  This program       *)
    96:D        0 (*      initializes the static read/write memory.  This program also     *)
    97:D        0 (*      allocates the temporary storage for any card that exists -       *)
    98:D        0 (*      independent of whether there is or is not a driver for it.       *)
    99:D        0 (*                                                                       *)
   100:D        0 (*      The driver modules consist of the actual assembly or PASCAL      *)
   101:D        0 (*      routines that deal with a specific interface card.  There is     *)
   102:D        0 (*      also an executable program segment for each driver module.       *)
   103:D        0 (*      This program searches the select code table in the static r/w    *)
   104:D        0 (*      initialized by the KERNEL general_0 module for all select codes *)
   105:D        0 (*      that have the right interface card ( HPIB drivers will search    *)
   106:D        0 (*      for the 98624 interface ).  This program will then set up the    *)
   107:D        0 (*      driver tables to point to the correct drivers.                   *)
   108:D        0 (*                                                                       *)
   109:D        0 (*      The rest of the IOLIB modules are high-level modules that are    *)
   110:D        0 (*      used by an end user in his/her application program.              *)
   111:D        0 (*                                                                       *)
   112:D        0 (*      The KERNEL and some set of driver modules will exist in the      *)
   113:D        0 (*      SYSTEM.INITLIB file as object code ( not EXPORT text ).  The     *)
   114:D        0 (*      export text will reside on the SYSTEM.LIBRARY file.  The rest    *)
   115:D        0 (*      of the library will reside on the SYSTEM.LIBRARY.                *)
   116:D        0 (*                                                                       *)
   117:D        0 (*************************************************************************)
```

```
118:D    0 $PAGE$
119:D    0 (*************************************************************************)
120:D    0 (*                                                                       *)
121:D    0 (*      REFERENCES :                                                     *)
122:D    0 (*                                                                       *)
123:D    0 (*                                                                       *)
124:D    0 (*      1.  9826 I/O Designers Guide          ( ---------- )           *)
125:D    0 (*                                                                       *)
126:D    0 (*      2.  68000 Manual                      ( Motorola )             *)
127:D    0 (*                                                                       *)
128:D    0 (*      3.  Pascal alpha site ERS             (---------- )            *)
129:D    0 (*                                                                       *)
130:D    0 (*      4.  Pascal I/O Library ERS            ( ---------- )           *)
131:D    0 (*                                                                       *)
132:D    0 (*      5.  9826 HPL EIO & IOD listings       ( ---------- )           *)
133:D    0 (*                                                                       *)
134:D    0 (*      6.  9826 HPL Misc. I/O Doc.           ( ---------- )           *)
135:D    0 (*                                                                       *)
136:D    0 (*      7.  9826 card documentation           ( Mfg. Specs. )          *)
137:D    0 (*                                                                       *)
138:D    0 (*      8.  Pascal I/O Library IRS            ( ---------- )           *)
139:D    0 (*                                                                       *)
140:D    0 (*                                                                       *)
141:D    0 (*************************************************************************)
142:D    0
```



```
143:D    0 $PAGE$
144:D    0 PROGRAM hpib_initialize ( INPUT , OUTPUT );
145:S
146:S
147:S              { This module has a program segment so that there is
148:S                an executable entry point into the module.
149:D    1           At INITLIB time this program is executed. }
```

```
150:D        1 $PAGE$
151:D        1 EXTERNAL MODULE exth;
152:S                  {
153:S                    date    08/25/81
154:S                    update  10/02/81
155:S
156:S                    purpose This module is a declaration of the importation text for
157:S                            the external drivers.
158:S
159:S                    note    The assembly language code that is imported needs to be
160:S                            called 'exth'.  The routines need to be called
161:S                            'exth_@@@@@' - eh_init referenced below would be
162:S                            exth_eh_init.  'eh' refers to external HP-IB.
163:D        1         }
164:S
165:D        1 IMPORT  $SEARCH 'IOLIB:KERNEL.CODE'$
166:D        1         sysglobals , iodeclarations ;
167:S
168:D        1 EXPORT
169:S
170:D        1   PROCEDURE eh_init   ( temp : ANYPTR );
171:D        1   PROCEDURE eh_isr    ( temp : PISRIB );
172:D        1   PROCEDURE eh_rdb    ( temp : ANYPTR ;  VAR x : CHAR);
173:D        1   PROCEDURE eh_wtb    ( temp : ANYPTR ;  val   : CHAR);
174:D        1   PROCEDURE eh_rdw    ( temp : ANYPTR ;  VAR x : io_word);
175:D        1   PROCEDURE eh_wtw    ( temp : ANYPTR ;  val   : io_word);
176:D        1   PROCEDURE eh_rds    ( temp : ANYPTR ;  reg   : io_word;
177:D        2                                           VAR x : io_word);
178:D        1   PROCEDURE eh_wtc    ( temp : ANYPTR ;  reg   : io_word;
179:D        2                                           val   : io_word );
180:D        1   PROCEDURE eh_tfr    ( temp : ANYPTR ;  bcb   : ANYPTR );
181:D        1   PROCEDURE eh_send   ( temp : ANYPTR ;  val   : CHAR );
182:D        1   PROCEDURE eh_ppoll  ( temp : ANYPTR ;  VAR x : CHAR );
183:D        1   PROCEDURE eh_clr    ( temp : ANYPTR ;  line  : io_bit );
184:D        1   PROCEDURE eh_set    ( temp : ANYPTR ;  line  : io_bit );
185:D        1   PROCEDURE eh_test   ( temp : ANYPTR ;  line  : io_bit ;
186:D        2                                           VAR x : BOOLEAN );
187:D        1   PROCEDURE eh_end    ( temp : ANYPTR ;  VAR x : BOOLEAN );
188:S
189:D        1 END; { of exth }
```



```
190:D        1 $PAGE$
191:S
192:S
193:D        1 MODULE init_hpib;
194:S                  {
195:S                    date    08/25/81
196:S                    update  12/16/82
197:S                             8/01/83   JS  change rev number
198:S
199:S                    purpose This module initializes the HPIB drivers.
200:S
201:D        1         }
202:S
203:D        1 IMPORT    iodeclarations ;
204:S
205:S
206:D        1 EXPORT
207:D        1   VAR
208:D  -120  1     hpib_drivers  : drv_table_type;
209:S
210:S
211:D  -120  1   PROCEDURE io_init_hpib;
212:S
213:S
214:D  -120  1 IMPLEMENT
215:S
216:D  -120  1   IMPORT  sysglobals ,
217:D  -120  1           isr ,
218:D  -120  1           general_0 ,
219:D  -120  1           exth ;
220:S
221:S
222:D        1   PROCEDURE io_init_hpib;
223:D        2   VAR
224:D    -2  2     io_isc        : type_isc;
225:D    -6  2     dummy         : INTEGER;
226:D    -8  2     io_lvl        : io_byte;
227:C        2   BEGIN
228:S
229:C        2     io_revid := io_revid + ' H3.0';     { HPIB revision added 2/5/82 TM }
230:S
231:C        2     { set up the driver tables }
232:S
233:C        2     WITH hpib_drivers DO BEGIN
234:C        3       hpib_drivers := dummy_drivers;
235:C        3       iod_init  := eh_init;
236:C        3       iod_isr   := eh_isr;
237:C        3       iod_rdb   := eh_rdb;
238:C        3       iod_wtb   := eh_wtb;
239:C        3       iod_rdw   := eh_rdw;
240:C        3       iod_wtw   := eh_wtw;
241:C        3       iod_rds   := eh_rds;
242:C        3       iod_wtc   := eh_wtc;
243:C        3       iod_end   := eh_end;
244:C        3       iod_tfr   := eh_tfr;
245:C        3       iod_send  := eh_send;
246:C        3       iod_ppoll := eh_ppoll;
247:C        3       iod_set   := eh_set;
248:C        3       iod_clr   := eh_clr;
249:C        3       iod_test  := eh_test;
```

```
250:C        3     END; { of WITH }
251:S
252:S
253:C        2     { set up drivers for the interfaces }
254:S
255:C        2     FOR io_isc:=iominisc TO iomaxisc DO
256:C        3       WITH isc_table[io_isc] DO BEGIN
257:S
258:C        4         IF ( card_id = hp98624 ) OR
259:C        5            ( card_id = internal_hpib)
260:C        5           THEN BEGIN
261:S
262:C        5             io_drv_ptr:=ADDR(hpib_drivers);
263:S
264:C        5             WITH io_drv_ptr^, io_tmp_ptr^ DO BEGIN
265:C        6               IF io_isc=7
266:C        7                 THEN BEGIN
267:C        7                   io_lvl := 3;
268:C        7                 END
269:C        7                 ELSE BEGIN
270:C        7                   io_lvl:={(ioread_byte(io_isc,3) DIV 16) MOD 4)+3;
271:C        7                 END; { of IF }
272:S
273:C        6               { if the card exists then link in an ISR for it }
274:C        6               { ??? - what happens if an ISR fires during init }
275:S
276:C        6               IF myisrib.INTREGADDR <> NIL
277:C        7                 THEN BEGIN
278:C        7                   { remove any existant isr }
279:C        7                   ISRUNLINK(io_lvl,                { interrupt level      }
280:C        7                             addr(myisrib));        { ptr to the isrib     }
281:C        7                 END; { of IF }
282:S
283:C        6               PERMISRLINK(iod_isr,                               { isr           }
284:C        6                           ANYPTR(INTEGER(card_ptr)+3),           { int reg addr  }
285:C        6                           192,                                   { int reg mask  }
286:C        6                           192,                                   { int reg value }
287:C        6                           io_lvl,                                { int level     }
288:C        6                           ADDR(myisrib));                        { isrib info    }
289:C        6             END; { of WITH // DO BEGIN }
290:S
291:C        5           END; { of IF card_id }
292:S
293:C        4       END; { of FOR io_isc WITH isc_table[io_isc] BEGIN }
294:S
295:S
296:C        2     { call the actual driver initialization }
297:S
298:S               { this is seperate from the set up stuff in case
299:S                 there are 2 or more cards connected - and generate
300:C        2        an isr }
301:S
302:C        2     FOR io_isc:=iominisc TO iomaxisc DO
303:C        3       WITH isc_table[io_isc] DO
304:C        4       IF ( card_id = hp98624 ) OR ( card_id = internal_hpib )
305:C        5         THEN BEGIN
306:C        5           CALL(io_drv_ptr^.iod_init , io_tmp_ptr);
307:C        5         END; { of WITH IF }
308:S
309:S
```



```
310:C        2   END; { of io_init_hpib }
311:S
312:C        1 END; { of MODULE init_hpib }
```

```
   313:D        1 $PAGE$
   314:S
   315:D        1 IMPORT    init_hpib ,
   316:D        1           LOADER ;                                    { 367 TM 9/22/82 }
   317:S
   318:S
   319:C        1 BEGIN
   320:C        1   io_init_hpib;
   321:C        1   MARKUSER;                                           { 367 TM 9/22/82 }
   322:C        1 END. { of hpib_initialize }

No errors. No warnings.
             ***** Nonstandard language features enabled *****
```

Description

HEAPT supports the Pascal procedures: NEW, DISPOSE, MARK and RELEASE.

Usage

The Compiler emits calls to the routines when the $HEAP_DISPOSE$ directive has been turned on.

```
     1:S                                                    (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1983.
     4:S          All rights are reserved.  Copying or other
     5:S          reproduction of this program except for archival
     6:S          purposes is prohibited without the prior
     7:S          written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                        RESTRICTED RIGHTS LEGEND
    11:S
    12:S          Use, duplication, or disclosure by the Government
    13:S          is subject to restrictions as set forth in
    14:S          paragraph (b) (3) (B) of the Rights in Technical
    15:S          Data and Computer Software clause in
    16:S          DAR 7-104.9(a).
    17:S
    18:S          HEWLETT-PACKARD COMPANY
    19:D        0 Fort Collins, Colorado                          *)
    20:S
    21:S
    22:D        0 (* 3/2/83 - 3:32 pm  *)
    23:D        0 $modcal,debug off,range off$
    24:D        0 $stackcheck off$              (* 2-Mar-83 -- 3:32 pm *)
    25:S
    26:D        0 external module sysglobals;
    27:D        1   export
    28:D        1     var
    29:D    -2  1       escapeco: -32768..32767;
    30:D    -6  1       FIBptr: integer;
    31:D   -10  1       tryrec: integer;
    32:D   -14  1       heapmax: integer;
    33:D   -18  1       heapbase: integer;
    34:D   -22  1       ioresult: integer;
    35:D        1   end;
    36:S
    37:D        1 module hpm;
    38:S
    39:D        1 import
    40:D        1    heapmax,       { integer/anyptr }
    41:D        1    heapbase;      { integer/anyptr }
    42:S
    43:D        1 export
    44:S
    45:D        1   procedure Hestablish;
    46:D        1   procedure New (var object:integer; bytesize:integer);
    47:D        1   procedure Dispose (var object:integer; bytesize:integer);
    48:D        1   procedure Mark (var nextfreeword:integer);
    49:D        1   procedure Release (nextfreeword:integer);
    50:S
    51:D        1 implement
    52:D        1   const
    53:D        1     nilptr = 0;             {integer equivalent of nil pointer}
    54:D        1     maxptr = 16777215;      {biggest physical address}
    55:D        1     smallest = 2;           {smallest allocatable object (words)}
    56:D        1     stdblk = 16;            {largest "standard" object}
    57:D        1     otherlist = stdblk+1;   {for nonstandard objects}
    58:D        1   type
    59:D        1     stdlists = smallest..stdblk;
    60:D        1     allists = smallest..otherlist;
```



```
    61:D        1     pointer = nilptr..maxptr;
    62:D        1     pointertrick =  packed record
    63:D        1                       case boolean of
    64:D        1                     true:  ( longform: integer );
    65:D        1                     false: ( highbyte: 0..255;
    66:D        1                              lower3: pointer  )
    67:D        1                   end;
    68:D        1     mp = ^memblock;
    69:D        1     memblock = packed record
    70:D        1                  smallsize: 0..255;
    71:D        1                  link: pointer;
    72:D        1                  case  (smallsize=0)  boolean of
    73:D        1                 true: ( bigsize: integer )
    74:D        1                 end;
    75:D        1     freearray =  array [allists] of pointer;
    76:D        1     dsap = ^dsa;
    77:D        1     dsa = record
    78:D        1             fakeone: memblock;
    79:D        1             freelist: freearray;
    80:D        1             wordsdisposed: integer;
    81:D        1             dirty: boolean;
    82:D        1           end;
    83:S          {
    84:S          procedure Newwords (var p:anyptr; wordlen:integer);  external;
    85:D        1 }
    86:D        1 function longpointer (p:pointer): integer;
    87:D    -4  2 var  t: pointertrick;
    88:C        2 begin
    89:C        2   t.lower3 := p;   t.highbyte := 255;
    90:C        2   longpointer := t.longform
    91:C        2 end;
    92:S
    93:D        1 function shortpointer (p:integer): pointer;
    94:D    -4  2 var  t: pointertrick;
    95:C        2 begin
    96:C        2   t.longform := p;  shortpointer := t.lower3
    97:C        2 end;
    98:S
    99:D        1 function getsize (p:pointer): integer;
   100:D    -4  2 var  size: integer;
   101:C        2 begin
   102:C        2   with mp(p)^ do
   103:C        3     begin
   104:C        3       size := mp(p)^.smallsize;
   105:C        3       if size = 0 then size := bigsize;
   106:C        3       getsize := size;
   107:C        3     end;
   108:C        2 end;
   109:S
   110:D        1 procedure putsize (p:pointer; size:integer);
   111:C        2 begin
   112:C        2   with mp(p)^ do
   113:C        3     if size > otherlist then
   114:C        4       begin  smallsize := 0; bigsize := size  end
   115:C        4     else
   116:C        4       smallsize := size;
   117:C        2 end;
   118:S
   119:D        1 procedure Hestablish;
   120:D        2 const
```

```
121:D        2     emptydsa =
122:D        2       dsa [ fakeone:memblock[smallsize:(smallest-1),link:nilptr],
123:D        2             freelist:freearray[(otherlist-smallest+1) of nilptr],
124:D        2             wordsdisposed:0, dirty:false ];
125:D        2   var
126:D    -2  2     i: allists;
127:C        2   begin
128:C        2     Newwords(anyptr(heapbase),(sizeof(dsa)+1) div 2);
129:C        2     dsap(heapbase)^ := emptydsa;
130:C        2   end;  {Hestablish}
131:S
132:D        1   procedure recombine;
133:D        2   label 1,2;
134:D        2   var
135:D   -28  2     fakeaddr,f,g,this,thislink,flimit,highmark: pointer;
136:D   -42  2     fsize,gsize,wordcount: integer;  i: allists;
137:C        2   begin
138:C        2     with dsap(heapbase)^ do
139:C        3       begin
140:C        3         highmark := shortpointer(heapmax);
141:C        3         fakeaddr := shortpointer(integer(addr(fakeone)));
142:C        3         fakeone.link := nilptr;
143:C        3         fakeone.smallsize := smallest-1;
144:C        3         for i := smallest to otherlist do
145:C        4           while freelist[i] <> nilptr do
146:C        5             begin
147:C        5               f := freelist[i];
148:C        5               freelist[i] := mp(f)^.link;
149:C        5               this := fakeaddr;
150:C        5               thislink := mp(this)^.link;
151:C        5               while thislink <> nilptr do
152:C        6                 if thislink > f then goto 2
153:C        7                 else
154:C        7                   begin
155:C        7                     this := thislink;
156:C        7                     thislink := mp(this)^.link;
157:C        7                   end;
158:C        5 2:              mp(f)^.link := thislink;
159:C        5               mp(this)^.link := f;
160:C        5             end;
161:C        3         f := fakeone.link;  wordcount := 0;
162:C        3         while f <> nilptr do
163:C        4           begin
164:C        4             if highmark < f then goto 1;
165:C        4             fsize := getsize(f);
166:C        4             flimit := f+fsize+fsize;
167:C        4             while mp(f)^.link = flimit do {adjacent}
168:C        5               begin
169:C        5                 g := mp(f)^.link;
170:C        5                 mp(f)^.link := mp(g)^.link;
171:C        5                 gsize := getsize(g);
172:C        5                 flimit := g+gsize+gsize;
173:C        5               end; {while}
174:C        4             if highmark <= flimit then
175:C        5               begin
176:C        5                 heapmax := longpointer(f);
177:C        5                 goto 1;
178:C        5               end;
179:C        4             fsize := (flimit-f) div 2;
180:C        4             wordcount := wordcount+fsize;
```



```
181:C        4             putsize(f,fsize);
182:C        4             if fsize > otherlist then i:=otherlist
183:C        5                                  else i:=fsize;
184:C        4             g := mp(f)^.link;
185:C        4             mp(f)^.link := freelist[i];
186:C        4             freelist[i] := f;
187:C        4             f := g;
188:C        4           end; {f<>nil}
189:C        3 1:      dirty := false;  wordsdisposed := wordcount;
190:C        3       end; {with heapbase}
191:C        2   end;  {recombine}
192:S
193:D        1   procedure hreturn (p:pointer; wordlen:integer);
194:D    -2  2   var  list: allists;
195:C        2   begin
196:C        2     with dsap(heapbase)^,mp(p)^ do
197:C        3       begin
198:C        3         if wordlen < 0
199:C        4           then wordlen := -wordlen
200:C        4           else wordsdisposed := wordsdisposed+wordlen;
201:C        3         dirty := true;
202:C        3         if wordlen > otherlist then list := otherlist
203:C        4                                else list := wordlen;
204:C        3         putsize(p,wordlen);
205:C        3         link := freelist[list];
206:C        3         freelist[list] := p;
207:C        3       end; {with}
208:C        2   end;  {hreturn}
209:S
210:D        1   procedure hallocate (var p:pointer; wordlen:integer);
211:D    -4  2   var  longp: anyptr;
212:S
213:D        2     function tryit: pointer;
214:D        3     label 1,2;
215:D        3     var
216:D   -12  3       temp,prev,next: pointer;
217:D   -28  3       lfrag,j,len,size: integer;
218:C        3     begin
219:C        3       with dsap(heapbase)^ do
220:C        4         begin  len := wordlen;
221:C        4           if len < otherlist then
222:C        5             begin
223:C        5               temp := freelist[len];
224:C        5               if temp <> nilptr then
225:C        6                 begin
226:C        6                   freelist[len] := mp(temp)^.link;
227:C        6                   goto 1
228:C        6                 end;
229:C        5             end;
230:C        4           lfrag := len+smallest;
231:C        4           for size := lfrag to stdblk do
232:C        5             begin
233:C        5               temp := freelist[size];
234:C        5               if temp <> nilptr then
235:C        6                 begin
236:C        6                   freelist[size] := mp(temp)^.link;
237:C        6                   hreturn(temp+len+len,len-size);
238:C        6                   goto 1
239:C        6                 end;
240:C        5             end;
```

```
241:C        4           prev := nilptr;  next := freelist[otherlist];
242:C        4           while next <> nilptr do
243:C        5             begin
244:C        5               size := getsize(next);
245:C        5               if (size=len) or (size>=1frag) then goto 2;
246:C        5               prev := next; next := mp(next)^.link;
247:C        5             end;
248:C        4 2:        temp := next;
249:C        4           if next <> nilptr then
250:C        5             begin
251:C        5               if prev = nilptr
252:C        6                 then freelist[otherlist] := mp(next)^.link
253:C        6                 else mp(prev)^.link:=mp(next)^.link;
254:C        5               if size <> len then
255:C        6                 hreturn(next+len+len,len-size);
256:C        5             end;
257:C        4 1:        if temp <> nilptr then
258:C        5             wordsdisposed := wordsdisposed-wordlen;
259:C        4           tryit := temp;
260:C        4         end;  {with heapbase}
261:C        3     end;  {tryit}
262:S
263:C        2   begin {hallocate}
264:C        2     p := tryit;
265:C        2     if (p=nilptr) and dsap(heapbase)^.dirty then
266:C        3       begin  recombine;  p := tryit  end;
267:C        2     if p = nilptr then
268:C        3       begin
269:C        3         Newwords(longp,wordlen);
270:C        3         p := shortpointer(integer(longp));
271:C        3       end;
272:C        2   end;  {hallocate}
273:S
274:D        1   procedure Mark (var nextfreeword:pointer);
275:C        2     begin
276:C        2       nextfreeword := heapmax;
277:C        2     end;
278:S
279:D        1   procedure Release (nextfreeword:integer);
280:C        2     begin
281:C        2       heapmax := nextfreeword;
282:C        2       if heapbase <> nilptr then recombine;
283:C        2     end;
284:S
285:D        1   procedure New (var object:integer; bytesize:integer);
286:D    -8  2   var  ptr: pointer;  wordsize: integer;
287:C        2   begin
288:C        2     wordsize := (bytesize+1) div 2;
289:C        2     if heapbase = nilptr then
290:C        3       Newwords(anyptr(object),wordsize)
291:C        3     else
292:C        3       begin
293:C        3         if wordsize < smallest then wordsize := smallest;
294:C        3         if dsap(heapbase)^.wordsdisposed < wordsize then
295:C        4           Newwords(anyptr(object),wordsize)
296:C        4         else
297:C        4           begin
298:C        4             hallocate(ptr,wordsize);
299:C        4             object := longpointer(ptr);
300:C        4           end;
```



```
301:C        3       end;
302:C        2   end;  {New}
303:S
304:D        1   procedure Dispose (var object:integer; bytesize:integer);
305:D    -8  2   var  ptr: pointer;  wordsize: integer;
306:C        2   begin
307:C        2     if heapbase <> nilptr then
308:C        3       begin
309:C        3         wordsize := (bytesize+1) div  2;
310:C        3         if wordsize < smallest then wordsize := smallest;
311:C        3         ptr:=shortpointer(object);
312:C        3         if ptr = nilptr then escape(-3);
313:C        3         hreturn(ptr,wordsize)
314:C        3       end;
315:C        2     object := nilptr;
316:C        2   end;  {Dispose}
317:S
318:C        1 end.  {heapmanager}
319:S

No errors. No warnings.
          ***** Nonstandard language features enabled *****
```

HPHIL

Description

HPHIL provides driver routines for the Hewlett-Packard Human Interface Link (HP-HIL).

Requirements

A804XDVR, SYSGLOBALS, SYSDEVS, ISR, and ASM.

```
Pascal [Rev 3.0M  6/ 4/84] HPHIL.TEXT              26-Dec-84 21:54:10 Page 1   *** File name: HPHIL ***

     1:S                                                    (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1984.
     4:S          All rights are reserved.  Copying or other
     5:S          reproduction of this program except for archival
     6:S          purposes is prohibited without the prior
     7:S          written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                       RESTRICTED RIGHTS LEGEND
    11:S
    12:S          Use, duplication, or disclosure by the Government
    13:S          is subject to restrictions as set forth in
    14:S          paragraph (b) (3) (B) of the Rights in Technical
    15:S          Data and Computer Software clause in
    16:S          DAR 7-104.9(a).
    17:S
    18:S          HEWLETT-PACKARD COMPANY
    19:D        0 Fort Collins, Colorado                        *)
    20:S
    21:D        0 $sysprog$
    22:D        0 $stackcheck off$ $heap_dispose off$
    23:D        0 $iocheck off$ $range off$ $ovflcheck off$
    24:D        0 $debug off$
    25:S
    26:D        0 program loopinit;
    27:S
    28:D        1 module hphil; { HP Human Interface Loop }
    29:S
    30:D        1 import a804xdvr,sysdevs,sysglobals,asm;
    31:S
    32:D        1 export
    33:D        1   FUNCTION INITLOOP:BOOLEAN;     {4/6/84 SFB}
    34:S
    35:D        1 implement
    36:D        1 const
    37:D        1   readctrl = hex('FB'); { 8042 COMMANDS }
    38:D        1   writectrl= hex('EB');
    39:D        1   startcmd = hex('E0');
    40:D        1   readbusy = hex('02');
    41:D        1   trigger  = hex('C5');
    42:S
    43:S
    44:D        1 type
    45:D        1   stat5datatype = packed record { data bye with status 5 }
    46:D        1                     case integer of
    47:D        1                     0     :(b:byte);
    48:D        1                     1     :(lerror  : boolean;          {7}
    49:D        1                       padding : 0..3;           {6..5}
    50:D        1                       polling : boolean;          {4}
    51:D        1                       command : boolean;          {3}
    52:D        1                       devaddr : 0..7);          {2..0}
    53:D        1                   end;
    54:S
    55:D        1   lpctrltype = packed record    { loop control byte }
    56:D        1                   case boolean of
    57:D        1                   true :(b:byte);
    58:D        1                   false:(doreconfigure: boolean;
    59:D        1                          pad6,pad5    : boolean;
    60:D        1                          cookkbd      : boolean;
```

```
Pascal [Rev 3.0M  6/ 4/84] HPHIL.TEXT              26-Dec-84 21:54:10 Page 2   *** File name: HPHIL ***

    61:D        1                          pad3         : boolean;
    62:D        1                          noreconfig   : boolean;
    63:D        1                          noerrs       : boolean;
    64:D        1                          doautopoll   : boolean )
    65:D        1                 end;
    66:S
    67:D        1   var
    68:D     -8 1   oldstatus5isr : kbdhooktype;
    69:D    -16 1   oldclrio      : procedure;
    70:S
    71:D        1   procedure dummydataproc(var statbyte,databyte:byte; var done:boolean);
    72:C        2   begin end;
    73:D        1   procedure dummyopsproc(op:loopdvrop);
    74:C        2   begin end;
    75:S
    76:D        1   procedure lcommand;
    77:D        2   const
    78:D        2     timeout20= hex('FE');
    79:C        2   begin
    80:C        2     WITH LOOPCONTROL^ DO        {4/6/84 SFB}
    81:C        3      if loopisok or loopinconfig then
    82:C        4      begin
    83:C        4        sendcmd(startcmd);
    84:C        4        senddata(8+loopdevice);   { command & address }
    85:C        4        senddata(loopcmd);        { id command }
    86:C        4        senddata(timeout20);      { 20 milli secs }
    87:C        4        loopcmddone:=false; looperror:=false;
    88:C        4        sendcmd(trigger);
    89:C        4        repeat call(kbdpollhook,true);
    90:C        5        until loopcmddone or looperror;
    91:C        4      end
    92:C        4      else looperror:=true;
    93:C        2   end;  { lcommand }
    94:S
    95:D    -16 1   { status6 isr for describe operation }
    96:D        1   procedure describedata(var statbyte,databyte:byte; var done:boolean);
    97:C        2   begin { load describe record for the current loop device }
    98:C        2    WITH LOOPCONTROL^ DO                {4/10/84 SFB}
    99:C        3     with loopdevices[loopdevice] do     {4/6/84 SFB}
   100:C        4     case devstate of
   101:C        5     1: begin descrip.darray[1]:=chr(databyte); devstate:=2; end;
   102:C        5     2: begin
   103:C        5          descrip.darray[2]:=chr(databyte);      { header byte }
   104:C        5          if descrip.numaxes>0 then devstate:=3
   105:C        6                               else devstate:=11;
   106:C        5        end;                                     { low order count }
   107:C        5     3: begin descrip.darray[4]:=chr(databyte); devstate:=4; end;
   108:C        5     4: begin
   109:C        5          descrip.darray[3]:=chr(databyte);      { high order count }
   110:C        5          if descrip.abscoords then devstate:=5
   111:C        6                               else devstate:=11;
   112:C        5        end;
   113:C        5     5..10:begin { swap even / odd bytes }
   114:C        5             descrip.darray[devstate-1+(ord(odd(devstate))*2)]:=chr(databyte);
   115:C        5             devstate:=devstate+1+ord(not descrip.size16);
   116:C        5             if (devstate>6) and (descrip.numaxes=1) then devstate:=11
   117:C        6             else
   118:C        6             if (devstate>8) and (descrip.numaxes=2) then devstate:=11;
   119:C        5           end;
   120:C        5     11: begin   { prompts & buttons }
```

```
   121:C        5           if descrip.hasprompts then descrip.darray[11]:=chr(databyte);
   122:C        5           devstate:=12;
   123:C        5         end;
   124:C        5     otherwise
   125:C        5     end; { case devstate }
   126:C        2   end;  { describedata }
   127:S
   128:S
   129:D        1   procedure rawshift;    { change 8042 to raw mode }
   130:D        2   var
   131:D    -4  2     oldpriority : integer;
   132:D    -6  2     ctrlreg     : lpctrltype;
   133:D    -8  2     tempstuff   : byte;
   134:C        2   begin
   135:C        2    WITH LOOPCONTROL^ DO          {4/6/84 SFB}
   136:C        3     if not rawmode then
   137:C        4     begin
   138:C        4       oldpriority:=intlevel;    { lock out interupts }
   139:C        4       setintlevel(7);
   140:C        4       cmd_read_1(readctrl,tempstuff); ctrlreg.b:=tempstuff;
   141:S
   142:C        4       ctrlreg.doautopoll:=false;
   143:C        4       ctrlreg.noerrs:=true;
   144:C        4       ctrlreg.noreconfig:=false;
   145:C        4       sendcmd(writectrl);
   146:C        4       senddata(ord(ctrlreg.b));
   147:S
   148:C        4       repeat  { wait for polling to stop }
   149:C        5         cmd_read_1(readbusy,loopdata);
   150:C        5       until not odd(loopdata div 4);
   151:C        4       rawmode:=true;
   152:C        4       setintlevel(oldpriority); { restore interupts }
   153:C        4     end;
   154:C        2   end;  { rawshift }
   155:S
   156:D        1   procedure normshift; { change 8042 to normal mode }
   157:D        2   var
   158:D    -4  2     oldpriority : integer;
   159:D    -6  2     ctrlreg     : lpctrltype;
   160:D    -8  2     tempstuff   : byte;
   161:C        2   begin
   162:C        2    WITH LOOPCONTROL^ DO          {4/6/84 SFB}
   163:C        3     if rawmode then
   164:C        4     begin
   165:C        4       oldpriority:=intlevel;    { lock out interupts }
   166:C        4       setintlevel(7);
   167:C        4       cmd_read_1(readctrl,tempstuff); ctrlreg.b:=tempstuff;
   168:C        4       ctrlreg.doautopoll:=true;
   169:C        4       ctrlreg.noerrs:=true;
   170:C        4       ctrlreg.noreconfig:=false;
   171:S
   172:C        4       sendcmd(writectrl); senddata(ord(ctrlreg.b)); { start auto polling }
   173:S
   174:C        4       rawmode := false;
   175:C        4       setintlevel(oldpriority);
   176:C        4     end;
   177:C        2   end;  { normshift }
   178:S
   179:D        1   procedure describe;   { issue describe for loopdevice }
   180:D        2   const
```



```
   181:D        2     idcmd    = hex('03');
   182:S
   183:D        2   var
   184:D    -4  2     oldpriority : integer;
   185:D    -6  2     i           : 1..11;
   186:C        2   begin { describe }
   187:C        2    oldpriority := intlevel;    { lockout interupts }
   188:C        2    setintlevel(7);
   189:C        2    status6hook:=describedata;
   190:C        2    WITH LOOPCONTROL^ DO          {4/10/84 SFB}
   191:C        3     with loopdevices[loopdevice] do
   192:C        4      begin
   193:C        4       devstate:=1;
   194:C        4       opsproc:=dummyopsproc;
   195:C        4       dataproc:=dummydataproc;
   196:C        4       for i:=1 to 11 do descrip.darray[i]:=chr(0); { clear the record }
   197:C        4       loopcmd:=idcmd;
   198:C        4       lcommand; { send the command and wait for it to finish }
   199:C        4      end;
   200:C        2    status6hook:=dummydataproc;
   201:C        2    setintlevel(oldpriority);
   202:C        2   end; { describe }
   203:S
   204:D        1   procedure checkloop;
   205:D        2   const
   206:D        2     readlstat = hex('FA');
   207:D        2   var
   208:D    -4  2     oldpriority : integer;
   209:C        2   begin
   210:C        2    WITH LOOPCONTROL^ DO          {4/6/84 SFB}
   211:C        3     BEGIN
   212:C        3      loopisok:=false;
   213:C        3      LOOPDATA := 0;              {4/24/84 SFB }
   214:C        3      oldpriority:=intlevel;    { lock out interupts }
   215:C        3      setintlevel(7);
   216:C        3      cmd_read_1(readlstat,loopdata);
   217:C        3      loopisok:=loopdata<128;
   218:C        3      setintlevel(oldpriority);
   219:C        3     END;                         {4/6/84 SFB}
   220:C        2   end;  { checkloop }
   221:S
   222:D        1   procedure configure;
   223:D        2   label 1;
   224:D        2   CONST
   225:D        2     READ_INTR_MASK = 4; { SFB 3/5/84 }
   226:D        2   var
   227:D    -4  2     temp : loopdvrptr;
   228:D    -6  2     i    : shortint;
   229:D    -8  2     OLDMASK : BYTE;     { SFB 3/5/84 }
   230:C        2   begin { configure }
   231:C        2    WITH LOOPCONTROL^ DO          {4/6/84 SFB}
   232:C        3     BEGIN
   233:C        3      CMD_READ_1(READ_INTR_MASK,OLDMASK);{ SAVE OLD 804X MASK - SFB 3/5/84 }
   234:C        3      CALL(MASKOPSHOOK,KBDMASK,0);      { ENABLE LOOP - SFB 3/5/84 }
   235:C        3      loopdevreading:=false;            { 3.0 BUG #39  3/17/84 }
   236:C        3      checkloop;
   237:C        3      if not loopisok then goto 1;
   238:C        3      rawshift;
   239:C        3      loopinconfig:=true;
   240:C        3      { get device description for all (7) devices }
```

```
   241:C        3      for i:=1 to 7 do
   242:C        4      begin
   243:C        4        loopdevice:=i;
   244:C        4        describe;
   245:C        4        if looperror then goto 1; { kick out & wait for status5, le_configured }
   246:C        4      end;
   247:S
   248:C        3      { find and attach drivers to devices }
   249:C        3      for i:=1 to 7 do
   250:C        4      begin
   251:C        4        loopdevice:=i;
   252:C        4        temp:=loopdriverlist;
   253:C        4        while temp<>nil do
   254:C        5        with loopdevices[loopdevice] do
   255:C        6        begin
   256:C        6          if (temp^.daddr=0) or (temp^.daddr=loopdevice) then
   257:C        7            if (descrip.id>=temp^.lowid) and (descrip.id<=temp^.highid) then
   258:C        8            begin   { hook up the driver }
   259:C        8              opsproc:=temp^.opsproc;
   260:C        8              dataproc:=temp^.dataproc;
   261:C        8              call(opsproc,resetdevice);    { tell driver to reset }
   262:C        8              temp:=nil;    { cancel the search }
   263:C        8            end
   264:C        8            else temp:=temp^.next
   265:C        8          else temp:=temp^.next;    { 3.0 BUG FIX  4/6/84 }
   266:C        6        end; { while .. with }
   267:C        4      end;   { for }
   268:S
   269:C        3    1:loopinconfig:=false;
   270:C        3      normshift;
   271:C        3      IF ODD(OLDMASK) THEN
   272:S                 CALL(MASKOPSHOOK,0,KBDMASK);      { DISABLE LOOP IF IT WAS PREVIOUSLY
   273:C        3                                             OFF - SFB 3/5/84 }
   274:C        3    END;                            {4/6/84 SFB}
   275:C        2    end;  { configure }
   276:S
   277:D        1    procedure resettheloop; { clearIO call }
   278:C        2     begin
   279:C        2      configure; call(oldclrio);
   280:C        2    end;
   281:S
   282:D        1    procedure endcmdproc(var statbyte,databyte:byte; var done:boolean);
   283:C        2    begin
   284:C        2      WITH LOOPCONTROL^ DO        {4/6/84 SFB}
   285:C        3       BEGIN
   286:C        3        loopcmddone := true; status6hook:=dummydataproc;
   287:C        3        loopdata    := databyte;
   288:C        3       END;
   289:C        2    end;
   290:S
   291:D        1    PROCEDURE SENDHPHILCMD(OP : HPHILOP);         {4/9/84 SFB}
   292:C        2    BEGIN
   293:C        2     CASE OP OF
   294:C        3      RAWSHIFTOP  : RAWSHIFT;
   295:C        3      NORMSHIFTOP : NORMSHIFT;
   296:C        3      CHECKLOOPOP : CHECKLOOP;
   297:C        3      CONFIGUREOP : CONFIGURE;
   298:C        3      LCOMMANDOP  : LCOMMAND;
   299:C        3     END;
   300:C        2    END;
```



```
   301:S
   302:D        1    procedure status5isr(var statbyte,databyte:byte; var done:boolean);
   303:D    -2  2    var status : stat5datatype;
   304:C        2    begin
   305:C        2     status.b:=databyte;
   306:C        2     WITH LOOPCONTROL^ DO        {4/6/84 SFB}
   307:C        3      if status.lerror then
   308:C        4       begin
   309:C        4        looperror:=true;
   310:C        4        status6hook:=dummydataproc;
   311:C        4        if status.b=le_loopdown then
   312:C        5         loopisok:=false
   313:C        5        else
   314:C        5         if status.b=le_configured then
   315:C        6          begin     { loop has been reconfigured }
   316:C        6           call(oldstatus5isr,statbyte,databyte,done); { let keyboard proc know }
   317:C        6           if not loopinconfig then configure;
   318:C        6           loopdevreading:=false;              { 3.0 bug #39   3/17/84 }
   319:C        6          end;
   320:C        4       end
   321:C        4      else
   322:C        4       if status.command then
   323:C        5        begin
   324:C        5         if status.polling then    { end of polled data }
   325:C        6          begin
   326:C        6           status6hook:=dummydataproc;     { ignore next data }
   327:C        6           call(loopdevices[loopdevice].opsproc,dataended);
   328:C        6           loopdevreading:=false;                { 3.0 bug #39  3/17/84 }
   329:C        6          end
   330:C        6         else
   331:C        6          begin     { command ending }
   332:C        6           status6hook:=endcmdproc;
   333:C        6          end;
   334:C        5        end
   335:C        5       else
   336:C        5        if status.polling then
   337:C        6         begin       { polling data starting }
   338:C        6          { check if device now reading        3.0 bug #39   3/17/84}
   339:C        6          if loopdevreading then                                {3.0 bug #39 3/17/84}
   340:C        7           call(loopdevices[loopdevice].opsproc, dataended){3.0 bug #39 3/17/84}
   341:C        7          else                                               {3.0 bug #39 3/17/84}
   342:C        7            loopdevreading:=true;                            {3.0 bug #39 3/17/84}
   343:C        6          loopdevice:=status.devaddr;
   344:C        6          status6hook:=loopdevices[loopdevice].dataproc;
   345:C        6          call(loopdevices[loopdevice].opsproc,datastarting);
   346:C        6         end
   347:C        6        else
   348:C        6         begin       { data starting for current command }
   349:C        6          looperror:= loopdevice<>status.devaddr;
   350:C        6         end;
   351:C        2    end;  { status5isr }
   352:S
   353:D        1    function initloop:boolean;
   354:C        2    begin
   355:C        2      IF LOOPCONTROL = NIL THEN     {4/9/84 SFB}
   356:C        3       BEGIN
   357:C        3        NEW(LOOPCONTROL);           {4/9/84 SFB}
   358:C        3        WITH LOOPCONTROL^ DO        {4/6/84 SFB}
   359:C        4         BEGIN
   360:C        4          initloop:=false;
```

```
   361:C        4          rawmode :=false;
   362:C        4          loopisok :=false;
   363:S
   364:C        4          if kbdtype=itfkbd then
   365:C        5           begin
   366:C        5            oldstatus5isr:=status5hook;       { save old hook }
   367:C        5            status5hook:=status5isr;
   368:C        5            status6hook:=dummydataproc;
   369:C        5            oldclrio    :=cleariohook;        { save old hook }
   370:C        5            cleariohook:=resettheloop;
   371:C        5            HPHILCMDHOOK:=SENDHPHILCMD;     {4/9/84 SFB}
   372:C        5            configure;
   373:C        5            initloop := true;
   374:C        5           end;
   375:C        4         END;      {WITH LOOPCONTROL^}
   376:C        3       END         {IF LOOPCONTROL = NIL}
   377:C        3       ELSE
   378:C        3        BEGIN                          {4/9/84 SFB}
   379:C        3         CONFIGURE;                    {IF INITLOOP ALREADY DONE JUST CONFIGURE}
   380:C        3         INITLOOP := FALSE;            {DON'T MARKUSER}
   381:C        3        END;
   382:C        2    end;  { initloop }
   383:S
   384:C        1 end;    { loopinterface  module }
   385:S
   386:D        1 import hphil, loader;
   387:S
   388:C        1 begin
   389:C        1   if initloop then markuser;
   390:C        1 end.
   391:S

No errors. No warnings.
             ***** Nonstandard language features enabled *****
```

Description

INIT contains the following:

- ISR: Installs and removes interrupt service handlers using ISRIBs.
- MISC: Miscellaneous support utilities and DAM for unblocked devices.
- FS: File support routines. The IMPLEMENT portion of the listing has been suppressed due to the proprietary nature of the software.
- INITUNITS: Creates the unit table.
- LDR: Interface to Loader and miscellaneous support.
- SETUPSYS: The initialization.

Usage

INIT is used by the operating system (OS) kernel (in INITLIB).

Many routines in FS are called directly by compiler generated code.

Notes

Includes: ISR, MISC, FS, INITUNITS, LDR and SETUPSYS.

```
     1:S                                                    (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1983.
     4:S           All rights are reserved.  Copying or other
     5:S           reproduction of this program except for archival
     6:S           purposes is prohibited without the prior
     7:S           written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                       RESTRICTED RIGHTS LEGEND
    11:S
    12:S           Use, duplication, or disclosure by the Government
    13:S           is subject to restrictions as set forth in
    14:S           paragraph (b) (3) (B) of the Rights in Technical
    15:S           Data and Computer Software clause in
    16:S           DAR 7-104.9(a).
    17:S
    18:S           HEWLETT-PACKARD COMPANY
    19:D         0 Fort Collins, Colorado                        *)
    20:S
    21:S
    22:D         0 $modcal$
    23:D         0 $heap_dispose off$
    24:D         0 $iocheck off$
    25:D         0 $range off$ $ovflcheck off$
    26:D         0 $debug off$
    27:D         0 $stackcheck off$
    28:D         0 $search  'INITLOAD', 'ASM'$
    29:S
    30:D         0 program setupsys(input, output,  keyboard, listing);
    31:S
    31:D         1   $include 'ISR'$;
    32:D         1 module isr;
    33:S
    34:D         1 import sysglobals,asm;
    35:S
    36:D         1 export
    37:S
    38:D         1 procedure isrlink(procentry    : isrproctype;
    39:D         2                   lintregaddr  : charptr;
    40:D         2                   lintregmask  : byte;
    41:D         2                   lintregvalue : byte;
    42:D         2                   lintlevel    : byte;
    43:D         2                   isribp       : pisrib);
    44:S
    45:D         1 procedure permisrlink(procentry    : isrproctype;
    46:D         2                       lintregaddr  : charptr;
    47:D         2                       lintregmask  : byte;
    48:D         2                       lintregvalue : byte;
    49:D         2                       lintlevel    : byte;
    50:D         2                       isribp       : pisrib);
    51:S
    52:D         1 procedure isrunlink(lintlevel : byte;
    53:D         2                     isribp    : pisrib);
    54:S
    55:D         1 procedure isrchange(procentry : isrproctype;
    56:D         2                     isribp    : pisrib);
    57:S
    58:D         1 implement
    59:S
```



```
    60:D         1 procedure isrlink(procentry    : isrproctype;
    61:D         2                   lintregaddr  : charptr;
    62:D         2                   lintregmask  : byte;
    63:D         2                   lintregvalue : byte;
    64:D         2                   lintlevel    : byte;
    65:D         2                   isribp       : pisrib);
    66:C         2 begin
    67:C         2   with isribp^ do
    68:C         3     begin
    69:C         3       intregaddr  := lintregaddr;
    70:C         3       intregmask  := lintregmask;
    71:C         3       intregvalue := lintregvalue;
    72:C         3       proc        := procentry;
    73:C         3       link        := interrupttable[lintlevel];
    74:C         3     end;
    75:C         2   interrupttable[lintlevel]:=isribp;
    76:C         2 end;
    77:S
    78:D         1 procedure permisrlink(procentry    : isrproctype;
    79:D         2                       lintregaddr  : charptr;
    80:D         2                       lintregmask  : byte;
    81:D         2                       lintregvalue : byte;
    82:D         2                       lintlevel    : byte;
    83:D         2                       isribp       : pisrib);
    84:D         2 var
    85:D    -4   2   nxtlink : pisrib;
    86:D    -8   2   plinkp  : ^pisrib;
    87:S
    88:C         2 begin
    89:C         2   nxtlink := interrupttable[lintlevel];
    90:C         2   plinkp  := addr(interrupttable[lintlevel]);
    91:S
    92:C         2   while nxtlink <> perminttable[lintlevel] do
    93:C         3     begin
    94:C         3       plinkp  := addr(nxtlink^.link);
    95:C         3       nxtlink := nxtlink^.link;
    96:C         3     end;
    97:S
    98:C         2   with isribp^ do
    99:C         3     begin
   100:C         3       intregaddr  := lintregaddr;
   101:C         3       intregmask  := lintregmask;
   102:C         3       intregvalue := lintregvalue;
   103:C         3       proc        := procentry;
   104:C         3       link        := nxtlink;
   105:C         3     end;
   106:S
   107:C         2   plinkp^                := isribp;
   108:C         2   perminttable[lintlevel] := isribp;
   109:C         2 end;
   110:S
   111:D         1 procedure isrunlink(lintlevel : byte;
   112:D         2                     isribp    : pisrib);
   113:D         2 var
   114:D    -4   2   nxtlink : pisrib;
   115:D    -8   2   plinkp  : ^pisrib;
   116:S
   117:C         2 begin
   118:C         2   nxtlink := interrupttable[lintlevel];
   119:C         2   plinkp  := addr(interrupttable[lintlevel]);
```

```
   120:S
   121:C        2   while  (nxtlink <> nil) and (nxtlink <> isribp) do
   122:C        3     begin
   123:C        3       plinkp  := addr(nxtlink^.link);
   124:C        3       nxtlink := nxtlink^.link;
   125:C        3     end;
   126:S
   127:C        2   if nxtlink <> nil then
   128:C        3     begin
   129:C        3       if perminttable[lintlevel] = nxtlirk then
   130:C        4         perminttable[lintlevel] := nxtlirk^.link;
   131:S
   132:C        3       plinkp^ := nxtlink^.link;
   133:C        3     end;
   134:C        2 end;
   135:S
   136:D        1 procedure isrchange(procentry : isrproctype;
   137:D        2                     isribp    : pisrib);
   138:D        2 var
   139:D    -4  2   saveintlevel:integer;
   140:S
   141:C        2 begin
   142:C        2   saveintlevel:=intlevel;
   143:C        2   setintlevel(7); (disable all interrupts)
   144:C        2   with isribp^ do   proc:=procentry;
   145:C        2   setintlevel(saveintlevel);
   146:C        2 end;
   147:S
   148:C        1 end   (module isr)
   149:S
   150:C        2   $include 'ISR'$;
   150:D        1   $include 'MISC'$;
   151:D        1 module misc;    (homeless orphans)
   152:S
   153:D        1 import sysglobals, asm;
   154:S
   155:D        1 export
   156:S
   157:D        1 const
   158:S
   159:D        1   null  = 0;    nullchar  = chr(0);
   160:D        1                 homechar  = chr(1);
   161:D        1   etx   = 3;
   162:D        1   bell  = 7;    bellchar  = chr(7);
   163:D        1   bs    = 8;    leftchar  = chr(8);
   164:D        1   tab   = 9;    cteol     = chr(9);
   165:D        1   lf    = 10;   downchar  = chr(10);
   166:D        1   vt    = 11;   cteos     = chr(11);
   167:D        1   ff    = 12;   clearscr  = chr(12);
   168:D        1   cr    = 13;   eol       = chr(13);
   169:D        1   dle   = 16;
   170:D        1   esc   = 27;   escchar   = chr(27);
   171:D        1   fsp   = 28;   rightchar = chr(28);
   172:D        1   us    = 31;   upchar    = chr(31);
   173:D        1   del   = 127;
   174:D        1   cntrl = 255;
   175:S
   176:D        1 type
   177:D        1 (* CATALOGUE INFORMATION, zero entry refers to the directory itself *)
   178:S
```



```
   179:D        1   catentry = record
   180:D        1     cname:  tid;                (name of file or directory)
   181:D        1     ceft:   shortint;           (external file type (LIF))
   182:D        1     ckind:  filekind;           (file kind)
   183:S              cpsize: integer;            (physical size of file
   184:D        1                                         or of total data space on volume)
   185:S              clsize: integer;            (logical size of file
   186:D        1                                         or unused space on medium)
   187:S              cstart: integer;            (starting location of file
   188:D        1                                         or first possible data location)
   189:D        1     cblocksize: integer;        (size of a sector or block)
   190:D        1     ccreatedate, clastdate:  daterec;   (creation, last modified dates)
   191:D        1     ccreatetime, clasttime:  timerec;   (creation, last modified times)
   192:S              cextra1,                    (extension
   193:S                                                 or total possible number of files
   194:D        1                                         or requested number of files)
   195:S              cextra2:   integer;         (secondary discretionary field
   196:D        1                                         or start index of requested catalog)
   197:D        1     cinfo:  string[20];         (comment or miscellaneous information)
   198:S
   199:D        1     end;
   200:S
   201:D        1     passentry = record
   202:D        1                 pbits: integer;
   203:D        1                 pword: passtype;
   204:D        1                 end;
   205:S
   206:D        1 var
   207:D    -2  1   idle: byte;              { idle character -- 3.0 bug jws 3/20/84 }
   208:S
   209:D    -2  1 procedure getioerrmsg(var s :string; lastior : integer);
   210:D    -2  1 procedure printerror(errorcode, lastior: integer);
   211:D    -2  1 procedure upc(var s: string);
   212:D    -2  1 function ueovbytes(unit: unitnum): integer;
   213:D    -2  1 procedure unblockeddam(anyvar f: fib; unum: unitnum; request: damrequesttype);
   214:D    -2  1 procedure initfilekinds;
   215:D    -2  1 procedure lockup;
   216:D    -2  1 procedure lockdown;
   217:S
   218:S
   219:D    -2  1 implement
   220:S
   221:D        1 procedure lockup;
   222:C        2 begin
   223:C        2   locklevel := locklevel + 1;
   224:C        2 end;
   225:S
   226:D        1 procedure lockdown;
   227:D    -4  2 var saveio: integer;
   228:D    -6  2     saveec: shortint;
   229:C        2 begin
   230:C        2 locklevel := locklevel - 1;
   231:C        2 if locklevel = 0 then
   232:C        3   begin
   233:C        3   saveio := ioresult;
   234:C        3   saveec := sysescapecode;
   235:C        3   while actionspending > 0 do
   236:C        4     begin
   237:C        4     call(deferredaction[actionspending]);
   238:C        4     actionspending := actionspending - 1;
```

```
239:C        4     end;
240:C        3   ioresult := saveio;
241:C        3   sysescapecode := saveec;
242:C        3   end;
243:C        2 end;
244:S
245:S
246:D        1 procedure upc(var s: string);
247:D    -2  2 var i: shortint;
248:D    -3  2     c: char;
249:C        2 begin
250:C        2  for i := 1 to strlen(s) do
251:C        3    begin
252:C        3    c := s[i];           {this assignment saves two or three subscripts}
253:C        3    if (c >= 'a') and (c <= 'z') then s[i] := chr(ord(c)+ (ord('A')-ord('a')));
254:C        3    end;
255:C        2 end;
256:S
257:D        1 function ueovbytes(unit: unitnum): integer;
258:D        2   label 1;
259:D    -4  2   var return_bytes: integer;
260:D    -8  2       saved_ioresult: integer;
261:D    -9  2       saved_reportbit: boolean;
262:D   -12  2       tracks: shortint;
263:D  -674  2       f: fib;
264:S
265:D        2 function try_tracks(number_of_tracks: shortint): boolean;
266:C        3 begin {try_tracks}
267:C        3   f.fpeof := number_of_tracks*(30*256)-unitable^[unit].byteoffset;
268:C        3   f.funit := unit; f.fileid := 0;
269:C        3   call (unitable^[unit].tm, addr(f), readbytes, f.fbuffer, 256, f.fpeof-512);
270:C        3   if ioresult=0 then
271:C        4     begin
272:C        4       return_bytes := f.fpeof; { <<< desired side-effect! }
273:C        4       try_tracks := true
274:C        4     end {if}
275:C        4   else
276:C        4     try_tracks := false
277:C        4 end; {try_tracks}
278:S
279:C        2 begin {ueovbytes}
280:C        2 with unitable^[unit] do
281:C        3  if not uisblkd then ueovbytes := maxint
282:C        4  else if (letter<>'H') and (letter<>'F') then ueovbytes := umaxbytes
283:C        5  else
284:C        5     begin
285:C        5     saved_reportbit := ureportchange;  ureportchange := false;
286:C        5     saved_ioresult := ioresult;
287:C        5     return_bytes := 0;              {in case none of the tries succeed!!!}
288:C        5     if letter='H' then
289:C        6       if try_tracks(150) then goto 1;         {DS disc init'ed by a 9895}
290:C        5     for tracks := 61 to 67 do
291:C        6       if not try_tracks(tracks) then goto 1;  {SS disc init'ed by a 9885}
292:C        5     if not try_tracks(73) then {goto 1};      {SS disc init'ed by a 9895}
293:C        5 1:  ioresult := saved_ioresult;
294:C        5     ureportchange := saved_reportbit;
295:C        5     if return_bytes<=umaxbytes then ueovbytes := return_bytes
296:C        6                                else ueovbytes := umaxbytes;
297:C        5     end;
298:C        2 end; {ueovbytes}
```



```
299:S
300:D        1 procedure getioerrmsg(var s : string; lastior : integer);
301:D    -4  2 var dummy: integer;
302:C        2 begin
303:C        2   case lastior of
304:C        3     ord(inoerror      ):  s := '(no I/O error reported)';
305:C        3     ord(zbadblock     ):  s := 'block parity error';
306:C        3     ord(ibadunit      ):  s := 'illegal unit number';
307:C        3     ord(zbadmode      ):  s := 'illegal IO request';
308:C        3     ord(ztimeout      ):  s := 'device timeout';
309:C        3     ord(ilostunit     ):  s := 'volume has gone off-line';
310:C        3     ord(ilostfile     ):  s := 'file lost in dir';
311:C        3     ord(ibadtitle     ):  s := 'bad file name';
312:C        3     ord(inoroom       ):  s := 'no room on vol';
313:C        3     ord(inounit       ):  s := 'logical volume not found';
314:C        3     ord(inofile       ):  s := 'file not found';
315:C        3     ord(idupfile      ):  s := 'dup dir entry';
316:C        3     ord(inotclosed    ):  s := 'file already open';
317:C        3     ord(inotopen      ):  s := 'file not open';
318:C        3     ord(ibadformat    ):  s := 'bad input format';
319:C        3     ord(znosuchblk    ):  s := 'block number out of range';
320:C        3     ord(znodevice     ):  s := 'device absent or unaccessible';
321:C        3     ord(zinitfail     ):  s := 'medium formatting/sparing failed';
322:C        3     ord(zprotected    ):  s := 'medium is write protected';
323:C        3     ord(zstrangei     ):  s := 'unexpected interrupt';
324:C        3     ord(zbadhardware  ):  s := 'hardware fault';
325:C        3     ord(zcatchall     ):  s := 'unrecognized error state';
326:C        3     ord(zbaddma       ):  s := 'DMA absent or unavailable';
327:C        3     ord(inotvalidsize ):  s := 'file size not compatible with type';
328:C        3     ord(inotreadable  ):  s := 'file not opened for reading';
329:C        3     ord(inotwriteable ):  s := 'file not opened for writing';
330:C        3     ord(inotdirect    ):  s := 'file not opened for direct access';
331:C        3     ord(idirfull      ):  s := 'no room in directory';
332:C        3     ord(istrovfl      ):  s := 'string subscript out of range';
333:C        3     ord(ibadclose     ):  s := 'bad file close string parameter';
334:C        3     ord(ieof          ):  s := 'tried to read or write past eof';
335:C        3     ord(zuninitialized):  s := 'medium uninitialized';
336:C        3     ord(znoblock      ):  s := 'block not found';
337:C        3     ord(znotready     ):  s := 'device not ready or medium absent';
338:C        3     ord(znomedium     ):  s := 'medium absent';
339:C        3     ord(inodirectory  ):  s := 'no directory on volume';
340:C        3     ord(ibadfiletype  ):  s := 'file type illegal or does not match';
341:C        3     ord(ibadvalue     ):  s := 'parameter illegal or out of range';
342:C        3     ord(icantstretch  ):  s := 'file cannot be extended';
343:C        3     ord(ibadrequest   ):  s := 'undefined operation for unit/file';
344:C        3     ord(inotlockable  ):  s := 'file not lockable';
345:C        3     ord(ifilelocked   ):  s := 'file already locked';
346:C        3     ord(ifileunlocked ):  s := 'file not locked';
347:C        3     ord(idirnotempty  ):  s := 'directory not empty';
348:C        3     ord(itoomanyopen  ):  s := 'too many files open on device';
349:C        3     ord(inoaccess     ):  s := 'access to file not allowed';
350:C        3     ord(ibadpass      ):  s := 'invalid password';
351:C        3     ord(ifilenotdir   ):  s := 'file is not a directory';
352:C        3     ord(inotondir     ):  s := 'operation not allowed on directory';
353:C        3     ord(ineedtempdir  ):  s := 'cannot create /WORKSTATIONS/TEMP_FILES';
354:C        3     ord(isrmcatchall  ):  s := 'unrecognized SRM error';
355:C        3     ord(zmediumchanged):  s := 'medium may have been changed';
356:S
357:C        3     otherwise
358:C        3       begin setstrlen(s,0);
```

```
359:C        3             strwrite(s, 1, dummy, 'ioresult was ', lastior:1);
360:C        3       end;
361:S
362:C        3   end (*IO ERRORS*) ;
363:C        2 end;
364:S
365:D        1 procedure printerror(errorcode,lastior: integer);
366:D        2 label 1;
367:D  -164  2 var   s,st: string[80];
368:D  -164  2       excp_line['excp_line']: integer;
369:D  -164  2       excp_pc['excp_pc']: integer;
370:C        2 begin
371:C        2   writeln(cteol,bellchar); write(cteol);
372:S
373:C        2   if errorcode > 0 then
374:C        3     writeln(output,'Abnormal termination.  Halt code ',errorcode:1)
375:C        3   else
376:C        3     begin
377:C        3       case errorcode of
378:C        4     0:  s := 'normal termination not caught by GO';
379:C        4    -1:  s := 'abnormal termination not caught by GO';
380:C        4    -2:  s := 'not enough memory';
381:C        4    -3:  s := 'reference to NIL pointer';
382:C        4    -4:  s := 'integer overflow';
383:C        4    -5:  s := 'divide by zero';
384:C        4    -6:  s := 'real math overflow';
385:C        4    -7:  s := 'real math underflow';
386:C        4    -8:  s := 'value range error';
387:C        4    -9:  s := 'case value range error';
388:S
389:C        4   -10:  getioerrmsg(s,lastior);           (* IORESULT <> 0 *)
390:S
391:C        4   -11:  s := 'CPU word access to odd address';
392:C        4   -12:  s := 'CPU bus error';
393:C        4   -13:  s := 'illegal CPU instruction';
394:C        4   -14:  s := 'CPU privilege violation';
395:C        4   -15:  s := 'bad argument: SIN/COS';
396:C        4   -16:  s := 'bad argument: LN';
397:C        4   -17:  s := 'bad argument: SQRT';
398:C        4   -18:  s := 'bad argument: real/BCD conversion';
399:C        4   -19:  s := 'bad argument: BCD/real conversion';
400:C        4   -20:  s := 'stopped by user';
401:C        4   -21:  s := 'unassigned CPU trap';
402:C        4 (*-22:                          *************** call to debugger ********)
403:C        4   -23:  goto 1;                   (******** give no message !! *****)
404:C        4   -24:  s := 'macro parameter not 0..9 or A..Z';
405:C        4   -25:  s := 'undefined macro parameter';
406:C        4   -26:  s := 'I/O routine error';
407:C        4   -27:  s := 'graphics routine error';
408:C        4   -28:  s := 'ram parity error';
409:C        4   -29:  s := 'misc floating pt hardware error';
410:S
411:C        4 otherwise
412:C        4         s := 'undocumented error'
413:C        4       end; (*CASE ERRORCODE*)
414:C        3       writeln(output,
415:C        3 '------------------------------------------------');
416:C        3       writeln('error ',errorcode:1,': ',s,cteol); write(cteol);
417:C        3       if excp_line >= 0 then write('line number:  ',excp_line:1, '   ');
418:C        3       writeln('PC value:    ',excp_pc:1);
```



```
419:C        3     end;
420:C        2 1: end (*PRINTERROR*) ;
421:S
422:S
423:S
424:D    -2  1 (*DIRECTORY ACCESS METHOD FOR DIRECT UNIT OPEN,   ASSUMES NO DIRECTORY *)
425:S
426:D        1 procedure unblockeddam(anyvar f: fib; unum: unitnum; request: damrequesttype);
427:D        2 type vidptr = ^vid;
428:D    -2  2 var buf: shortint;
429:C        2 begin
430:C        2 ioresult := 0;
431:C        2 with f, unitable^[unum] do
432:C        3  case request of
433:C        4   stretchit,                      {can't do anything}
434:C        4   purgefile, closefile:           {nothing to do};
435:C        4                                   {changed stripname for version 2.2 on 4-May-83}
436:C        4   stripname: begin                {move ftitle to ftid and set ftitle to null}
437:C        4                 if strlen(ftitle) > tidleng then
438:C        5                   begin
439:C        5                     ioresult := ord(ibadtitle);
440:C        5                     setstrlen(ftid,0);
441:C        5                   end
442:C        5                 else
443:C        5                   ftid := ftitle;
444:C        4                 setstrlen(ftitle,0);
445:C        4               end;
446:C        4   getvolumename: vidptr(addr(f))^ := uvid;
447:C        4   setvolumename: uvid := vidptr(addr(f))^;
448:C        4   openunit,
449:C        4   openvolume,
450:C        4   createfile,
451:C        4   openfile:  begin
452:C        4               fileid := 0;
453:C        4               ureportchange := false;
454:C        4               if request = openunit then fpeof := umaxbytes
455:C        5               else if not uisblkd then fpeof := maxint
456:C        6                 else begin
457:C        6                        FPEOF := MAXINT;   {CHANGED FOR 9122 4/11/84 SFB}
458:C        6                        call(tm, addr(f), readbytes, buf, 2, 0); {TOUCH DISC}
459:S                                {UNITABLE^[UNUM].UMAXBYTES WAS VALIDATED BY TM CALL -
460:S                                 IE DISC CONTROLLER SENT PRESENT MEDIA VALUE IF SMART
461:C        6                        ENOUGH}
462:C        6                        fpeof := ueovbytes(unum);  {MOVED 4/11/84 SFB}
463:C        6                        {UEOVBYTES SET TO UMAXBYTES IF BLOCKED & NOT 9885/9895}
464:C        6                      end;
465:C        4               ureportchange := true; umediavalid := true;
466:C        4               fleof := fpeof;
467:C        4               fisnew := false;
468:C        4               if     not uisblkd  then
469:C        5                    if not fistextvar then am := tm
470:C        6                                      else am := serialtextamhook
471:C        6               else if not fbuffered  then am := amtable^[untypedfile]
472:C        6               else if not fistextvar then am := amtable^[datafile]
473:C        7                                      else am := amtable^[fkind];
474:C        4               end;
475:S
476:C        4   otherwise ioresult := ord(ibadrequest);
477:C        4  end;
478:C        2 end;
```

```
479:S
480:D    -2  1 (*  ACCESS METHOD FOR UNBUFFERED TRANSFERS *)
481:D    -2  1 (updates fpos, checks logical limits of file, calls "stretch" if necessary)
482:S
483:D        1 procedure unbuffedam(fp: fibp;  request: amrequesttype;
484:D        2                      anyvar buffer: window; buffsize, position: integer);
485:D        2 label 1;
486:C        2 begin
487:C        2 with fp^, unitable^[funit] do
488:C        3   case request of
489:C        4     readbytes,writebytes:
490:C        4     begin
491:C        4     fpos := position + buffsize;
492:C        4     if fpos > fleof then
493:C        5       if (request=readbytes) then begin ioresult := ord(ieof); goto 1; end
494:C        6       else
495:C        6         begin
496:C        6         if fpos > fpeof then
497:C        7           begin
498:C        7           call(dam, fp^, funit, stretchit);
499:C        7           if fpos > fpeof then begin ioresult := ord(ieof); goto 1; end
500:C        8           end;
501:C        6         fleof := fpos; fmodified := true;
502:C        6         end;
503:C        4     call(tm, fp, request, buffer, buffsize, position);
504:C        4     end;
505:C        4     flush: call(tm, fp, request, buffer, buffsize, position);
506:C        4     otherwise ioresult := ord(ibadrequest);
507:C        4   end;
508:C        2 1:
509:C        2 end;
510:S
511:S
512:D    -2  1 (*  ACCESS METHOD FOR SERIAL TEXT INPUT DEVICES *)
513:D    -2  1 (converts a carriage return character to an 'end of line' indication)
514:S
515:D        1 procedure serialtextam(fp: fibp;  request: amrequesttype;
516:D        2                        anyvar buffer: window; buffsize, position: integer);
517:D    -4  2 var i: integer;
518:C        2 begin
519:C        2 with fp^, unitable^[funit] do
520:C        3   begin
521:C        3   call(tm, fp, request, buffer, buffsize, position);
522:C        3   if ioresult = ord(inoerror) then
523:C        4     if request = readbytes then
524:C        5     begin
525:C        5     feoln := buffer[buffsize-1] = eol;
526:C        5     for i := 0 to buffsize - 1 do if buffer[i]=eol then buffer[i] := ' ';
527:C        5     end;
528:C        3   end;
529:C        2 end;
530:S
531:D    -2  1 (* ACCESS METHOD FOR DATA FILES *)
532:D    -2  1 (accomplishes general purpose buffering)
533:S
534:D        1 procedure standardam(fp: fibp;  request: amrequesttype;
535:D        2                      anyvar buffer: window; buffsize, position: integer);
536:D        2 label 1,2;
537:D        2 var lastblock, block, oldfleof, oldfpos,                {rdq}
538:D   -36  2     firstpos, firstbytes, middlebytes, endbytes, i: integer;
```



```
539:D   -37  2     c: char;
540:S
541:D        2   procedure flushbuffer;
542:D    -4  3   var bufsize: integer;
543:C        3   begin with fp^ do
544:C        4     if fbufchanged (block buffer has been written) then
545:C        5       begin
546:C        5       bufsize := fleof - flastpos;
547:C        5       if bufsize > fblksize then bufsize := fblksize;
548:C        5       call (unitable^[funit].tm, fp, writebytes, fbuffer, bufsize, flastpos);
549:C        5       if ioresult <> ord(inoerror) then goto 1;
550:C        5       fbufchanged := false;
551:C        5       end;
552:C        3   end;
553:S
554:D        2   procedure fetchbuffer;
555:D    -8  3   var i, bufsize: integer;
556:C        3   begin if block<>lastblock then with fp^ do
557:C        5     begin
558:C        5     flushbuffer;
559:C        5     lastblock := block;  flastpos := lastblock*fblksize;
560:C        5     bufsize := oldfleof - flastpos;
561:C        5     if bufsize <= 0 then bufsize := 0 else
562:C        6       begin
563:C        6       if bufsize > fblksize then bufsize := fblksize;
564:C        6       call (unitable^[funit].tm, fp, readbytes, fbuffer, bufsize, flastpos);
565:C        6       if ioresult <> ord(inoerror) then goto 1;
566:C        6       end;
567:C        5     for i := bufsize to fblksize-1 do fbuffer[i] := chr(0);
568:C        5     end;
569:C        3   end;
570:S
571:C        2 begin with fp^ do
572:C        3   begin
573:C        3   ioresult := ord(inoerror);
574:C        3   oldfleof := fleof; oldfpos := fpos;                   {rdq}
575:C        3   case request of
576:C        4     flush: begin
577:C        4            flushbuffer;
578:C        4            call(unitable^[funit].tm, fp, flush, fp^, 0, 0);
579:C        4            end;
580:C        4     writeeol: begin
581:C        4               c := eol;
582:C        4               standardam(fp, writebytes, c, 1, position);
583:C        4               end;
584:C        4     readtoeol:
585:C        4       begin
586:C        4       middlebytes := 0;
587:C        4       fpos := position;
588:C        4       if buffsize > fleof - position then buffsize := fleof - position;
589:C        4       lastblock := (flastpos - flastpos mod fblksize) div fblksize;
590:C        4       while buffsize > 0 do
591:C        5         begin
592:C        5         block := fpos div fblksize;
593:C        5         fetchbuffer;
594:C        5         firstpos := fpos - flastpos;
595:C        5         firstbytes := fblksize - firstpos;
596:C        5         if firstbytes > buffsize then firstbytes := buffsize;
597:C        5         for i := 0 to firstbytes-1 do
598:C        6           if fbuffer[firstpos+i]=eol then
```

```
599:C        7           begin firstbytes := i; buffsize := i; goto 2; end;
600:C        5       2: moveleft(fbuffer[firstpos], buffer[1+middlebytes], firstbytes);
601:C        5       middlebytes := middlebytes + firstbytes;
602:C        5       fpos := fpos + firstbytes;
603:C        5       buffsize := buffsize - firstbytes;
604:C        5       end;
605:C        4     buffer[0] := chr(middlebytes);
606:C        4     if middlebytes>0 then fp^.feoln := false;
607:C        4     end;
608:C        4   readbytes,writebytes:
609:C        4     begin
610:C        4     fpos := position + buffsize;
611:C        4     if fpos > fleof then
612:C        5       if (request = readbytes) then
613:C        6         begin {feoln := true;} ioresult := ord(ieof); fpos:=oldfpos; goto 1; end
614:C        6       else begin
615:C        6            if fpos > fpeof then
616:C        7                begin
617:C        7                call(unitable^[funit].dam, fp^, funit, stretchit);
618:C        7                if fpos > fpeof then begin ioresult := ord(ieof); fpos := oldfpos;
619:C        8                                           goto 1;
620:C        8                                      end;
621:C        7                end;
622:C        6            fleof := fpos; fmodified := true;
623:C        6            end;
624:S
625:C        4     if flastpos < 0 then lastblock := -1
626:C        5     else lastblock := flastpos div fblksize;
627:C        4     block     := position div fblksize;
628:C        4     if (buffsize=1) and (block=lastblock)   {the most common case!}   then
629:C        5       if request = readbytes then buffer[0] := fbuffer[position mod fblksize]
630:C        6       else begin
631:C        6            fbuffer[position mod fblksize] := buffer[0];
632:C        6            fbufchanged := true;
633:C        6            end
634:C        6     else
635:C        5       begin
636:C        5       firstpos := (-position) mod fblksize;
637:C        5       if firstpos >= buffsize then
638:C        6           begin firstbytes := buffsize; endbytes := 0; end
639:C        6       else begin firstbytes := firstpos; endbytes :=fpos mod fblksize; end;
640:C        5       middlebytes := buffsize - firstbytes - endbytes;
641:S
642:C        5       if firstbytes > 0 then
643:C        6         begin
644:C        6         fetchbuffer;
645:C        6         if request=readbytes
646:C        7           then moveleft(fbuffer[fblksize-firstpos], buffer, firstbytes)
647:C        7         else
648:C        7           begin
649:C        7           moveleft(buffer, fbuffer[fblksize-firstpos], firstbytes);
650:C        7           fbufchanged := true;
651:C        7           end;
652:C        6         block := block + 1;
653:C        6         end;
654:C        5       while middlebytes > 0 do
655:C        6         begin
656:C        6         if request=readbytes
657:C        7           then begin
658:C        7                fetchbuffer;
```



```
659:C        7                moveleft(fbuffer, buffer[firstbytes], fblksize)
660:C        7                end
661:C        7           else begin
662:C        7                flushbuffer;
663:C        7                lastblock := block;  flastpos := lastblock *  fblksize;
664:C        7                moveleft(buffer[firstbytes], fbuffer, fblksize);
665:C        7                fbufchanged := true;
666:C        7                end;
667:C        6         middlebytes := middlebytes -  fblksize;
668:C        6         firstbytes := firstbytes + fblksize;
669:C        6         block := block + 1;
670:C        6         end;
671:C        5       if endbytes > 0 then
672:C        6         begin
673:C        6         fetchbuffer;
674:C        6         if request=readbytes
675:C        7           then moveleft(fbuffer, buffer[firstbytes], endbytes)
676:C        7           else begin
677:C        7                moveleft(buffer[firstbytes], fbuffer, endbytes);
678:C        7                fbufchanged := true;
679:C        7                end;
680:C        6         end;
681:C        5       end;
682:C        4     if fistextvar then if request = readbytes then
683:C        6       for i := 0 to buffsize-1 do
684:C        7       if buffer[i] = eol then begin feoln := true; buffer[i] := ' '; end
685:C        8       else feoln := false;
686:C        4     end;
687:C        4     otherwise ioresult := ord(ibadrequest);
688:C        4   end;
689:C        3 end;
690:C        2 1:
691:C        2 end;
692:S
693:S
694:D        1 procedure initfilekinds;
695:D    -2  2 var fk: filekind;
696:C        2 begin
697:C        2   serialtextamhook := serialtextam;
698:C        2
699:C        2   {new(efttable);                                  ALREADY DONE IN BOOT LOADER}
700:C        2   new(amtable);
701:C        2   new(suffixtable);
702:C        2
703:C        2   for fk := untypedfile to lastfkind do
704:C        3     begin
705:C        3     suffixtable^[fk]              := '';          {no suffix    }
706:C        3     amtable^[fk]                  := unbuffedam;  {no buffering}
707:C        3     efttable^[fk]                 := 0;           {unassociated LIF file type}
708:C        3     end;
709:S
710:C        2     efttable^   [untypedfile]     := 3;           {LIF directory}
711:C        2                                                   {no suffix}
712:C        2                                                   {no buffering}
713:S
714:C        2     suffixtable^[badfile]         := 'BAD';       {bad block indication}
715:C        2     efttable^   [badfile]         := 2;           {LIF bad block marker}
716:C        2                                                   {no buffering}
717:S
718:C        2     efttable^   [datafile]        := -5622;       {DCD Pascal data file}
```

```
Pascal [Rev 3.0M  6/ 4/84] MISC.TEXT              26-Dec-84 21:04:28 Page 13   *** File name: INIT ***

  719:C        2                                                 {no suffix}
  720:C        2   amtable^    [datafile]         := standardam;  {general purpose buffering}
  721:S
  722:C        2   suffixtable^[codefile]         := 'CODE';      {code file suffix}
  723:C        2   efttable^   [codefile]         := -5582;       {DCD Pascal code file}
  724:C        2                                                 {no buffering}
  725:S
  726:C        2   suffixtable^[sysfile]          := 'SYSTM';     {suffix for system file}
  727:C        2   efttable^   [sysfile]          := -5822;       {DCD system (boot) file}
  728:C        2                                                 {no buffering}
  729:C        2 end;
  730:S
  731:S
  732:C        1 end  {miscellaneous stuff module}
  733:S
  734:C        2   $include 'MISC'$;
 2130:C        2 $LIST ON$
 2131:S
 2132:C        2   $LIST OFF, include 'FS'$;
 2132:D        1   $LIST  ON, include 'INITUNITS'$;
 2133:D        1 module initunits;
 2134:S
 2135:D        1 import sysglobals,mini,asm,fs;
 2136:S
 2137:D        1 export
 2138:S
 2139:D        1   procedure nounit(fp: fibp; request: amrequesttype; anyvar buffer: window;
 2140:D        2                                                       bufsize, position: integer);
 2141:D        1   procedure nodam(anyvar f: fib; unum: unitnum; request: damrequesttype);
 2142:D        1   procedure noisr(isribptr: pisrib);
 2143:D        1   procedure unitioinit;
 2144:S
 2145:D        1 implement
 2146:S
 2147:D        1 procedure crtio$alias 'sysdevs_crtio'$
 2148:D        2           (fp: fibp; request: amrequesttype; anyvar buffer: window;
 2149:D        2                                              bufsize, position: integer); EXTERNAL;
 2150:S
 2151:D        1 procedure kbdio$alias 'sysdevs_kbdio'$
 2152:D        2           (fp: fibp; request: amrequesttype; anyvar buffer: window;
 2153:D        2                                              bufsize, position: integer); EXTERNAL;
 2154:S
 2155:D        1 procedure boottm $alias 'bootdammodule_boottm'$
 2156:D        2           (fp: fibp; request: amrequesttype; anyvar buffer: window;
 2157:D        2                                              bufsize, position: integer); EXTERNAL;
 2158:S
 2159:D        1 procedure bootdam $alias 'bootdammodule_bootdam'$
 2160:D        2            (anyvar f: fib; unum: unitnum; request: damrequesttype); EXTERNAL;
 2161:S
 2162:D        1 procedure initbootdam $alias 'bootdammodule_initbootdam'$ ; EXTERNAL;
 2163:S
 2164:D        1 procedure unblockeddam $alias 'MISC_UNBLOCKEDDAM'$
 2165:D        2             (anyvar f: fib; unum: unitnum; request: damrequesttype); EXTERNAL;
 2166:S
 2167:D        1 procedure noisr(isribptr: pisrib);
 2168:C        2 begin
 2169:C        2 end;
 2170:S
 2171:D        1 procedure initsysisr;
 2172:D    -4  2 var i:integer;
```

```
Pascal [Rev 3.0M  6/ 4/84] INITUNITS.TEXT         26-Dec-84 21:04:28 Page 14   *** File name: INIT ***

 2173:C        2 begin
 2174:C        2   for i:= 1 to 7 do
 2175:C        3     begin
 2176:C        3       interrupttable[i] := nil;
 2177:C        3       perminttable[i]   := nil;
 2178:C        3     end;
 2179:C        2 end;
 2180:S
 2181:D        1 procedure initunitentry(un: unitnum;
 2182:D        2                         p_dam: damtype;
 2183:D        2                         p_am:  amtype;
 2184:D        2                       { p_sc, }
 2185:D        2                       { p_ba, }
 2186:D        2                       { p_du, }
 2187:D        2                       { p_dv: byte; }
 2188:D        2                       { p_byteoffset: integer; }
 2189:D        2                       { p_devid: shortint; }
 2190:D        2                         p_uvid: vid;
 2191:D        2                       { p_dvrtemp: integer }
 2192:D        2                       { p_letter: char; }
 2193:D        2                       { p_offline: boolean }
 2194:D        2                         p_uisinteractive: boolean;
 2195:D        2                       { p_umediavalid: boolean; }
 2196:D        2                       { p_uuppercase: boolean; }
 2197:D        2                       { p_uisfixed: boolean; }
 2198:D        2                       { p_ureportchange: boolean; }
 2199:D        2                       { p_pad: 0..1 }
 2200:D        2                         p_uisblkd: boolean {;}
 2201:D   -18  2                       { p_umaxbytes: shortint } );
 2202:C        2 begin {initunitentry}
 2203:C        2   with unitable^[un] do begin
 2204:C        3     dam             := p_dam;
 2205:C        3     tm              := p_am;
 2206:C        3     sc              := 0;
 2207:C        3     ba              := 0;
 2208:C        3     du              := 0;
 2209:C        3     dv              := 0;
 2210:C        3     byteoffset      := 0;
 2211:C        3     devid           := 0;
 2212:C        3     uvid            := p_uvid;
 2213:C        3     dvrtemp         := 0;
 2214:C        3     letter          := chr(0);
 2215:C        3     offline         := false;
 2216:C        3     uisinteractive  := p_uisinteractive;
 2217:C        3     umediavalid     := false;
 2218:C        3     uuppercase      := not p_uisblkd;
 2219:C        3     uisfixed        := false;
 2220:C        3     ureportchange   := false;
 2221:C        3     pad             := 0;
 2222:C        3     uisblkd         := p_uisblkd;
 2223:C        3     if uisblkd then
 2224:C        4       umaxbytes     := 0;
 2225:C        3   end {with}
 2226:C        3 end; {initunitentry}
 2227:S
 2228:D        1 procedure nounit(fp: fibp; request: amrequesttype; anyvar buffer: window;
 2229:D        2                                                    bufsize, position: integer);
 2230:C        2 begin ioresult := ord(znodevice) end;
 2231:S
 2232:D        1 procedure nodam(anyvar f: fib; unum: unitnum; request: damrequesttype);
```

```
Pascal [Rev 3.0M  6/ 4/84] INITUNITS.TEXT          26-Dec-84 21:04:28 Page 15   *** File name: INIT ***

  2233:C        2 begin ioresult := ord(znodevice) end;
  2234:S
  2235:D        1 procedure initunitable;
  2236:D    -2  2 var i: unitnum;
  2237:C        2 begin
  2238:C        2 new(unitable);
  2239:C        2 for i := 0 to maxunit do
  2240:C        3 initunitentry
  2241:C        3   (i, nodam      , nounit, '',         false,false);
  2242:S
  2243:C        2 initunitentry
  2244:C        2   (1, unblockeddam,crtio,  'CONSOLE',  true, false);
  2245:C        2 initunitentry
  2246:C        2   (2, unblockeddam,kbdio,  'SYSTERM',  false,false);
  2247:C        2 initunitentry
  2248:C        2   (3, bootdam    , boottm, '',         false,true );
  2249:C        2 initunitentry
  2250:C        2   (6, unblockeddam,crtio,  'PRINTER',  false,false);
  2251:S
  2252:C        2 end;
  2253:S
  2254:D        1 procedure unitioinit;
  2255:D    -2  2 var i: unitnum;
  2256:D  -664  2     f: fib;
  2257:C        2 begin
  2258:C        2   initsysisr;
  2259:S
  2260:C        2   sysunit := 3;
  2261:C        2   initunitable;
  2262:C        2
  2263:C        2 end;
  2264:S
  2265:C        1 end
  2266:S
  2267:C        2   $LIST  ON, include 'INITUNITS'$;
  2267:D        1   $include 'LDR'$;
  2268:D        1 module ldr;
  2269:S
  2270:D        1 import sysglobals, misc, fs, loader, asm;
  2271:S
  2272:D        1 export
  2273:D        1    procedure openlinkfile(var filename: string);
  2274:D        1    procedure load(filetogo:fid;  permanent: boolean);
  2275:S
  2276:D        1    procedure initsysunit;
  2277:D        1    procedure lockfiles;
  2278:D        1    procedure openfiles;
  2279:S
  2280:D        1    procedure go;
  2281:D        1    procedure loadrom(name: fid);
  2282:S
  2283:S
  2284:D        1 implement
  2285:S
  2286:D        1 procedure lockfiles;
  2287:C        2 begin
  2288:C        2   close(input);
  2289:C        2   close(output,     'LOCK');
  2290:C        2   close(gfiles[2]^);                      {KEYBOARD}
  2291:C        2   close(gfiles[4]^, 'LOCK');              {LISTING}
```

```
Pascal [Rev 3.0M  6/ 4/84] LDR.TEXT                26-Dec-84 21:04:28 Page 16   *** File name: INIT ***

  2292:C        2 end;
  2293:S
  2294:D        1 procedure openfiles;
  2295:C        2 begin
  2296:C        2   if not fibp(addr(input   ))^.freadable  then
  2297:C        3                                 reset  (input,      'CONSOLE:');
  2298:C        2   with   fibp(addr(output  ))^ do
  2299:C        3      begin
  2300:C        3      if (fleof>0) and fwriteable then close  (output,     'LOCK');
  2301:C        3      if not fwriteable then         rewrite(output,     'CONSOLE:');
  2302:C        3      end;
  2303:S
  2304:C        2   if not fibp(gfiles[2]     )^.freadable  then          {KEYBOARD}
  2305:C        3                                 reset  (gfiles[2]^, 'SYSTERM:');
  2306:C        2   with   fibp(gfiles[4]     )^ do
  2307:C        3      begin                                              {LISTING}
  2308:C        3      if (fleof>0) and fwriteable then close  (gfiles[4]^, 'LOCK');
  2309:C        3      if not fwriteable then         rewrite(gfiles[4]^, 'PRINTER:LISTING.ASC');
  2310:C        3      end;
  2311:C        2 end;
  2312:S
  2313:D        1 procedure initsysunit;
  2314:D    -4  2 var i: integer;
  2315:C        2 begin
  2316:C        2   syvid := '#';  strwrite(syvid,2,i,sysunit:1);
  2317:C        2   i := findvolume(syvid, true);
  2318:C        2   dkvid := syvid;
  2319:C        2 end; (*INITUNITABLE*)
  2320:S
  2321:D        1 procedure go;
  2322:D    -4  2 var  lastioresult:integer;
  2323:D    -8  2      userheap: anyptr;
  2324:D    -9  2      done: boolean;
  2325:D   -14  2      modptr: moddescptr;
  2326:C        2 begin {GO}
  2327:C        2   if entrypoint = nil then writeln('(no start address)')
  2328:C        3   else
  2329:C        3       begin
  2330:C        3       page(output);
  2331:C        3       mark(userheap);
  2332:C        3       modptr := entrypoint;
  2333:C        3       repeat
  2334:C        4         done := modptr^.lastmodule;
  2335:C        4         if modptr^.startaddr<>0 then
  2336:C        5           begin
  2337:C        5           userprogram(modptr^.startaddr,userstack); (*** ALL PROGRAMS ARE ENTERED HERE ****)
  2338:C        5           if escapecode <> 0 then
  2339:C        6               begin
  2340:C        6               lastioresult:=ioresult;(save ioresult)
  2341:C        6               release(userheap);
  2342:C        6               if escapecode <> -1 then printerror(escapecode,lastioresult);
  2343:C        6               done := true;
  2344:C        6               end;
  2345:C        5           end;
  2346:C        4         modptr := modptr^.link;
  2347:C        4       until done;
  2348:C        3       end;
  2349:C        2 end;  {GO}
  2350:S
  2351:D  -122  1 procedure loadrom(name: fid);
```

```
2352:D  -144  2 var modptr: moddescptr; upcname: tid;
2353:C        2 begin
2354:C        2   entrypoint := nil;
2355:C        2   if strlen(name) <= tidleng then
2356:C        3     begin
2357:C        3     upcname := name;    upc(upcname);
2358:C        3     modptr := sysdefs;
2359:C        3     while modptr <> nil do with modptr^ do
2360:C        5       begin
2361:C        5       if (startaddr<>0) and
2362:C        6                        ((progname=name) or (ucase and (progname=upcname))) then
2363:C        6         begin
2364:C        6         entrypoint := modptr;
2365:C        6         modptr := nil;
2366:C        6         end
2367:C        6       else modptr := link;
2368:C        5       end;
2369:C        3     end;
2370:C        2 end;
2371:S
2372:D        1 procedure openlinkfile(var filename: string);
2373:D    -4  2 var extra:      addrec;
2374:C        2 begin {openlinkfile}
2375:C        2   getbytes(extra.a,  sizeof(addrec));  {save old value of loadfib}
2376:C        2   extra.arp^ := loadfib;
2377:C        2   getbytes(loadfib.a, sizeof(fib,1));   {create a FIB for new file}
2378:C        2   finitb(loadfib.fbp^, nil, -1);        {untyped, unbuffered file }
2379:C        2   reset(loadfib.php^, filename, 'shared');
2380:C        2   openlinkf(extra);
2381:C        2 end;
2382:S
2383:D        1 procedure printunres;
2384:D    -4  2 var modptr:     moddescptr;
2385:D    -8  2     strptr:     addrec;
2386:D   -10  2     i:          shortint;
2387:C        2 begin
2388:C        2   modptr := newmods;
2389:C        2   while modptr <> endmod do with modptr^ do
2390:C        4     begin
2391:C        4     if not resolved then
2392:C        5      for i := 0 to listsize - 1 do
2393:C        6       if listaddr^[i]<>0 then
2394:C        7        begin
2395:C        7        strptr.a := extaddr.a + listaddr^[i];
2396:C        7        writeln(strptr.syp^);
2397:C        7        end;
2398:C        4     modptr := link;
2399:C        4     end;
2400:C        2   writeln;
2401:C        2   writeln('The external references above were unresolved.');
2402:C        2   writeln;
2403:C        2   escape(-1);
2404:C        2 end;
2405:S
2406:D  -122  1 procedure load(filetogo:fid;  permanent: boolean);
2407:S
2408:D  -126  2 var modnum, ecode:   shortint;
2409:D  -130  2     icode: integer;
2410:S
2411:D        2   procedure printwhy;
```



```
2412:D   -82  3   var s: string80;
2413:C        3   begin
2414:C        3   getioerrmsg(s, ioresult);
2415:C        3   writeln('cannot open ''',filetogo, '''');
2416:C        3   writeln(s);
2417:C        3   end;
2418:S
2419:C        2 begin
2420:S
2421:S          (*This routine must:
2422:S            1. Set "ENTRYPOINT" to nil (no start address) or correct value.
2423:S            2. Load executable code into heap.
2424:S            3. Set "SYSDEFS"  to user program symbol table.
2425:C        2   4. Set "USERSTACK" to initial SP value. *)
2426:S
2427:C        2 loadfib.php := nil;
2428:C        2 try
2429:C        3   releaseuser;          {get rid of last program}
2430:C        3   mark(lowheap.p);              highheap.a := userstack;
2431:C        3   startreloc := lowheap.a;
2432:C        3   newmods       := sysdefs;     endmod          := sysdefs;
2433:S
2434:C        3   writeln; writeln('Loading  ''',filetogo,'''');
2435:C        3   openlinkfile(filetogo);
2436:C        3   if fdirectory = nil then      begin printwhy; escape(-1); end;
2437:C        3   for modnum := 1 to fdirectory^[0].dnumfiles do
2438:C        4                               begin loadinfo(modnum, true, false); checkrev; end;
2439:C        3   allresolved := true; matchfile;
2440:C        3   highheap.a := userstack;
2441:S
2442:C        3   if not allresolved then
2443:C        4     begin
2444:C        4     filetogo := syslibrary;
2445:C        4     if syvid <> '' then openlinkfile(filetogo)
2446:C        5     else fdirectory := nil;
2447:C        4     if fdirectory = nil then printwhy
2448:C        5     else
2449:C        5      begin
2450:C        5       repeat
2451:C        6         libfound := false;
2452:C        6         modnum := fdirectory^[0].dnumfiles;
2453:C        6         while (modnum > 0) and not allresolved do
2454:C        7           begin
2455:C        7           loadinfo(modnum, false, true);
2456:C        7           checkrev;
2457:C        7           modnum := modnum - 1;
2458:C        7           end;
2459:C        6       until allresolved or not libfound;
2460:C        5       highheap.a := userstack;
2461:C        5      end;
2462:C        4     end;
2463:S
2464:C        3   if not allresolved then printunres;
2465:S
2466:C        3   countcode;
2467:S
2468:C        3   highheap.a := userstack - totalglobal;
2469:C        3   if highheap.a < (a5 - 32768) then escape(117);
2470:C        3   startglobal := userstack - a5;
2471:C        3   userstack := highheap.a;
```

```
2472:C        3   zeromem(highheap.p, totalglobal);
2473:S
2474:C        3   loadtext(true);
2475:S
2476:C        3   movedefs(startreloc+totalreloc);
2477:S
2478:C        3   lowheap.a := startdefs + totaldefs;
2479:C        3   release(lowheap.p);
2480:S
2481:C        3   if not permanent then
2482:C        4    if entrypoint = nil then
2483:C        5     begin writeln('no start address');
2484:C        5           escape(-1);
2485:C        5     end;
2486:S
2487:C        3 recover
2488:C        3   begin
2489:C        3   ecode := escapecode; icode := ioresult;
2490:C        3   closefiles;
2491:C        3   if ecode <> -1 then
2492:C        4    case ecode of
2493:C        5    100..105: begin
2494:C        5              writeln('can''t link object at byte ',wrongbyte:1);
2495:C        5              writeln(' in text record ',wrongrec:1);
2496:C        5              writeln(' of module ''', linkmodname.syp^, '''');
2497:C        5              end;
2498:C        5    110: writeln('symbol def''s nested too deeply');
2499:C        5    111: writeln('improper link info format');
2500:C        5    112: writeln('not enough memory to load');
2501:C        5    116: writeln('''',filetogo, ''' not a codefile');
2502:C        5    117: writeln('available global space exceeded');
2503:C        5    118: writeln('incorrect version number');
2504:C        5    otherwise printerror(ecode,icode);
2505:C        5    end;
2506:C        3   releaseuser;
2507:C        3   escape(-1);
2508:C        3   end;
2509:C        2 end;  {LOAD}
2510:S
2511:C        1 end
2512:C        2   $include 'LDR'$;
2513:S
2514:D        1 import sysglobals, asm, loader, misc, fs, initunits, ldr;
2515:S
2516:D        1 var keyboard: text;
2517:D        1     listing:  text;
2518:S
2519:D    -4  1     i: integer;
2520:S
2521:D        1 procedure initfiles;
2522:C        2 begin
2523:C        2   {INPUT    }  new(gfiles[0]);
2524:C        2   {OUTPUT   }  new(gfiles[1]);
2525:C        2   {KEYBOARD }  new(gfiles[2]);
2526:C        2   {LISTING  }  new(gfiles[4]);
2527:C        2 end;  (*INITFILES*)
2528:D    -4  1
2529:C        1 procedure dummydebug(p1,p2,p3: integer); begin end;
2530:C        1 procedure dummycleario; begin end;
2531:S
```



```
2532:S
2533:S
2534:C        1 begin
2535:S
2536:C        1 initvects;              {SET UP VECTORS, INTERRUPT TABLE, ETC.}
2537:C        1
2538:S          {*****************************************************************************
2539:C        1  MISCELLANEOUS LOW LEVEL STUFF}
2540:S
2541:C        1    idle:=250;                { 3.0 bug fix -- assume old char rom  3/20/84}
2542:C        1    locklevel := 0; actionspending := 0;
2543:C        1    debugger := dummydebug;
2544:C        1    cleariohook := dummycleario;
2545:C        1    gfiles[1] := nil;
2546:S
2547:S          {*****************************************************************************
2548:C        1  INITIALIZE SUFFIX AND ACCESS METHOD TABLES}
2549:S
2550:C        1    initfilekinds;
2551:S
2552:S          {*****************************************************************************
2553:S           initialize ISR's,
2554:S                     DEVICE DRIVERS,
2555:S                     ACCESS METHODS,
2556:S                     DIRECTORY ACCESS,
2557:S                     DEFAULT UNIT TABLE,
2558:C        1            CLEAR UNITS }
2559:S
2560:C        1    unitioinit;
2561:S
2562:S          {*****************************************************************************
2563:C        1  FIND SYSTEM AND DEFAULT VOLUME NAMES}
2564:S
2565:C        1    initsysunit;
2566:S
2567:S          {*****************************************************************************
2568:C        1  create files INPUT, OUTPUT, KEYBOARD and LISTING}
2569:S
2570:C        1    initfiles;
2571:C        1    openfiles;
2572:S
2573:S          {*****************************************************************************
2574:C        1  INITIALIZE LOADER}
2575:C        1
2576:C        1    markuser;
2577:C        1    syslibrary := '*LIBRARY';
2578:C        1
2579:C        1  end.  {module initsys}
2580:S
2581:C        1
```

No errors. 13 warnings.
***** Nonstandard language features enabled *****

INITLOAD

Description

INITLOAD is system-wide type and variable declarations, interface to internal minifloppy driver, boot-load routines, the linking loader and the boot-up control program.

Usage

INITLOAD is part of SYSTEM_P.

Notes

Includes: GLOBALS, MINI, BOOTDAM, LOADER and INITLOAD.

```
Pascal [Rev 3.0M  6/ 4/84] INITLOAD.TEXT          26-Dec-84 21:01:21 Page 1   *** File name: INITLOAD ***

     1:D        0 $modcal, range off, ovflcheck off, debug off, iocheck off, stackcheck off$
     2:D        0 program initload;
     3:S
     4:D        1 $search  'ASM'    $
     5:S
     5:D        1 $include 'GLOBALS'$;
     6:D        1 module sysglobals;
     7:S
     8:D        1 export
     9:S
    10:D        1 type fsidctype = packed array[1..20] of char;
    11:D        1 const
    12:D        1   fsidc = fsidctype['Rev. 3.01  20-Sep-84'];
    13:D        1                          {*20 CHARS: VERSION,DATE,TIME OF FILE SYS*)
    14:D        1   mminint = -32768;      {*MINIMUM SHORT INTEGER VALUE*)
    15:D        1   mmaxint =  32767;      {*MAXIMUM SHORT INTEGER VALUE*)
    16:D        1   maxunit = 50;          {*MAXIMUM PHYSICAL UNIT NUMBER*)
    17:D        1   passleng = 16;         {*NUMBER OF CHARS IN A PASSWORD*)
    18:D        1   vidleng = 16;          {*NUMBER OF CHARS IN A VOLUME NAME*)
    19:D        1   tidleng = 16;          {*NUMBER OF CHARS IN A FILE TITLE*)
    20:D        1   fidleng = 120;         {*NUMBER OF CHARS IN FILE NAME*)
    21:D        1   fblksize = 512;        {*STANDARD FILE BUFFER LENGTH*)
    22:D        1   maxsc=63;              {*LARGEST SELECT CODE *)
    23:D        1   minlevel=1;            {*LOWEST INTERRUPT LEVEL*)
    24:D        1   maxlevel=6;            {*LARGEST MASKABLE INTERRUPT LEVEL*)
    25:S
    26:S
    27:D        1 type
    28:S
    29:D        1   byte = 0..255;
    30:D        1   shortint = mminint..mmaxint;
    31:D        1   ipointer = ^integer;
    32:D        1   charptr = ^char;
    33:D        1   textptr = ^text;
    34:D        1   string80 = string[80];
    35:D        1   string255 = string[255];
    36:D        1   suffixtype = string[5];
    37:D        1   string255ptr = ^string255;
    38:S
    39:S
    40:D        1   unitnum = 0..maxunit;
    41:D        1   vid = string[vidleng];
    42:D        1   tid = string[tidleng];
    43:D        1   fid = string[fidleng];
    44:D        1   passtype = string[passleng];
    45:S
    46:D        1   filekind = (untypedfile,      {directory entry}
    47:D        1               badfile,          {bad blocks}
    48:D        1               codefile,         {executable or linkable}
    49:D        1               textfile,         {UCSD format text with editor environment}
    50:D        1               asciifile,        {L.I.F. ASCII format text strings}
    51:D        1               datafile,         {file of  <data type, e.g. char, integer,etc.>}
    52:D        1               sysfile,          {system (BOOT) file}
    53:D        1      fkind7,  fkind8,  fkind9,
    54:D        1      fkind10, fkind11, fkind12,
    55:D        1      fkind13, fkind14, lastfkind);      {reserved for future expansion}
    56:S
    57:S
    58:D        1                                                          (*FILE INFORMATION*)
    59:S
```

```
Pascal [Rev 3.0M  6/ 4/84] GLOBALS.TEXT           26-Dec-84 21:01:21 Page 2   *** File name: INITLOAD ***

    60:D        1 window = packed array [0..maxint] of char;
    61:D        1 windowp = ^window;
    62:S
    63:S
    64:D        1 fibp = ^fib;
    65:S
    66:D        1 amrequesttype = (readbytes, writebytes, flush, writeeol, readtoeol, clearunit,
    67:D        1          setcursor, getcursor, startread, startwrite, unitstatus, seekeof);
    68:D        1 amtype = procedure (fp: fibp; request: amrequesttype;
    69:D        1                     anyvar buffer: window; bufsize, position: integer);
    70:D        1 eotproc = procedure(fp: fibp);
    71:S
    72:D        1 fib = packed record
    73:D        1             fwindow: windowp;                          (*BUFFER VARIABLE...F^ *)
    74:D        1             flistptr: fibp;                             (* LIST OF OPEN FILES *)
    75:D        1         {declaration and type information}
    76:D        1             frecsize: integer;                  (* SIZE OF ONE LOGICAL RECORD *)
    77:D        1             feft: shortint;                             (* EXTERNAL FILE TYPE *)
    78:D        1             fkind: filekind;                                (* FILE KIND *)
    79:D        1             fistextvar:  boolean;                  (* FILE IS LINE FORMATTED *)
    80:D        1             fbuffered:   boolean;                (* HAS 512 BYTE BLOCK BUFFER *)
    81:D        1             fanonymous: boolean;                          (* FILE HAS NO NAME *)
    82:D        1             fisnew:  boolean;                  (* WAS CREATED THIS ASSOCIATION *)
    83:D        1             freadable, fwriteable: boolean;             (* FILE ACCESS RIGHTS *)
    84:D        1         {state information}
    85:D        1             freadmode, fbufvalid: boolean;          (*F^ AND LOOKAHEAD STATES *)
    86:D        1             feoln: boolean;                           (* F^ IS AN END OF LINE *)
    87:D        1             feof: boolean;                  (* TRIED TO READ PAST END OF FILE *)
    88:D        1             fmodified: boolean;                       (* FILE HAS CHANGED SIZE *)
    89:D        1             fbufchanged: boolean;               (* BUFFER NEEDS TO BE WRITTEN *)
    90:D        1         {file size and position}
    91:D        1             fpos:  integer;           (* FILE POINTER, CURRENT FILE POSITION *)
    92:D        1             fleof: integer;         (*LOGICAL END OF FILE, CURRENT FILE SIZE *)
    93:D        1             fpeof: integer;        (*PHYSICAL END OF FILE, MAXIMUM FILE SIZE *)
    94:D        1         {buffering and low level formatting information}
    95:D        1             flastpos:  integer;                   (* FILE POSITION OF BUFFER *)
    96:D        1             freptcnt:  shortint;                  (* SPACE COMPRESSION COUNT *)
    97:D        1             am:  amtype;                              (*BUFFER METHOD MODULE *)
    98:D        1         {file association info}
    99:D        1             fstartaddress: integer;          (*EXECUTION ADDRESS IN BOOT FILE *)
   100:D        1             fvid: vid;                                        (* VOLUME NAME *)
   101:D        1             ffpw: passtype;                                (* FILE PASSWORD *)
   102:D        1             ftid: tid;                                        (* FILE NAME *)
   103:D        1             pathid:  integer;     (* ADDITIONAL SYSTEM DEPENDENT INFORMATION *)
   104:D        1             fanonctr: shortint;                           (* TEMP FILE IDENTIFIER *)
   105:D        1             foptstring: string255ptr;                  (* OPTIONAL STRING PARAM *)
   106:D        1         {byte block transfer information}
   107:D        1             fileid:  integer; (* START BYTE OF FILE, OR OTHER IDENTIFICATION *)
   108:D        1             fb0,fb1,                                      (* FOR FUTURE EXPANSION *)
   109:D        1             fnosrmtemp,                        (*TRUE IF NO SRM TEMP FILE CREATED *)
   110:D        1             fwaitonlock,                       (*TRUE IF SRM SHOULD WAIT FOR LOCK *)
   111:D        1             fpurgeoldlink,                (*TRUE IF OLD SRM LINK IS TO BE PURGED *)
   112:D        1             foverwritten,                      (*TRUE IF OPENED WITH OVERWRITE *)
   113:D        1             fsavepathid,                  (*TRUE IF PATHID NOT UNIQUE TO FILEID *)
   114:D        1             flockable,                       (*TRUE IF FILE OPENED AS LOCKABLE *)
   115:D        1             flocked,                                 (*TRUE IF FILE IS LOCKED *)
   116:D        1             fbusy        : boolean;                 (*TRUE IF DRIVER IS ACTIVE *)
   117:D        1             funit        : unitnum;                      (*PHYSICAL UNIT NUMBER *)
   118:D        1             feot         : eotproc;          (*CALLED WHEN TRANSFER COMPLETES *)
   119:D        1             fxpos        : integer;                  (* X POSITION FOR  GOTOXY *)
```

```
   120:D        1            fypos       : integer;              (* Y POSITION FOR  GOTOXY *)
   121:D        1            foldfileid   : integer;    (* FILEID FOR OLD SRM FILE ON REWRITE *)
   122:D        1        {for future expansion}
   123:D        1            fextra:  array[0..2] of integer;
   124:D        1            fextra2: shortint;
   125:D        1        {large miscellaneous fields sometimes present}
   126:D        1            case integer of
   127:D        1              0: ( {minimal FIB ends here} );
   128:D        1              1: (ftitle:  fid);        (* FILE NAME, EXCEPT VOLUME AND SIZE *)
   129:D        1              2: (fbuffer: packed array [0..fblksize-1] of char);
   130:D        1         end (*FIB*) ;
   131:S
   132:S
   133:D        1 damrequesttype = (getvolumename, setvolumename, getvolumedate, setvolumedate,
   134:D        1  changename, purgename,
   135:D        1  openfile, createfile, overwritefile, closefile, purgefile,
   136:D        1  stretchit, makedirectory, crunch, opendirectory, closedirectory, catalog,
   137:D        1  stripname, setunitprefix, openvolume, duplicatelink, openparentdir,
   138:D        1  catpasswords,setpasswords,lockfile,unlockfile,openunit);
   139:S
   140:D        1 damtype = procedure (anyvar f: fib; unum:unitnum; request: damrequesttype);
   141:S
   142:D        1 unitentry =  {unitable entry definition}
   143:D        1     packed record
   144:D        1       dam: damtype;                    {directory access method}
   145:D        1       tm:  amtype;                     {byte block transfer method}
   146:D        1       sc: byte;                        {select code}
   147:D        1       ba: byte;                        {bus address}
   148:D        1       du: byte;                        {disc unit}
   149:D        1       dv: byte;                        {disc volume}
   150:D        1       byteoffset: integer;             {physical starting byte of volume}
   151:D        1       devid: integer;                  {identifier (Amigo identify sequence)}
   152:D        1       uvid: vid;                       {volume id}
   153:D        1       dvrtemp: integer;                {temp for driver use only; init to 0!}
   154:D        1       dvrtemp2: shortint;              {temp for driver use only; init to 0!}
   155:D        1       letter: char;                    {device specifier letter}
   156:D        1       offline,                         {unit absent or down flag}
   157:D        1       uisinteractive,                  {user can edit input}
   158:D        1       umediavalid,                     {medium not changed since last access}
   159:D        1       uuppercase:   boolean;           {volume name must be uppercased}
   160:D        1       uisfixed: boolean;               {fixed/removeable media flag}
   161:D        1       ureportchange: boolean;          {driver mode: report/ignore media change}
   162:D        1       pad: 0..1;                       {    (bit not used yet)  }
   163:D        1       case uisblkd: boolean of         {blocked volume flag}
   164:D        1         true: (umaxbytes: integer)     {volume size in bytes }
   165:D        1     end;  {unitentry}
   166:S
   167:D        1 unitabletype = array [unitnum] of unitentry;                  (*0 NOT USED*)
   168:D        1 amtabletype  = array[filekind] of amtype;
   169:D        1 suftabletype = array[filekind] of suffixtype;
   170:D        1 efttabletype = array[filekind] of shortint;
   171:S
   172:D        1 unitableptr     = ^unitabletype;
   173:D        1 amtableptrtype  = ^amtabletype;
   174:D        1 suftableptrtype = ^suftabletype;
   175:D        1 efttableptrtype = ^efttabletype;
   176:S
   177:S
   178:D        1 iorsltwd =
   179:S                  { *note* the ioresult enumerations have been partitioned into two
```



```
   180:S                       mutually-exclusive groups: those beginning with 'z' are reserved
   181:S                       for the low-level drivers , and those beginning
   182:D        1              with 'i' are reserved for the higher-level routines.)
   183:S
   184:D        1             (inoerror,zbadblock,ibadunit,zbadmode,ztimeout,
   185:D        1              ilostunit,ilostfile,ibadtitle,inoroom,inounit,
   186:D        1              inofile,idupfile,inotclosed,inotopen,ibadformat,
   187:D        1              znosuchblk,znodevice,zinitfail,zprotected,
   188:D        1              zstrangei,zbadhardware,zcatchall,zbaddma,
   189:D        1              inotvalidsize,inotreadable,inotwriteable,inotdirect,
   190:D        1              idirfull,istrovfl,ibadclose,ieof,
   191:D        1              zuninitialized,znoblock,znotready,znomedium,
   192:D        1              inodirectory,ibadfiletype,ibadvalue,icantstretch,
   193:D        1              ibadrequest, inotlockable, ifilelocked, ifileunlocked,
   194:D        1              idirnotempty, itoomanyopen, inoaccess, ibadpass, ifilenotdir,
   195:D        1              inotondir, ineedtempdir, isrmcatchall, zmediumchanged,
   196:D        1              {end marker} endioerrs);
   197:S
   198:D        1 proctype=procedure;
   199:D        1 sctype=0..maxsc;
   200:D        1 leveltype=minlevel..maxlevel;
   201:D        1 pisrib = ^isrib;
   202:D        1 isrproctype = procedure(isribptr: pisrib);
   203:D        1 isrib = {isr information block}
   204:D        1    packed record
   205:D        1      intregaddr: charptr;       {interrupt register address}
   206:D        1      intregmask: byte;          {interrupt register mask}
   207:D        1      intregvalue: byte;         {interrupt register target value after masking}
   208:D        1      chainflag: boolean;        {chaining flag}
   209:D        1      proc: isrproctype;         {isr}
   210:D        1      link: pisrib;              {pointer to next isrib in linked list}
   211:D        1    end;
   212:D        1
   213:D        1 inttabletype = array [1..7] of pisrib;
   214:S
   215:D        1 action = procedure;
   216:S
   217:D        1 daterec          = packed record
   218:D        1                      year        : 0..100;       (*100 IS TEMP DISK FLAG*)
   219:D        1                      day         : 0..31;        (*DAY OF MONTH*)
   220:D        1                      month       : 0..12;        (*0 ==> DATE NOT MEANINGFUL*)
   221:D        1                    end;
   222:S
   223:D        1 timerec          = packed record
   224:D        1                      hour        : 0..23;
   225:D        1                      minute      : 0..59;
   226:D        1                      centisecond : 0..5999;
   227:D        1                    end;
   228:S
   229:D        1 datetimerec      = packed record
   230:D        1                      date        : daterec;
   231:D        1                      time        : timerec;
   232:D        1                    end;
   233:S
   234:D        1 var
   235:S          (********** MANY OF THE FOLLOWING HAVE HARDCODED
   236:D        1             ADDRESSES IN COMPILER AND ELSEWHERE  ======*)
   237:S
   238:D        1 (*** ERROR RECOVERY  ***)
   239:D    -2  1   sysescapecode:        shortint;
```

```
  240:D    -6  1   openfileptr:  anyptr;
  241:D   -10  1   recoverblock: anyptr;
  242:S
  243:D   -10  1 (*** MEMORY MANAGEMENT ***)
  244:D   -14  1   heapmax:      anyptr;
  245:D   -18  1   heapbase:     anyptr;
  246:S
  247:D   -18  1 (*** I/O DRIVERS ***)
  248:D   -22  1   sysioresult:          integer;
  249:D   -26  1   hardwarestatus:       integer;
  250:D   -30  1   locklevel:      integer;
  251:D   -34  1   unitable: unitableptr;
  252:D   -62  1   interrupttable: inttabletype;
  253:D   -66  1   endisrhook:  integer;
  254:D   -70  1   actionspending: integer;
  255:S
  256:D   -70  1 (*** FILE SYSTEM ***)
  257:D   -94  1   gfiles: array[0..5] of textptr;
  258:S            {   [0] INPUT          [3] (unused)
  259:S                [1] OUTPUT         [4] LISTING
  260:D   -94  1       [2] KEYBOARD       [5] (unused)   }
  261:S
  262:D   -98  1   amtable:     amtableptrtype;                  {pointer to access methods }
  263:D  -102  1   suffixtable: suftableptrtype;                 {pointer to list of suffixes}
  264:D  -106  1   efttable:    efttableptrtype;                 {pointer to LIF file types}
  265:D  -110  1   sysunit: integer;
  266:D  -146  1   syvid,dkvid: vid;                            (*SYSUNIT VOLID & DEFAULT VOLID*)
  267:D  -268  1   syslibrary: fid;
  268:S
  269:D  -268  1   (*** DEBUGGER HOOK ***)
  270:D  -276  1   debugger: procedure(p1, p2, p3: integer);
  271:S
  272:D  -276  1   (*** CLEAR I/O HOOK ***)
  273:D  -284  1   cleariohook : procedure;
  274:S
  275:D  -312  1   perminttable: inttabletype;
  276:S
  277:D  -392  1   deferredaction: array[1..10] of action;
  278:S
  279:D  -400  1   serialtextamhook: amtype;            {access method for serial devices}
  280:S
  281:D  -400  1   sysname[-574]: packed array[1..10] of char;
  282:S
  283:D  -400  1   sysflag[-302]: packed record
  284:D  -400  1     reserved1, reserved2, nointhpib, crtconfigreg,
  285:D  -400  1     nokeyboard, highlightsxorbiggraphics, biggraphics,
  286:D  -400  1     alpha50: boolean;
  287:D  -400  1     end;
  288:S
  289:D  -400  1   sysflag2[hex('FFFEDA')]: packed record
  290:D  -400  1                               pad7to1    : 0..127;
  291:D  -400  1                               prompresent: boolean;
  292:D  -400  1                             end;
  293:S
  294:S            endsysvars : shortint;   {must be last thing
  295:D  -402  1                             in SYSGLOBALS}
  296:S
  297:D  -402  1 implement
  298:C        1 end
  299:C        2 $include 'GLOBALS'$;
```



```
  299:D        1 $include 'MINI'   $;
  300:D        1 module mini;
  301:S
  302:S
  303:D        1 import
  304:D        1   sysglobals, asm (, misc);
  305:S
  306:S
  307:D        1 export
  308:S
  309:D        1   procedure miniio(fp: fibp; request: amrequesttype; anyvar buffer: window;
  310:D        2                                                      length, position: integer);
  311:D        1   procedure ioresc(ioresult_value: iorsltwd);
  312:S
  313:D        1   function iorval: iorsltwd;
  314:S
  315:D        1 implement
  316:S
  317:S
  318:D        1 var
  319:D    -2  1   most_recent_escapecode: shortint;  {for post-mortem diagnostic purposes only!!!}
  320:S
  321:S
  322:D        1 procedure ioresc(ioresult_value: iorsltwd);
  323:C        2   begin (ioresc)
  324:C        2     ioresult := ord(ioresult_value);
  325:C        2     escape(-10)
  326:C        2   end; (ioresc)
```

```
   327:D    -2  1 $page$
   328:S
   329:D        1 function iorval: iorsltwd;
   330:S
   331:D        2   type
   332:D        2     te = {table entry type}
   333:D        2       record
   334:D        2         b{basic error}: shortint;
   335:D        2         p{pascal error}: iorsltwd
   336:D        2       end;
   337:D        2     ect_type = {error conversion table type}
   338:D        2       array[1..29] of te;
   339:S
   340:D        2   const
   341:D        2     error_conversion_table = ect_type[
   342:D        2       te[b: 1066, p:zinitfail      ],     {bad track 0, side 0            }
   343:D        2       te[b: 2066, p:zinitfail      ],     {more than 4 spares             }
   344:D        2       te[b: 3066, p:zbadhardware   ],     {write fault or lost data       }
   345:D        2       te[b: 4066, p:znomedium      ],     {timeout during init            }
   346:D        2       te[b: 1080, p:znomedium      ],     {no media or door open          }
   347:D        2       te[b: 2080, p:znomedium      ],     {no media or door open          }
   348:D        2       te[b: 3080, p:znomedium      ],     {no media or door open          }
   349:D        2       te[b: 8080, p:zmediumchanged ],     {media changed                  }
   350:D        2       te[b: 9080, p:znomedium      ],     {media changed during operation }
   351:D        2       te[b: 1081, p:znoblock       ],     {track not found                }
   352:D        2       te[b: 2081, p:zbadhardware   ],     {restore error                  }
   353:D        2       te[b: 3081, p:zbadhardware   ],     {track 0 not found after reset  }
   354:D        2       te[b: 4081, p:zbadhardware   ],     {read lost data error           }
   355:D        2       te[b: 5081, p:zbadhardware   ],     {write lost data error or fault }
   356:D        2       te[b: 6081, p:zbadhardware   ],     {address lost data error        }
   357:D        2       te[b: 7081, p:znoblock       ],     {address CRC error during write }
   358:D        2       te[b: 1083, p:zprotected     ],     {write protect error            }
   359:D        2       te[b: 2083, p:zprotected     ],     {write protect error            }
   360:D        2       te[b: 1084, p:znoblock       ],     {read record not found/d bit set}
   361:D        2       te[b: 2084, p:znoblock       ],     {write record not found         }
   362:D        2       te[b: 3084, p:znoblock       ],     {address (track) not found      }
   363:D        2       te[b: 1087, p:znoblock       ],     {address CRC error              }
   364:D        2       te[b: 1088, p:zbadblock      ],     {read CRC error                 }
   365:D        2       te[b: 6090, p:zstrangei      ],     {unexpected interrupt           }
   366:D        2       te[b: 7090, p:zstrangei      ],     {INT during wrt trk handshaking }
   367:D        2       te[b: 8090, p:znomedium      ],     {timeout waiting for interrupt  }
   368:D        2       te[b: 9090, p:zcatchall      ],     {INT mask > 2 (flpy locked out) }
   369:D        2       te[b:11090, p:zbadhardware   ],     {timeout; floppy not responding }
   370:D        2       te[b: 1082, p:znodevice      ] ];   {2nd drive not present         }
   371:S
   372:D        2   var
   373:D    -2  2     i: shortint;
   374:S
   375:C        2   begin {iorval}
   376:C        2     most_recent_escapecode := escapecode;  {for post-mortem diagnostic purposes only!!!}
   377:C        2     iorval := inoerror;  {value returned if unrecognized}
   378:C        2     for i := 1 to 29 do
   379:C        3       with error_conversion_table[i] do
   380:C        4         if escapecode=b then iorval := p;
   381:C        2   end; {iorval}
```



```
   382:D    -2  1 $page$
   383:S
   384:D        1 procedure miniio(fp: fibp; request: amrequesttype; anyvar buffer: window;
   385:D        2                                                    length, position: integer);
   386:D        2   const
   387:D        2     bootrev1 = chr(255);
   388:S
   389:D        2   var
   390:D        2     bootrevflag[318]: char;
   391:D        2     minidrive[-301]: packed record num: 0..255 end;
   392:D    -4  2     buf: charptr;
   393:D    -8  2     uep: ^unitentry;
   394:S
   395:D        2   procedure clear_unit;
   396:C        3     begin {clear_unit}
   397:C        3       try
   398:C        4         f_pwr_on;
   399:C        4       recover
   400:C        4         if iorval<>inoerror
   401:C        5           then ioresc(iorval)
   402:C        5           else escape(escapecode);
   403:C        3     end; {clear_unit}
   404:S
   405:S
   406:D        2   procedure transfer(sector: integer);
   407:S
   408:D        3     var
   409:D    -1  3       transfer_successful: boolean;
   410:D    -6  3       whole_sectors: integer;
   411:D   -10  3       whole_sector_bytes: integer;
   412:D   -10  3       fubuffer[-558]: packed array[0..255] of byte;
   413:D   -12  3       saved_escapecode: shortint;
   414:D   -14  3       ior: iorsltwd;
   415:S
   416:C        3     begin {transfer}
   417:S
   418:C        3       repeat
   419:C        4         try
   420:S
   421:C        5           whole_sectors := length div 256;
   422:C        5           if whole_sectors>0 then
   423:C        6             begin
   424:C        6               case request of
   425:C        7                 readbytes, startread:
   426:C        7                   flpymread(whole_sectors, sector, buf^);
   427:C        7                 writebytes, startwrite:
   428:C        7                   flpymwrite(whole_sectors, sector, buf^);
   429:C        7               end; {case}
   430:C        6               sector := sector+whole_sectors;
   431:C        6               whole_sector_bytes := whole_sectors*256;
   432:C        6               buf := addr(buf^, whole_sector_bytes);
   433:C        6               length := length-whole_sector_bytes;
   434:C        6             end; {if}
   435:S
   436:C        5           if length>0 then
   437:C        6             case request of
   438:C        7               readbytes, startread:
   439:C        7                 begin
   440:C        7                   flpyread(sector, fubuffer);
   441:C        7                   moveleft(fubuffer, buf^, length);
```

```
442:C  7                end;
443:C  7              writebytes, startwrite:
444:C  7                begin
445:C  7                  moveleft(buf^, fubuffer, length);
446:C  7                  flpy_wrt(sector, fubuffer);
447:C  7                end;
448:C  7            end; {case}
449:S
450:C  5          transfer_successful := true;
451:S
452:C  5        recover
453:C  5          begin
454:C  5            ior := iorval;
455:C  5            if ior=inoerror then {unrecognized escapecode}
456:C  6              begin
457:C  6                saved_escapecode := escapecode;
458:C  6                try
459:C  7                  f_pwr_on;
460:C  7                recover
461:C  7                  {do nothing};
462:C  6                escape(saved_escapecode);
463:C  6              end
464:C  6            else if ior=zmediumchanged then
465:C  7              begin
466:C  7                uep^.umediavalid := false;
467:C  7                if uep^.ureportchange then
468:C  8                  ioresc(ior);
469:C  7                transfer_successful := false;  {retry}
470:C  7              end {with}
471:C  7            else
472:C  7              ioresc(ior);
473:C  5          end; {recover}
474:S
475:C  4      until transfer_successful;
476:S
477:C  3    end; {transfer}
```



```
478:D   -8  2  $page$
479:S
480:C       2  begin {miniio}
481:C       2    uep := addr(unitable^[fp^.funit]);
482:C       2    if uep^.offline then
483:C       3      ioresult := ord(znodevice)
484:C       3    else
485:C       3      begin
486:C       3        { lockup; }
487:C       3        try
488:C       4          if bootrevflag=bootrev1 then
489:C       5            if uep^.du=0 then {ok}
490:C       6            else ioresc(znodevice)
491:C       6          else
492:C       5            if uep^.du in [0,1] then minidrive.num := uep^.du
493:C       6            else ioresc(znodevice);
494:S
495:C       4          case request of
496:C       5            clearunit:
497:C       5              clear_unit;
498:S
499:C       5            unitstatus:
500:C       5              fp^.fbusy := false;
501:S
502:C       5            flush:
503:C       5              {do nothing};
504:S
505:C       5            readbytes, writebytes, startread, startwrite:
506:C       5              begin
507:C       5                if uep^.ureportchange and not uep^.umediavalid then
508:C       6                  ioresc(zmediumchanged);
509:C       5                buf := addr(buffer);
510:C       5                if (position mod 256<>0) or odd(integer(buf)) then
511:C       6                  ioresc(zbadmode);
512:C       5                if (position<0) or (length<0) or (position+length>fp^.fpeof) then
513:C       6                  ioresc(ieof);
514:C       5                transfer((position+fp^.fileid+uep^.byteoffset) div 256);
515:C       5              end;
516:S
517:C       5            otherwise
518:C       5              ioresc(ibadrequest);
519:C       5          end; {case}
520:S
521:C       4          ioresc(inoerror);  {set ioresult & perform lockdown}
522:C       4        recover
523:C       4          begin
524:C       4            { lockdown; }
525:C       4            if escapecode<>-10 then escape(escapecode);
526:C       4            if request in [startread, startwrite] then call (fp^.feot, fp);
527:C       4          end; {recover}
528:C       3      end; {else}
529:C       2  end; {miniio}
530:S
531:C       1 end {mini}
532:S
533:C       2 $include 'MINI'   $;
534:D       1 $include 'BOOTDAM'$;
535:D       1 module bootdammodule;
536:D       1 import sysglobals,asm,mini;
          1 export
```

```
   537:S
   538:D        1   type string2=string[2];
   539:S
   540:D    -52 1   var nodestr: string[50];
   541:S
   542:D    -52 1   procedure initbootdam;
   543:D    -52 1   procedure bootdam(anyvar f:fib; unum:unitnum; request:damrequesttype);
   544:D        1   procedure boottm(f:fibp; request: amrequesttype;
   545:D    -52 2                        anyvar buffer: window; bufsize,position: integer);
   546:D    -52 1   function srmnode(sc: shortint): string2;
   547:D    -52 1   function bootnode: string2;
   548:S
   549:D    -52 1 implement
   550:D    -52 1   type
   551:S
   552:D    -52 1     strptr = ^string255;
   553:D    -52 1     lifname  = packed array[1..10] of char;
   554:D    -52 1     word15   = 0..32767;
   555:D    -52 1     bcd      = 0..15;
   556:D    -52 1     tdate    = packed array[1..12] of bcd;
   557:D    -52 1     direntryp= ^direntry;
   558:D    -52 1     direntry = packed record
   559:D    -52 1                  fname    : lifname;
   560:D    -52 1                  ftype    : shortint;
   561:D    -52 1                  fstart   : integer;
   562:D    -52 1                  fsize    : integer;
   563:D    -52 1                  fdate    : tdate;
   564:D    -52 1                  lastvol  : boolean;
   565:D    -52 1                  volnumber: word15;
   566:D    -52 1                  extension: integer;
   567:D    -52 1                end;
   568:S
   569:D    -52 1     msustype = packed record
   570:D    -52 1                  mtype : byte;
   571:D    -52 1                  munit : byte;
   572:D    -52 1                  mscode: byte;
   573:D    -52 1                  maddr : byte;
   574:D    -52 1                end;
   575:D    -52 1   var
   576:D    -52 1     driverkey  [-293  {$FFFFFEDB}]: boolean;
   577:D    -52 1     boot_msus  [-292  {$FFFFFEDC}]: msustype;
   578:D    -52 1     boot_id    [16382 {$    3FFE}]: shortint;
   579:D    -52 1     boot_flags [16380 {$    3FFC}]: packed array[0..7] of boolean;
   580:S
   581:D    -53 1     gread        : boolean;
   582:S
   583:D    -53 1     {  FUNCTIONS AND PROCEDURES USED FROM THE BOOT ROM  }
   584:D    -53 1   procedure boot_lifhead(var dsize,dstart:integer); external;
   585:D        1   procedure boot_findfile(var entry:direntryp; var lname:lifname;
   586:D    -53 2                                                  dl,ds : integer); external;
   587:D    -53 1   function boot_minit(msus:msustype):boolean; external;
   588:D        1   function boot_mfopen(anyvar fname:string255; var xaddr,length:integer;
   589:D    -53 2                          var ftype:shortint):boolean; external;
   590:D    -53 1   procedure boot_mfclose; external;
   591:D        1   procedure boot_mread(sector,bytecount: integer;
   592:D    -53 2                        anyvar buffer:window; media:boolean); external;
   593:S
   594:D        1   function escio(escn:integer):iorsltwd;
   595:C        2   begin
   596:C        2     case escn of
```



```
   597:C        3     1,7 : escio:=znodevice;
   598:C        3     2   : escio:=znomedium;
   599:C        3     3   : escio:=znotready;
   600:C        3     4   : escio:=zbadblock;
   601:C        3     5   : escio:=zbadhardware;
   602:C        3     6   : escio:=zcatchall;
   603:C        3     otherwise  escape(escn);
   604:C        3     end; { case }
   605:C        2   end;
   606:S
   607:D        1   procedure bootdam(anyvar f:fib; unum:unitnum; request:damrequesttype);
   608:D        2   var
   609:D     -2 2     ftype       : shortint;
   610:D     -4 2     fk          : filekind;
   611:D     -8 2     dstart      : integer;
   612:D    -12 2     dsize       : integer;
   613:D    -16 2     dentry      : direntryp;
   614:D    -24 2     i,j         : integer;
   615:S
   616:D        2     procedure escesc(ecode:integer);
   617:D        3     var
   618:D     -4 3       tempec      : integer;
   619:C        3     begin { convert mini driver escape code to general code}
   620:C        3       if ecode<=0 then escape(ecode);
   621:C        3       if ecode = 1 then ioresult := ord(ibadtitle);
   622:C        3       tempec:=ecode mod 1000;
   623:C        3       if tempec=80 then escape(2)     { no medium }
   624:C        4       else if tempec=90 then escape(6){ bad error state }
   625:C        5            else escape(4);            { read error }
   626:C        3     end;
   627:S
   628:D        2     function minit(msus:msustype):boolean;
   629:D        3     var
   630:D     -4 3       tempec              : integer;
   631:C        3     begin
   632:C        3       if gread then minit:=boot_minit(msus)
   633:C        4       else
   634:C        4       begin
   635:C        4       try
   636:C        5         minit:=true;
   637:C        5         driverkey := false;
   638:C        5         boot_lifhead(dsize,dstart);
   639:C        5         driverkey := true;
   640:C        5       recover begin
   641:C        5                 driverkey := true;
   642:C        5                 if escapecode=-1 then minit:=false
   643:C        6                                  else escesc(escapecode);
   644:C        5                 end;
   645:C        4       end;
   646:C        3     end;
   647:S
   648:D        2     function mfopen(anyvar fname:string255; var xaddr,length:integer;
   649:D        3                     var ftype:shortint):boolean;
   650:D        3     var
   651:D    -10 3       tfname      : lifname;
   652:D    -14 3       i           : integer;
   653:D    -18 3       fk, fkind   : filekind;
   654:C        3     begin
   655:C        3       if gread then mfopen:=boot_mfopen(fname,xaddr,length,ftype)
   656:C        4       else
```

```
657:C  4       begin
658:C  4       try
659:C  5         mfopen:=true;
660:C  5         if (strlen(fname)=0) or (strlen(fname)>10) then escape(1);
661:C  5         for i:=1 to 10 do
662:C  6           if i>strlen(fname) then tfname[i]:=' ' else tfname[i]:=fname[i];
663:S
664:C  5         driverkey := false;
665:C  5         boot_findfile(dentry,tfname,dsize,dstart);
666:C  5         driverkey := true;
667:S
668:C  5         fkind:=datafile;  ftype:=dentry^.ftype;
669:C  5         for fk:=lastfkind downto untypedfile do
670:C  6           if efttable^[fk]=ftype then fkind:=fk;
671:C  5         if fkind=sysfile then xaddr:=dentry^.extension
672:C  6                          else xaddr:=0;
673:C  5         if fkind=datafile then length:=dentry^.extension
674:C  6                           else length:=dentry^.fsize*256;
675:C  5       recover
676:C  5         begin
677:C  5           driverkey := true;
678:C  5           mfopen:=false;
679:C  5           if escapecode<>-1 then escesc(escapecode);
680:C  5         end;
681:C  4       end;
682:C  3     end;
683:S
684:D  2     procedure mfclose;
685:C  3     begin
686:C  3       if gread then boot_mfclose;
687:C  3     end;
688:S
689:C  2   begin { bootdam }
690:C  2     ioresult:=ord(inoerror);
691:C  2     case request of
692:C  3     getvolumename:  strptr(addr(f))^ := '"BOOT_DEVICE"';
693:C  3     openfile:
694:C  3       try
695:C  4         j := 0;
696:C  4         for i := 1 to strlen(f.ftitle) do if f.ftitle[i] = '/' then j := i;
697:C  4         if strlen(f.ftitle)-j>tidleng then f.ftid := ''
698:C  5         else begin
699:C  5              setstrlen(f.ftid, strlen(f.ftitle)-j);
700:C  5              for i := 1 to strlen(f.ftid) do f.ftid[i] := f.ftitle[j+i];
701:C  5              end;
702:C  4         if minit(boot_msus) then
703:C  5         begin
704:C  5           ftype:=-1;
705:C  5           if mfopen(f.ftitle,f.fstartaddress,f.fpeof,ftype) then
706:C  6           begin
707:C  6             f.fkind:=datafile;
708:C  6             for fk:=lastfkind downto untypedfile do
709:C  7                   if efttable^[fk]=ftype then f.fkind:=fk;
710:C  6             if gread then f.fileid := 0
711:C  7                      else f.fileid := dentry^.fstart*256;
712:C  6             f.fleof:=f.fpeof;       f.fisnew:=false;
713:C  6             if not f.fbuffered then f.am:=amtable^[UNTYPEDFILE]
714:C  7             else f.am:=amtable^[f.fkind];
715:C  6           end
716:C  6           else ioresult:=ord(inofile);
```



```
717:C  5         end
718:C  5         else with boot_msus do
719:C  6           ioresult:=ord(inounit);
720:C  4       recover ioresult:=ord(escio(escapecode));
721:C  3     closefile:
722:C  3       try
723:C  4         mfclose;
724:C  4       recover ioresult:=ord(escio(escapecode));
725:C  3     otherwise
726:C  3       ioresult:=ord(ibadrequest);
727:C  3     end; {case request}
728:C  2   end; { bootdam }
729:S
730:D  1   procedure initbootdam;
731:D  2   type
732:D  2     lrec = packed record
733:D  2              pad  : packed array[0..15] of char;
734:D  2              mbptr : ^char;     { offset 16 }
735:D  2              mbsize: integer;   { offset 20 }
736:D  2            end;
737:D  2   var
738:D  2     boot_space  [-300]:^lrec;
739:C  2   begin
740:C  2     if boot_id<3 then gread:=false
741:C  3     else gread:=not boot_flags[7];
742:S
743:C  2     if gread then newbytes(boot_space^.mbptr,boot_space^.mbsize );
744:S
745:C  2   end; {initboot dam}
746:S
747:D  1   procedure boottm(f:fibp; request: amrequesttype;
748:D  2                   anyvar buffer: window; bufsize,position: integer);
749:C  2   begin
750:C  2     if gread then
751:C  3     begin
752:C  3       with f^ do
753:C  4       begin
754:C  4         ioresult:=ord(inoerror);
755:C  4         if request=readbytes then
756:C  5         begin
757:C  5           if ((position+bufsize)>fpeof) or
758:C  6              (position<0)   then ioresult:=ord(znoblock)
759:C  6           else
760:C  6           if (position mod 256)<>0 then  ioresult:=ord(zbadmode)
761:C  7           else
762:C  7           begin
763:C  7             try
764:C  8               boot_mread(position div 256,bufsize,buffer,false);
765:C  8             recover ioresult:=ord(escio(escapecode));
766:C  7           end;
767:C  5         end
768:C  5         else ioresult:=ord(zbadmode);
769:C  4       end;{ with }
770:C  3     end
771:C  3     else begin
772:C  3          driverkey := false;
773:C  3          miniio(f,request,buffer,bufsize,position);
774:C  3          driverkey := true;
775:C  3          end;
776:C  2   end;  { boottm }
```

```
   777:D   -53  1
   778:D        1   function srmnode(sc: shortint): string2;
   779:D        2   const scsize =  hex('10000'); nodecom = chr(128+6);
   780:D        2         command = hex('604003'); data    = hex('604005');
   781:D        2   type charptr = ^char;
   782:D    -4  2   var bnode: string2;
   783:D   -12  2       node, i: integer;
   784:C        2   begin
   785:C        2     try
   786:C        3     charptr(ord(sc*scsize+command))^ := nodecom;
   787:C        3     repeat until charptr(ord(sc*scsize+command))^ = chr(0);
   788:C        3     node := ord(charptr(ord(sc*scsize+data))^);
   789:C        3     if node < 10 then bnode := '0' else bnode := '';
   790:C        3     strwrite(bnode, strlen(bnode)+1, i, node:1);
   791:C        3     srmnode := bnode;
   792:C        3     recover if escapecode = -12 then srmnode := '' else escape(escapecode);
   793:C        2   end;
   794:S
   795:D        1   function bootnode: string2;
   796:D        2   const srmtype = 7*32+1;
   797:C        2   begin
   798:C        2   bootnode := '';
   799:C        2   if gread then with boot_msus do
   800:C        4     if mtype = srmtype then bootnode := srmnode(mscode);
   801:C        2   end;
   802:S
   803:C        1 end     { module }
   804:S
   805:S
   806:C        2 $include 'BOOTDAM'$;
   806:D        1 $include 'LOADER' $;
   807:D        1 module loader;
   808:S
   809:D        1 import sysglobals, asm;
   810:S
   811:D        1 export
   812:S
   813:D        1 const  blocksize = fblksize;
   814:D        1        vnlength = 7;
   815:D        1        fnlength = 15;
   816:D        1
   817:D        1 type
   818:S
   819:D        1   volname =  string[vnlength];
   820:D        1   filname =  string[fnlength];
   821:S
   822:D        1   dirrange = 0..mmaxint;         {0..MAXlongDIR; }
   823:S
   824:D        1   (* the following declaration serves for a "library directory" *)
   825:S
   826:D        1   direntry = record
   827:D        1              dfirstblk: shortint;          (*module starting block*)
   828:D        1              dlastblk: shortint;           (*block following end*)
   829:D        1          {NOTE:  for DIR[0], these refer to the library directory itself}
   830:D        1              case dfkind: filekind of
   831:D        1              untypedfile:                  (*library info in DIR[0]*)
   832:D        1                    (dvid: volname;         (*name of library*)
   833:D        1                     deovblk: shortint;     (*block following library*)
   834:D        1                     dnumfiles: dirrange;   (*num modules in library*)
   835:D        1                     dloadtime: shortint;   (*time of last modification*)
```



```
   836:D        1                     dlastboot: daterec );   (*most recent date setting*)
   837:D        1              datafile..lastfkind:
   838:D        1                    (dtid: filname;          (*title of module*)
   839:D        1                     dlastbyte: 1..fblksize; (*1..256 bytes in last block*)
   840:D        1                     daccess: daterec )      (*last modification date*)
   841:D        1            end (*direntry*) ;
   842:S
   843:D        1      ubyterec =          packed record ub: byte end;
   844:D        1      sbyterec =          packed record sb: -128..127 end;
   845:D        1      word =              0..65535;
   846:D        1      wordrec =           packed record w: word end;
   847:D        1      wordrecptr =        ^wordrec;
   848:D        1      wordlist =          packed array[0..maxint] of word;
   849:D        1      wordlistptr =       ^wordlist;
   850:S
   851:D        1      symboltable =       array[0..maxint] of char;
   852:D        1      symtableptr =       ^symboltable;
   853:S
   854:D        1      symbol =            string[255];
   855:D        1      symbolptr =         ^symbol;
   856:S
   857:D        1      datatype =          (sbyte, sword, sint, fltpt, ubyte, uword);
   858:D        1      reloctype =         (absolute, relocatable, global, general);
   859:S
   860:S
   861:D        1 referenceptr = packed record    {one or more present if type = general}
   862:D        1   case integer of
   863:D        1   0:(
   864:D        1   adr:  0..16383;              {word address of external symbol}
   865:D        1   op:   (addit, subit);        {add or subtract the modifying value}
   866:D        1   last: boolean);              {indicates end of list}
   867:S
   868:D        1   1:(w: word);                 {for comparisons}
   869:D        1 end;
   870:S
   871:D        1 gvrptr          = ^generalvalue;
   872:D        1 veptr           = ^valueextension;
   873:D        1 textdescptr     = ^textdescriptor;
   874:D        1 fileptr         = ^phyle;
   875:D        1 moddescptr      = ^moduledescriptor;
   876:D        1 refptrptr       = ^referenceptr;
   877:D        1 sortlistptr     = ^sortlist;
   878:D        1 moddirptr       = ^moduledirectory;
   879:D        1 filedirptr      = ^filedirectory;
   880:D        1 ptrtableptr     = ^ptrtable;
   881:D        1 patchdescptr    = ^patchdescriptor;
   882:S
   883:S
   884:D        1 addrec  =  record case integer of       {a universal address}
   885:D        1         -1:     (rp:    referenceptr);  {a two byte object; rest are 4 bytes:}
   886:S
   887:D        1          1:     (i:     integer);       {to change an address to integer}
   888:D        1          2:     (a:     integer);       {to do address arithmetic}
   889:D        1          3:     (p:    ^integer);       {to dereference}
   890:S
   891:D        1          4:     (sb:    ^sbyterec);     {^signed byte record}
   892:D        1          5:     (sw:    ^shortint);     {^signed word}
   893:D        1          6:     (si:    ^integer);      {^signed integer}
   894:D        1          7:     (fp:    ^real);         {^floating point}
   895:D        1          8:     (ub:    ^ubyterec);     {^unsigned byte record}
```

```
 896:D        1         9:      (uw:    wordrecptr);    {^unsigned word record}
 897:S
 898:D        1         10:     (gvp:   gvrptr);        {^generalvalue}
 899:D        1         11:     (vep:   veptr);         {^valueextension}
 900:D        1         12:     (tdp:   textdescptr);   {^textdescriptor}
 901:D        1         13:     (syp:   symbolptr);     {^symbol  (string)  }
 902:D        1         14:     (stp:   symtableptr);   {^symboltable}
 903:D        1         15:     (wlp:   wordlistptr);   {^wordlist}
 904:D        1         16:     (php:   fileptr);       {^phyle}
 905:D        1         17:     (mdp:   moddescptr);    {^moduledescriptor}
 906:D        1         18:     (rpp:   refptrptr);     {^referenceptr}
 907:D        1         19:     (sdp:   ^sortdesc);
 908:D        1         20:     (slp:   sortlistptr);   {^sortlist}
 909:D        1         21:     (drp:   moddirptr);     {^moduledirectory}
 910:D        1         22:     (fdp:   filedirptr);    {^filedirectory}
 911:D        1         23:     (bmp:   ^bitmap);
 912:D        1         24:     (fbp:   fibp);
 913:D        1         25:     (ptp:   ptrtableptr);   {^ptrtable}
 914:D        1         26:     (ilp:   ^indexlist);
 915:D        1         27:     (pdp:   patchdescptr);
 916:D        1         28:     (cp:    ^char);
 917:D        1         29:     (arp:   ^addrec);
 918:D        1         end;
 919:S
 920:D        1 generalvalue = packed record
 921:D        1   primarytype: reloctype;       {allows quick indication of most common types}
 922:D        1   datasize:    datatype;        {specifies 1, 2, 4 or 8 bytes, signed or not}
 923:D        1   patchable,                    {specifies self relative field in branch}
 924:D        1   valueextend: boolean;         {indicates the presence of valueextension}
 925:D        1   case longoffset:  boolean of  {1 or 3 byte offset }
 926:D        1        false:    (short:         byte);              {unsigned 8 bits}
 927:D        1        true:     (long:          0..16777215);       {unsigned 24 bit value}
 928:D        1 end;
 929:S
 930:D        1 valueextension = packed record {present if valueextend bit above is set}
 931:D        1   case datatype of
 932:D        1             sbyte,sword,sint,
 933:D        1             ubyte,uword:       {value:    integer);
 934:D        1             fltpt:             {valuer:   real);
 935:D        1 end;
 936:S
 937:D        1 phyle = file of char;
 938:S
 939:S
 940:D        1 moduledirectory = packed record
 941:D        1      date:      daterec;        {date of creation}
 942:D        1      revision:  daterec;        {producer's revision date number}
 943:D        1      producer:  char;           {A = assembler, C = compiler, L = linker, etc.}
 944:D        1      systemid: byte;            {system version number (hard or soft, etc.)
 945:D        1      notice: string[80];        {space for whatever comments may be desired}
 946:D        1      directorysize: integer;    {size of module directory, in bytes}
 947:D        1      modulesize: integer;       {total size of module, in bytes}
 948:S               executable:       boolean;       {module is executable,
 949:D        1                                         has start address}
 950:D        1      relocatablesize: integer;       {number of relocatable bytes requested}
 951:D        1      relocatablebase: integer;       {current origin of relocatable code}
 952:D        1      globalsize:       integer;      {number of global bytes requested}
 953:D        1      globalbase:       integer;      {A5 relative origin of global area}
 954:S
 955:D        1      extblock,                       {module relative block of EXT table}
```



```
 956:D        1      extsize,                        {size of EXT table, in bytes}
 957:D        1      defblock,                       {module relative block of DEF table}
 958:D        1      defsize,                        {size of DEF table, in bytes}
 959:D        1      sourceblock,                    {module relative block of DEFINE SOURCE}
 960:D        1      sourcesize,                     {size of source, in bytes}
 961:S
 962:D        1      textrecords:     integer;       {number of TEXT records}
 963:S
 964:S                       {Remainder of directory is made up of variable length
 965:S                        records.  Strings begin and end on word (even byte)
 966:S                        boundaries. The directory itself may cross block
 967:S                        boundaries.  General value or address records
 968:S                        (GVR's,  see description later)
 969:S                        occuring below have the short variant offset; the
 970:S                        offset itself is the length of the GVR to assist in
 971:D        1               stepping quickly through the list.}
 972:S
 973:S
 974:D        1       {mname: string[ (variable) ];      {name of module}
 975:S                {
 976:S                {startaddress:    gvr;             {execution address, present only
 977:D        1       {                                   if executable}
 978:S                {
 979:D        1       {repeat for each text record       {list of TEXT records}
 980:D        1       {   textstart,                     {module relative block of TEXT record}
 981:D        1       {   textsize,                      {size of TEXT record, in bytes}
 982:D        1       {   refstart,                      {module relative block of REF table}
 983:D        1       {   refsize:      integer;         {size of REF table, in bytes}
 984:D        1       {   loadaddress:  gvr;             {location to load the TEXT}
 985:D        1       {end                    }
 986:D        1 end;
 987:S
 988:D        1 textdescriptor = record
 989:D        1          textstart,                      {module relative block of TEXT record}
 990:D        1          textsize,                       {size of TEXT record, in bytes}
 991:D        1          refstart,                       {module relative block of REF table}
 992:D        1          refsize:      integer;          {size of REF table, in bytes}
 993:D        1        end;
 994:S
 995:D        1 bitmap = packed array[0..maxint] of boolean;
 996:S
 997:D        1 patchdescriptor = record
 998:D        1    patchlist:   patchdescptr;           {head of list of patch descriptors}
 999:D        1    patchref:    addrec;
1000:D        1 end;
1001:S
1002:D        1 moduledescriptor = record
1003:D        1   link:         moddescptr;            {descriptors will be chained together}
1004:S
1005:D        1   case  patchmod: boolean of
1006:D        1   true: (
1007:D        1   patchlink:    moddescptr;            {patchmods are additionally linked}
1008:D        1   patchlist:    patchdescptr;          {head of list of patch descriptors}
1009:D        1   patchbase:    integer;               {relocatable address of patch space}
1010:D        1   patchsize:    integer);              {total bytes of patch space}
1011:S
1012:D        1   false:(
1013:D        1   defaddr:      addrec;                {address of DEF table}
1014:D        1   defsize:      integer;               {size of DEF table, in bytes}
1015:S
```

```
1016:D        1   case resolved: boolean of              {only present during loading}
1017:D        1   true: (startaddr: integer;
1018:D        1           progname: tid;
1019:D        1           ucase,lastmodule: boolean);
1020:D        1   false:(extaddr:       addrec;          {address of EXT table}
1021:D        1           listaddr:     wordlistptr;     {index pointers into EXT table}
1022:D        1           listsize:     shortint;        {number of entries in list}
1023:D        1           unresbits:    addrec;          {flag to indicate unresolved symbols}
1024:D        1           relocbase,                     {relocatable address of module}
1025:D        1           globase:      integer;         {global base address}
1026:D        1           relocdelta,
1027:D        1           globaldelta:  integer;         {deltas for relocation}
1028:D        1           filefib:      addrec;          {file info block for module}
1029:D        1           fileblock:    shortint;        {file relative block of module}
1030:D        1           directory:    addrec));        {memory for processing modules}
1031:D        1 end;
1032:S
1033:D        1 filedirectory = array[0..maxint] of direntry;
1034:S
1035:D        1 sortdesc = record
1036:D        1                 modp:   moddescptr;
1037:D        1                 case integer of
1038:D        1           1:    {ext:   symbolptr; n:   shortint);
1039:D        1           2:    {def:   addrec);
1040:D        1                 end;
1041:D        1 sortlist = array[0..maxint] of sortdesc;
1042:S
1043:D        1 indexlist = array[1..maxint] of shortint;
1044:S
1045:D        1 ptrtable =          array[0..maxint] of addrec;
1046:S
1047:D        1 var     {memory management: }
1048:D        1         a5['g_dollar']:  integer;
1049:D        1         lowheap,                         {next available memory}
1050:D    -8  1         highheap:       addrec;          {last available memory}
1051:S
1052:D    -8  1         {input file information: }
1053:D   -12  1         fdirectory:     filedirptr;      {old library directory pointer}
1054:D   -16  1         loadfib:        addrec;          {pointer to FIB for open file}
1055:D   -20  1         wrongbyte:      integer;         {error information}
1056:D   -24  1         linkmodname:    addrec;          {ptr to name of module being linked}
1057:S
1058:D   -24  1         {linker information:  }
1059:D   -28  1         newmods:        moddescptr;      {modules currently being processed}
1060:D   -29  1         allresolved:    boolean;         {flag indicating whether any ext's}
1061:D   -29  1         totalreloc,
1062:D   -38  1         totalglobal:    integer;         {total bytes of areas}
1063:D   -38  1         startreloc,
1064:D   -46  1         startglobal:    integer;         {starting addresses for areas}
1065:S
1066:D   -50  1         eheap:  anyptr;                  (*HEAP MARK FOR MEM MANAGING*)
1067:D   -54  1         edefs:  moddescptr;
1068:D   -58  1         userstack: integer;
1069:D   -62  1         entrypoint:  moddescptr;
1070:D   -66  1         eglobal:              integer;   (*PERMANENT BASE OF DATA AREA*)
1071:D   -70  1         sysdefs:        moddescptr;      {list of permanent module descriptors}
1072:S
1073:D   -74  1     endmod:     moddescptr;              {marks end of list of new modules}
1074:D   -75  1     libfound: boolean;
1075:D   -80  1     wrongrec:   integer;                 {error information}
```



```
1076:D   -80  1     startdefs,
1077:D   -88  1     totaldefs: integer;                  {DEF table information}
1078:S
1079:D   -88  1 procedure markuser;
1080:D   -88  1 procedure releaseuser;
1081:S
1082:D   -88  1 procedure loadinfo(modnum: shortint; all,resolveexts: boolean);
1083:D   -88  1 procedure openlinkf(extra: addrec);      {formerly 'openlinkfile'}
1084:D   -88  1 procedure getbytes(var p: integer; size: integer);
1085:D   -88  1 procedure checkrev;
1086:D   -88  1 procedure matchfile;
1087:D   -88  1 procedure countcode;
1088:D   -88  1 procedure loadtext(onheap: boolean);
1089:D   -88  1 procedure zeromem(start: anyptr; size: integer);
1090:D   -88  1 procedure closefiles;
1091:D   -88  1 procedure movedefs(newstartdefs: integer);
1092:S
1093:D   -88  1 procedure loadq(filetogo: fid);          {name of file to be loaded}
1094:D   -88  1 procedure initloader;
1095:S
1096:D   -88  1 implement
1097:S
1098:D   -88  1 type pointer = ^integer;
1099:D   -88  1      address = integer;
1100:S
1101:D   -88  1 var sysdeftable['sysdeftable']:
1102:D   -88  1                    moduledescriptor;  {initial ROM symbol table}
1103:S
1104:D        1 procedure evalgvr(var gvalue, gvptr: integer;
1105:D   -88  2                   modptr:  moddescptr); external;
1106:S
1107:D        1 procedure relocate(reftop, refindex: address;
1108:D        2                    var object:  address;
1109:D   -88  2                    modptr:  moddescptr); external;
1110:S
1111:D        1 procedure getbytes(var p: integer; size: integer);
1112:C        2 begin
1113:C        2   if odd(size) then size := size + 1;
1114:C        2   p := lowheap.a;       lowheap.a := lowheap.a + size;
1115:C        2   if lowheap.a > highheap.a then escape(112);
1116:C        2 end;
1117:S
1118:D        1 procedure readblocks(anyvar f: fib; anyvar obj: integer; size, block: integer);
1119:C        2 begin
1120:C        2   call (f.am, addr(f), readbytes, obj, size, block*fblksize);
1121:C        2   if ioresult <> ord(inoerror) then escape(-10);
1122:C        2 end;
1123:S
1124:D        1 procedure markuser;
1125:C        2 begin
1126:C        2   entrypoint := nil;    mark(eheap);
1127:C        2   eglobal := userstack; edefs := sysdefs;
1128:C        2 end;
1129:S
1130:D        1 procedure releaseuser;
1131:C        2 begin
1132:C        2   entrypoint := nil;    release(eheap);
1133:C        2   userstack := eglobal; sysdefs := edefs;
1134:C        2 end;
1135:S
```

```
1136:D       1 procedure openlinkf(extra: addrec);
1137:D    -4 2 var directsize: integer;
1138:D    -8 2     slop:       integer;
1139:C       2 begin if ioresult = 0 then
1140:C       3     begin
1141:C       3     if loadfib.fbp^.fkind <> codefile then escape(116);
1142:C       3     highheap.a := highheap.a - blocksize;
1143:C       3     if highheap.a < lowheap.a then escape(112);
1144:C       3     fdirectory := highheap.fdp;
1145:C       3     readblocks(loadfib.php^,fdirectory^,blocksize,0);
1146:C       3     directsize := (fdirectory^[0].dnumfiles + 1)*sizeof(direntry);
1147:C       3     slop := (-directsize) mod blocksize;
1148:C       3     highheap.a := highheap.a + blocksize - directsize - slop;
1149:C       3     if directsize > blocksize then
1150:C       4       begin
1151:C       4       if highheap.a < lowheap.a then escape(112);
1152:C       4       moveleft(fdirectory^, highheap.fdp^, blocksize);
1153:C       4       extra.a := highheap.a + blocksize;
1154:C       4       readblocks(loadfib.php^,extra.p^,directsize - blocksize,1);
1155:C       4       end;
1156:C       3     if slop > 0 then
1157:C       4        begin
1158:C       4        extra := highheap; highheap.a := highheap.a + slop;
1159:C       4        moveright(extra.fdp^, highheap.fdp^, directsize);
1160:C       4        end;
1161:C       3     fdirectory := highheap.fdp;
1162:C       3     end
1163:C       3   else begin fdirectory := nil;          {directory invalid}
1164:C       3              loadfib  := extra.arp^;    {restore chain of FIB's}
1165:C       3              lowheap  := extra;         {zap unopend FIB}
1166:C       3        end;
1167:C       2 end; {openlinkfile}
1168:S
1169:D       1 procedure matchdefext(
1170:D       2         var resolved, matched:          boolean;
1171:D       2             matchflag:                  byte;
1172:D       2             deftable, exttable:         symtableptr;
1173:D       2             extlist:                    wordlistptr;
1174:D       2             listlength:                 shortint;
1175:D       2             deftablelength:             integer);
1176:D   -88 2 external;
1177:S
1178:D       1 procedure match1(modptr: moddescptr);
1179:D    -5 2 var defmodptr: moddescptr; matched: boolean;
1180:C       2 begin with modptr^ do
1181:C       3   begin
1182:C       3   defmodptr := link;
1183:C       3   while not resolved and (defmodptr <> nil ) do
1184:C       4     begin
1185:C       4     if not defmodptr^.patchmod then
1186:C       5       matchdefext(resolved, matched, 0,
1187:C       5                   defmodptr^.defaddr.stp, extaddr.stp,
1188:C       5                   listaddr, listsize, defmodptr^.defsize);
1189:C       4     defmodptr := defmodptr^.link;
1190:C       4     end;
1191:C       3   allresolved := allresolved and resolved;
1192:C       3   end;
1193:C       2 end;
1194:S
1195:D       1 procedure matchfile;
```



```
1196:D    -4 2 var modptr: moddescptr;
1197:C       2 begin
1198:C       2  modptr := newmods;
1199:C       2  while modptr <> nil do
1200:C       3    begin
1201:C       3    match1(modptr);
1202:C       3    modptr := modptr^.link;
1203:C       3    end;
1204:C       2 end;
1205:S
1206:D       1 procedure loadinfo(modnum: shortint; all,resolveexts: boolean);
1207:D    -4 2 var modptr:              moddescptr;
1208:D    -5 2     matched:             boolean;
1209:D   -10 2     leftover:            address;
1210:D   -12 2     modblock:            shortint;
1211:S
1212:D       2 procedure alphalist(
1213:D       3             symtable:           symtableptr;
1214:D       3             list:               wordlistptr;
1215:D       3             listlength:         shortint);
***WARNING: (line 1216): External declarations will be treated as global
1216:D   -12 3 external;
1217:S
1218:D       2 procedure makelist(tableptr:            symtableptr;    {DEF or EXT table}
1219:D       3                    length:             integer;        {length of symbol table}
1220:D       3                    bound:              shortint;       {boundary condition}
1221:D       3                    var listptr:        wordlistptr;    {address of list}
1222:D       3                    var listlength:     integer);       {number of symbols}
1223:S
1224:D    -8 3 var i, l:       integer;
1225:D   -12 3     n, len:     shortint;
1226:D   -16 3     ptr:        addrec;
1227:S
1228:C       3 begin
1229:C       3   n := 0;       listptr := lowheap.wlp;
1230:C       3   l := length;  i := 0;
1231:S
1232:C       3   while l > 0 do
1233:C       4     begin
1234:C       4     n := n + 1;
1235:C       4     getbytes(ptr.a, sizeof(wordrec));   ptr.uw^.w := i;
1236:C       4     len := ord(tableptr^[i]);
1237:C       4     len := len + bound - (len mod bound);
1238:C       4     if bound = 2 {DEF table} then len := len + ord(tableptr^[i+len+1]);
1239:C       4     i := i + len;
1240:C       4     l := l - len;
1241:C       4     end;
1242:C       3   lastlength := n;
1243:C       3   alphalist(tableptr, listptr, n);
1244:C       3 end;    { makelist }
1245:S
1246:D       2 procedure getdeftable;
1247:S
1248:D    -4 3   var defptr:   addrec;
1249:D    -4 3       symptr1,
1250:D   -12 3       symptr2:  addrec;
1251:D   -20 3       i,len:    integer;
1252:D   -24 3       list:     wordlistptr;
1253:D   -28 3       size:     integer;
1254:S
```

```
1255:C        3   begin
1256:C        3     with modptr^ do
1257:C        4       begin
1258:C        4       defsize := directory.drp^.defsize;
1259:C        4       getbytes(defaddr.a, defsize);
1260:C        4       if defsize > 0 then
1261:C        5        begin
1262:C        5        getbytes(defptr.a,  defsize);
1263:C        5        readblocks(loadfib.php^,  {unitread(funit, ... iocheck;}
1264:C        5                         defptr.p^, defsize, modblock + directory.drp^.defblock);
1265:C        5        makelist(defptr.stp, defsize, 2, list, size);
1266:C        5        symptr2 := defaddr;
1267:C        5        for i := 0 to size-1 do
1268:C        6          begin
1269:C        6          symptr1.a := defptr.a+list^[i];
1270:C        6          len := strlen(symptr1.syp^);
1271:C        6          len := len + 2 - ord(odd(len));
1272:C        6          len := len + gvrptr(symptr1.a+len)^.short;
1273:C        6          fastmove(symptr1.p, symptr2.p, len);
1274:C        6          symptr2.a := symptr2.a+len;
1275:C        6          end;
1276:C        5        lowheap := defptr;
1277:C        5        end;
1278:C        4       end;
1279:C        3   end;
1280:S
1281:D        2   procedure getexttable;
1282:D    -4  3   var  size:    integer;
1283:C        3   begin with modptr^, directory.drp^ do
1284:C        4     begin
1285:C        4     if extsize < 8 then extsize := 8;
1286:C        4     getbytes(extaddr.a, extsize);
1287:C        4     if extsize > 8 then
1288:C        5      readblocks(loadfib.php^, extaddr.p^, extsize, modblock + extblock);
1289:C        4     extaddr.stp^[0] := chr(0);
1290:C        4     extaddr.stp^[4] := chr(0);
1291:C        4     makelist(extaddr.stp, extsize, 4, listaddr, size);
1292:C        4     listsize := size;
1293:C        4     listaddr^[0] := 0;
1294:C        4     listaddr^[1] := 0;
1295:S
1296:C        4     end;
1297:C        3   end;
1298:S
1299:S
1300:D        2   procedure match2;
1301:D    -4  3   var extmodptr: moddescptr;
1302:C        3   begin
1303:C        3     extmodptr := newmods;
1304:C        3     while extmodptr <> nil  do with extmodptr^ do
1305:C        5       begin
1306:C        5       if not patchmod then if not resolved then
1307:C        7         begin
1308:C        7         matchdefext(resolved, matched, 0,
1309:C        7                         modptr^.defaddr.stp, extaddr.stp,
1310:C        7                         listaddr, listsize, modptr^.defsize);
1311:C        7         allresolved := allresolved and resolved;
1312:C        7         end;
1313:C        5       extmodptr := link;
1314:C        5       end;
```



```
1315:C        3   end;
1316:S
1317:C        2 begin {loadinfo}
1318:C        2   modblock := (fblock +) fdirectory^[modnum].dfirstblk;
1319:C        2   modptr := lowheap.mdp;
1320:C        2   getbytes(leftover, sizeof(moduledescriptor,false,false));
1321:C        2   modptr^.directory := lowheap;
1322:C        2   getbytes(leftover, blocksize);
1323:C        2   with modptr^, directory.drp^ do
1324:C        3     begin
1325:C        3     readblocks(loadfib.php^,  directory.drp^, blocksize, modblock);
1326:C        3     if directorysize > blocksize then
1327:C        4       begin
1328:C        4       getbytes(leftover,directorysize-blocksize);
1329:C        4       readblocks(loadfib.php^,   {unitread(funit ... iocheck)
1330:C        4                         pointer(leftover)^,directorysize-blocksize,modblock+1);
1331:C        4       end
1332:C        4     else lowheap.a := directory.a+directorysize;
1333:C        3     getdeftable;
1334:C        3     allresolved := true;          matched := false;
1335:C        3     match2;
1336:C        3     if matched or all then
1337:C        4          begin
1338:C        4          libfound := true;
1339:C        4          link := newmods;         newmods := modptr;
1340:C        4          patchmod := false;
1341:C        4          resolved := false;                (set appropriate variant)
1342:C        4          filefib  := loadfib;      {remember mass memory location for later}
1343:C        4          fileblock:= modblock;
1344:S
1345:C        4          getexttable;
1346:C        4          matchdefext(resolved, matched,
1347:C        4             0{patchable bit  *** fix later! **},
1348:C        4             defaddr.stp, extaddr.stp,listaddr,
1349:C        4             listsize, defsize);
1350:C        4          if resolveexts then match1(modptr);
1351:C        4          end
1352:C        4     else lowheap.mdp := modptr;
1353:C        3     end;
1354:S
1355:C        2 end; {loadinfo}
1356:S
1357:D        1 procedure reversepointers;
1358:D   -12  2 var modptr, lastptr, nextptr: moddescptr;
1359:C        2 begin
1360:C        2   modptr := newmods; lastptr := endmod;
1361:C        2   while modptr <> endmod do with modptr^ do
1362:C        4     begin nextptr := link;      link := lastptr;
1363:C        4           lastptr := modptr;    modptr := nextptr;
1364:C        4     end;
1365:C        2   newmods := lastptr;
1366:C        2 end;
1367:S
1368:D        1 procedure countcode;
1369:D    -4  2 var modptr: moddescptr;
1370:C        2 begin
1371:C        2   reversepointers;
1372:C        2   modptr := newmods;
1373:C        2   totalreloc := 0; totalglobal := 0;
1374:C        2   while modptr <> endmod do with modptr^ do
```

```
1375:C        4     begin
1376:C        4     with directory.drp^ do
1377:C        5       begin
1378:C        5       totalreloc  := totalreloc  + relocatablesize + ord(odd(relocatablesize));
1379:C        5       totalglobal := totalglobal + globalsize      + ord(odd(globalsize));
1380:C        5       end;
1381:C        4     modptr := link;
1382:C        4     end;
1383:C        2 end;
1384:S
1385:D        1 procedure closefiles;
1386:C        2 begin
1387:C        2 while loadfib.php <> nil do
1388:C        3   begin
1389:C        3   {close(loadfib.php^);}
1390:C        3   with loadfib.fbp^, unitable^[funit] do
1391:C        4                                   call(dam, loadfib.fbp^, funit, closefile);
1392:C        3   loadfib.a := loadfib.a - sizeof(addrec);
1393:C        3   loadfib   := loadfib.arp^;
1394:C        3   end;
1395:C        2 end;
1396:S
1397:D        1 procedure resolve;
1398:D   -4   2 var modptr: moddescptr;
1399:D  -12   2     mrbase,mgbase:    integer;
1400:D  -14   2     len:          shortint;
1401:D  -16   2     i:            shortint;
1402:S
1403:C        2 begin
1404:C        2   modptr := newmods;
1405:C        2   mrbase := startreloc; mgbase := startglobal;
1406:C        2   while modptr <> endmod do with modptr^ do
1407:C        4     begin
1408:C        4     with directory.drp^ do
1409:C        5       begin
1410:C        5       relocbase := mrbase;    relocdelta :=  mrbase - relocatablebase;
1411:C        5       mrbase := mrbase + relocatablesize + ord(odd(relocatablesize));
1412:C        5       globase := mgbase;      globaldelta := mgbase - globalbase;
1413:C        5       mgbase := mgbase - globalsize       - ord(odd(globalsize));
1414:C        5       end;
1415:C        4     modptr := link;
1416:C        4     end;
1417:C        2 end; {resolve}
1418:S
1419:D        1   procedure createdefs;
1420:D   -4   2   var modptr:             moddescptr;
1421:D   -4   2       newmodptr,
1422:D  -24   2       sp,ptr1,ptr2,enddefs:       addrec;
1423:D  -26   2       len:                shortint;
1424:C        2   begin
1425:C        2   reversepointers;
1426:C        2   entrypoint := nil;
1427:C        2   modptr := newmods;
1428:C        2   startdefs := lowheap.a;
1429:C        2   while modptr <> endmod do with modptr^ do
1430:C        4     begin
1431:C        4     ptr1 := defaddr;
1432:C        4     enddefs.a := ptr1.a + defsize;
1433:C        4     getbytes(newmodptr.a, sizeof(moduledescriptor,false, true));
1434:C        4     with newmodptr.mdp^ do
```



```
1435:C        5     begin
1436:C        5     patchmod := false;  resolved := true;
1437:C        5     progname := modptr^.filefib.fbp^.ftid   (togo);
1438:C        5     ucase := unitable^[modptr^.filefib.fbp^.funit].uuppercase;
1439:C        5     if modptr^.directory.drp^.executable then
1440:C        6           begin
1441:C        6           sp.a := modptr^.directory.a+sizeof(moduledirectory);
1442:C        6           sp.a := sp.a+strlen(sp.syp^)+2-ord(odd(strlen(sp.syp^)));
1443:C        6           evalgvr(startaddr,sp.i,modptr);
1444:C        6           lastmodule := entrypoint=nil;
1445:C        6           entrypoint:= newmodptr.mdp;   {moddescptr(newmodptr.a - loaddelta);}
1446:C        6           end
1447:C        6     else begin startaddr := 0;
1448:C        6                lastmodule := false;
1449:C        6           end;
1450:C        5     link := sysdefs;
1451:C        5     sysdefs := newmodptr.mdp;   {moddescptr(newmodptr.a - loaddelta);}
1452:C        5     defaddr.a := lowheap.a      {- loaddelta};
1453:C        5     while ptr1.a < enddefs.a do
1454:C        6       begin
1455:C        6       len := strlen(ptr1.syp^) + 2 - ord(odd(strlen(ptr1.syp^)));
1456:C        6       getbytes(ptr2.a,len); fastmove(ptr1.p, ptr2.p, len);
1457:C        6       ptr1.a := ptr1.a + len;
1458:C        6       getbytes(ptr2.a, sizeof(generalvalue, false));
1459:C        6       with ptr2.gvp^ do
1460:C        7         begin primarytype := absolute;    datasize := sint;
1461:C        7               patchable := false;         valueextend := true;
1462:C        7               longoffset := false;        short := 6;
1463:C        7         end;
1464:C        6       getbytes(ptr2.a, sizeof(valueextension,sint));
1465:C        6       evalgvr(ptr2.vep^.value, ptr1.i, modptr);
1466:C        6       end;
1467:C        5     defsize := lowheap.a -(defaddr.a  (+ loaddelta));
1468:C        5     end;
1469:C        4   modptr := link;
1470:C        4   end;
1471:C        2   totaldefs := lowheap.a - startdefs;
1472:C        2   end;
1473:S
1474:D        1   procedure movedefs(newstartdefs: integer);
1475:D   -4   2   var modptr:             moddescptr;
1476:D   -8   2       defdelta:           integer;
1477:D  -12   2       previous:           ^addrec;
1478:C        2   begin
1479:C        2   defdelta := newstartdefs - startdefs;
1480:C        2   previous := addr(sysdefs);
1481:C        2   modptr   :=      sysdefs;
1482:C        2   while modptr <> endmod do with modptr^ do
1483:C        4     begin
1484:C        4     previous^.a := previous^.a + defdelta;
1485:C        4     if entrypoint = modptr then entrypoint := previous^.mdp;
1486:C        4     defaddr.a := defaddr.a + defdelta;
1487:C        4     previous := addr(link);
1488:C        4     modptr   :=      link;
1489:C        4     end;
1490:C        2   fastmove(pointer(startdefs), pointer(newstartdefs), totaldefs);
1491:C        2   startdefs := newstartdefs;
1492:C        2   end;
1493:D  -88   1
1494:D        1 procedure loadtext(onheap: boolean);
```

```
1495:S
1496:D        2 const maxrefsize = 254;
1497:S
1498:D    -4  2 var modptr:     moddescptr;     {current module being loaded}
1499:S
1500:D    -4  2     textbuffer,          {base of text record buffer}
1501:D    -4  2     object,object0,      {object in text record being modified by ref record}
1502:D    -4  2     refbuffer,           {base of ref table buffer}
1503:D    -4  2     oldindex:            {pointer to old text descriptors}
1504:D   -24  2                 addrec;
1505:S
1506:D   -24  2     loaddelta,           {loader displacement}
1507:D   -24  2     oldtextrec:          {text records left to process from old module}
1508:D   -32  2                 integer;
1509:S
1510:S
1511:S
1512:C        2 begin {procedure loadtext}
1513:C        2   if onheap then
1514:C        3     begin
1515:C        3     getbytes(textbuffer.a, totalreloc (+ blocksize));
1516:C        3     loaddelta := textbuffer.a - startreloc;
1517:C        3     end
1518:C        3   else loaddelta := 0;
1519:C        2   resolve;
1520:C        2   modptr := newmods;
1521:C        2   while modptr <> endmod do with modptr^, directory.drp^ do
1522:C        4     begin
1523:C        4     linkmodname.a := directory.a + sizeof(moduledirectory);
1524:C        4     oldindex.a := linkmodname.a + strlen(linkmodname.syp^) + 2 -
1525:C        4             ord(odd(strlen(linkmodname.syp^)));
1526:C        4     if executable then oldindex.a := oldindex.a + oldindex.gvp^.short;
1527:C        4     for oldtextrec := 1 to textrecords do
1528:C        5      with oldindex.tdp^ do
1529:C        6       begin
1530:C        6       oldindex.a := oldindex.a + sizeof(textdescriptor);
1531:C        6       evalgvr(object.i, oldindex.i, modptr);
1532:C        6       if textsize > 0 then
1533:C        7        begin
1534:C        7        if object.a >= startreloc
1535:C        8         then if object.a < startreloc + totalreloc
1536:C        9          then object.a := object.a  + loaddelta;
1537:S
1538:C        7        readblocks(modptr^.filefib.php^,  {unitread(modptr^.fileunit,...iocheck}
1539:C        7                                    object.p^,textsize, fileblock + textstart);
1540:C        7        if refsize > 0 then
1541:C        8         begin
1542:C        8         getbytes(refbuffer.a, refsize);
1543:C        8         readblocks(modptr^.filefib.php^,
1544:C        8                                    refbuffer.p^,refsize, fileblock + refstart);
1545:C        8         object0 := object;
1546:C        8         try relocate(lowheap.a, refbuffer.a, object.a, modptr);
1547:C        9         recover
1548:C        9           begin
1549:C        9           wrongbyte := object.a-object0.a;
1550:C        9           wrongrec := oldtextrec;
1551:C        9           escape(escapecode);
1552:C        9           end;
1553:C        8         lowheap := refbuffer;
1554:C        8         end;
```



```
1555:C        7       end;
1556:C        6      end; {for oldtextrec}
1557:C        4     modptr := link;
1558:C        4     end;
1559:S
1560:C        2   createdefs;
1561:C        2   closefiles;
1562:S
1563:C        2   if onheap then fastmove(textbuffer.p, pointer(startreloc), totalreloc);
1564:S
1565:C        2 end;
1566:S
1567:D        1 procedure zeromem(start: anyptr; size: integer);
1568:D    -8  2 var ptr, endp: addrec;
1569:C        2 begin
1570:C        2   ptr.p := start; endp.a := ptr.a + size;
1571:C        2   while ptr.a < endp.a do
1572:C        3     begin
1573:C        3     ptr.sw^ := 0;
1574:C        3     ptr.a := ptr.a + 2;
1575:C        3     end;
1576:C        2 end;
1577:S
1578:D        1 procedure checkrev;
1579:C        2 begin
1580:C        2 if newmods^.directory.drp^.systemid <> 3 then escape(118);
1581:C        2 end;
1582:S
1583:D  -122  1 procedure loadq(filetogo:fid);
1584:S
1585:D  -122  2 type str9 = packed array[1..9] of char;
1586:S
1587:D  -122  2 const loading = str9['Loading '''];
1588:S
1589:D  -126  2 var modnum, esc:   shortint;
1590:D  -130  2     extra:      addrec;
1591:D  -138  2     i, ior: integer;
1592:S
1593:C        2 begin
1594:S
1595:C        2 loadfib.php := nil;
1596:C        2 try
1597:C        3   releaseuser;          {get rid of last program}
1598:C        3   mark(lowheap.p);              highheap.a := userstack;
1599:C        3   startreloc := lowheap.a;
1600:C        3   newmods       := sysdefs;     endmod          := sysdefs;
1601:S
1602:C        3   getbytes(extra.a,   sizeof(addrec));  {save old value of loadfib}
1603:C        3   extra.arp^ := loadfib;
1604:C        3   getbytes(loadfib.a, sizeof(fib,1));   {create a FIB for new file}
1605:C        3   with loadfib.fbp^, unitable^[sysunit] do
1606:C        4     begin
1607:C        4     fbuffered := false;
1608:C        4     ftitle := filetogo;
1609:C        4     funit := sysunit;
1610:C        4     call(dam, loadfib.fbp^, sysunit, openfile);
1611:C        4     am := tm;
1612:C        4     end;
1613:C        3   openlinkf(extra);
1614:S
```

```
  1615:C        3   i := strlen(filetogo)+1; setstrlen(filetogo, i+9);
  1616:C        3   filetogo[i] := ',,,';
  1617:C        3   moveright(filetogo[1],       filetogo[10], i);
***WARNING: (line 1618): 'ADDR' of a constant may not be supported on other implementations
  1618:C        3   fastmove (addr(loading),     addr(filetogo[1]),  9);
  1619:C        3   cpymsg(filetogo);
  1620:S
  1621:C        3   if fdirectory = nil then escape(-10);
  1622:C        3   for modnum := 1 to fdirectory^[0].dnumfiles do
  1623:C        4                          begin loadinfo(modnum, true, false); checkrev; end;
  1624:C        3   allresolved := true; matchfile;
  1625:C        3   highheap.a := userstack;
  1626:S
  1627:C        3   if not allresolved then escape(-119);
  1628:S
  1629:C        3   countcode;
  1630:S
  1631:C        3   highheap.a := userstack - totalglobal;
  1632:C        3   if highheap.a < (a5 - 32768) then escape(117);
  1633:C        3   startglobal   := userstack - a5;
  1634:C        3   userstack := highheap.a;
  1635:C        3   zeromem(highheap.p, totalglobal);
  1636:S
  1637:C        3   loadtext(true);
  1638:S
  1639:C        3   movedefs(startreloc+totalreloc);
  1640:S
  1641:C        3   lowheap.a := startdefs + totaldefs;
  1642:C        3   release(lowheap.p);
  1643:S
  1644:C        3 recover
  1645:C        3   begin
  1646:C        3   esc := escapecode; ior := ioresult;
  1647:C        3   closefiles;
  1648:C        3   releaseuser;
  1649:C        3   ioresult := ior; escape(esc);
  1650:C        3   end;
  1651:C        2 end;
  1652:S
  1653:D        1 procedure initloader;
  1654:C        2 begin
  1655:C        2   sysdefs := addr(sysdeftable);
  1656:C        2   {findroms;                  {find and "load" rom modules }
  1657:C        2   userstack := a5 + 32208;              {GBASE OF SYSTEM_P -- SEE *LINKMAP}
  1658:C        2   markuser;
  1659:C        2 end;
  1660:S
  1661:C        1 end
  1662:C        2 $include 'LOADER' $;
  1663:S
  1664:D        1 import bootdammodule, sysglobals, asm, loader;
  1665:S
  1666:D        1 const splen = 20;
  1667:D        1 type stringsp = string[splen];
  1668:D        1 const sysprefix = stringsp['/WORKSTATIONS/SYSTEM'];
  1669:D   -12  1 var sysstart: string[11];
  1670:S
  1671:D        1 procedure unitioinit;
  1672:C        2 begin
  1673:C        2   sysunit := 0;
```



```
  1674:C        2   initbootdam;
  1675:C        2   newwords(unitable, sizeof(unitentry) div 2);
  1676:C        2   with unitable^[0] do begin
  1677:C        3       dam := bootdam;
  1678:C        3       tm  := boottm;
  1679:C        3       sc := 0;
  1680:C        3       ba := 0;
  1681:C        3       du := 0;
  1682:C        3       dv := 0;
  1683:C        3       byteoffset := 0;
  1684:C        3       devid := 0;
  1685:C        3       uvid := '';
  1686:C        3       dvrtemp := 0;                   {must always initially be zero!}
  1687:C        3       dvrtemp2 := -1;                 {must always initially be -1!}
  1688:C        3       letter := chr(0);
  1689:C        3       offline := false;               {always initially assume online}
  1690:C        3       uisinteractive := false;
  1691:C        3       umediavalid := false;           {assume media invalid until verified}
  1692:C        3       uuppercase := false;            {assume case is significant}
  1693:C        3       uisfixed := false;              {assume unit not fixed}
  1694:C        3       ureportchange := false;         {do not report media changes}
  1695:C        3       pad := 0;                       {not used yet}
  1696:C        3       uisblkd :=  true;
  1697:C        3       umaxbytes := 528 * fblksize;
  1698:C        3     end {with}
  1699:S
  1700:C        3 end;
  1701:S
  1702:D        1 procedure initfkinds;
  1703:D    -2  2 var fk: filekind;
  1704:C        2 begin
  1705:C        2   mark(amtable);              {TO AVOID ADDRESSING ERRORS}
  1706:C        2   new(efttable);
  1707:C        2   for fk := untypedfile to lastfkind do efttable^[fk] := 0;
  1708:C        2   efttable^[codefile] := -5582;
  1709:C        2 end;
  1710:S
  1711:D        1 procedure go;
  1712:D    -1  2 var  done: boolean;
  1713:D    -6  2      modptr: moddescptr;
  1714:C        2 begin {GO}
  1715:C        2   if entrypoint <> nil then
  1716:C        3     try
  1717:C        4       modptr := entrypoint;
  1718:C        4       repeat
  1719:C        5         if modptr^.startaddr<>0 then
  1720:C        6           userprogram(modptr^.startaddr, userstack);
  1721:C        5         done := modptr^.lastmodule;
  1722:C        5         modptr := modptr^.link;
  1723:C        5       until done;
  1724:C        4     recover escape(escapecode);
  1725:C        2 end;  {GO}
  1726:S
  1727:D   -82  1 procedure add(var s: string; st: string80);
  1728:D   -86  2 var i: integer;
  1729:C        2 begin
  1730:C        2   for i := 1 to strlen(st) do
  1731:C        3     begin setstrlen(s, strlen(s)+1); s[strlen(s)] := st[i]; end;
  1732:C        2 end;
  1733:S
```

```
1734:D        1 procedure createsysname;
1735:D        2 type stringsysp = packed array[1..10] of char;
1736:D        2 const sysp      = stringsysp['SYSTEM_P'];
1737:D    -4  2 var i,j: shortint;
1738:D    -5  2     c: char;
1739:C        2 begin
1740:C        2  sysstart := 'INITLIB';
1741:C        2  i := 0;
1742:C        2  repeat i := i + 1; until (i=10) or (sysname[i]<>sysp[i]);
1743:C        2  if i < 10 then
1744:C        3    begin
1745:C        3    setstrlen(sysstart, strmax(sysstart));
1746:C        3    if i < 8 then i := 3 else i := 7;
1747:C        3    j := 4;
1748:C        3    repeat
1749:C        4      i := i + 1;
1750:C        4      c := sysname[i];
1751:C        4      if c <> ' ' then begin j := j + 1; sysstart[j] := c; end;
1752:C        4    until (i=10) or (c=' ');
1753:C        3    setstrlen(sysstart, j);
1754:C        3    end;
1755:C        2 end;
1756:S
1757:D        1 procedure trysrmboot;
1758:D    -4  2 var   s2: string2;
1759:D   -46  2       il: string[40];
1760:C        2 begin
1761:C        2  s2 := bootnode;
1762:C        2  if strlen(s2) > 0 then
1763:C        3    begin
1764:C        3    nodestr := sysprefix; add(nodestr, s2); add(nodestr, '/');
1765:C        3    il := nodestr; add(il, sysstart);
1766:C        3    try loadq(il);
1767:C        4    recover if escapecode <> -10 then escape(escapecode);
1768:C        3    if entrypoint = nil then
1769:C        4      begin
1770:C        4      setstrlen(nodestr, splen+1); nodestr[splen+1] := '/';
1771:C        4      il := nodestr; add(il, sysstart);
1772:C        4      try loadq(il);
1773:C        5      recover if escapecode <> -10 then escape(escapecode);
1774:C        4      end;
1775:C        3    end;
1776:C        2 end;
1777:S
1778:C        1 begin
1779:C        1    powerup;                          {SET UP STACK, GLOBAL AREA, ETC.}
1780:C        1    unitioinit;                       {SET UP UNIT TABLE ENTRY FOR BOOTING}
1781:C        1    initfkinds;                       {SET UP MINIMAL FILE TYPE TABLE}
1782:C        1    initloader;                       {SET UP LOADER }
1783:S
1784:C        1    try
1785:C        2      createsysname;
1786:C        2      trysrmboot;
1787:C        2      if entrypoint = nil then
1788:C        3         begin
1789:C        3         nodestr := '';
1790:C        3         loadq(sysstart);
1791:C        3         end;
1792:C        2      cpymsg('');                       {CLEAR SCREEN, PUT UP COPYRIGHT}
1793:C        2      go;                               {EXECUTE ALL PROGRAMS IN SYSSTART}
```



```
1794:C        2    recover errmsg;                   {PRINT ERROR MESSAGE }
1795:C        1    while true do;
1796:C        1 end.
1797:S

No errors. 2 warnings.
            ***** Nonstandard language features enabled *****
```

Description

IOLIB is the high-level, user-accessible I/O facilities.

Requirements

I/O library kernel (IODECLARATIONS, etc.)

```
 1:S                                                    (*
 2:S
 3:S          (c) Copyright Hewlett-Packard Company, 1984.
 4:S         All rights are reserved.  Copying or other
 5:S         reproduction of this program except for archival
 6:S         purposes is prohibited without the prior
 7:S         written consent of Hewlett-Packard Company.
 8:S
 9:S
10:S                     RESTRICTED RIGHTS LEGEND
11:S
12:S         Use, duplication, or disclosure by the Government
13:S         is subject to restrictions as set forth in
14:S         paragraph (b) (3) (B) of the Rights in Technical
15:S         Data and Computer Software clause in
16:S         DAR 7-104.9(a).
17:S
18:S         HEWLETT-PACKARD COMPANY
19:D       0 Fort Collins, Colorado                         *)
20:S
21:S
22:D       0 $MODCAL ON$
23:D       0 $PARTIAL_EVAL ON$
24:D       0 $STACKCHECK ON$
25:D       0 $RANGE OFF$
26:D       0 $DEBUG OFF$
27:D       0 $OVFLCHECK OFF$
```



```
28:D       0 $PAGE$
29:D       0 (*************************************************************************)
30:D       0 (*                                                                       *)
31:D       0 (*           RELEASED        VERSION         3.0                         *)
32:D       0 (*                                                                       *)
33:D       0 (*************************************************************************)
34:D       0 (*                                                                       *)
35:D       0 (*                                                                       *)
36:D       0 (*      IOLIB           IOLIB                                           *)
37:D       0 (*                                                                       *)
38:D       0 (*                                                                       *)
39:D       0 (*************************************************************************)
40:D       0 (*                                                                       *)
41:D       0 (*                                                                       *)
42:D       0 (*      library       -  IOLIB                                          *)
43:D       0 (*      name          -  IOLIB                                          *)
44:D       0 (*      module(s)     -  general_1                                      *)
45:D       0 (*                    -  hpib_1                                         *)
46:D       0 (*                    -  general_2                                      *)
47:D       0 (*                    -  general_3                                      *)
48:D       0 (*                    -  general_4                                      *)
49:D       0 (*                    -  hpib_0                                         *)
50:D       0 (*                    -  hpib_2                                         *)
51:D       0 (*                    -  hpib_3                                         *)
52:D       0 (*                    -  serial_0                                       *)
53:D       0 (*                    -  serial_3                                       *)
54:D       0 (*                                                                      *)
55:D       0 (*      author        -                                                 *)
56:D       0 (*      phone         -                                                 *)
57:D       0 (*                                                                      *)
58:D       0 (*      date          -  June  1 , 1981                                 *)
59:D       0 (*      update        -  June  4,  1984                                 *)
60:D       0 (*      release       -  June  4,  1984                                 *)
61:D       0 (*                                                                      *)
62:D       0 (*      source        -  IOLIB:IOLIB.TEXT                               *)
63:D       0 (*      object        -  IOLIB:IOLIB.CODE                               *)
64:D       0 (*                                                                      *)
65:D       0 (*************************************************************************)
```

```
    66:D        0 $PAGE$
    67:D        0 (*************************************************************************)
    68:D        0 (*                                                                       *)
    69:D        0 (*                                                                       *)
    70:D        0 (*      This is the source code for an external procedures library       *)
    71:D        0 (*      to be used for general purpose interfacing on the HP 9826.       *)
    72:D        0 (*                                                                       *)
    73:D        0 (*      The library consists of 3 primary sets of modules -              *)
    74:D        0 (*                                                                       *)
    75:D        0 (*              1.      KERNEL modules                                   *)
    76:D        0 (*              2.      driver modules                                   *)
    77:D        0 (*              3.      IOLIB  modules                                   *)
    78:D        0 (*                                                                       *)
    79:D        0 (*      The KERNEL modules consist of the following modules -            *)
    80:D        0 (*                                                                       *)
    81:D        0 (*              1.      iodeclarations  ( contains static r/w space )    *)
    82:D        0 (*              2.      iocomasm                                         *)
    83:D        0 (*              3.      general_0       ( initialization & low level     *)
    84:D        0 (*                                        routines like ioread/iowrite) *)
    85:D        0 (*      The KERNEL modules also have an executable program segement      *)
    86:D        0 (*      that gets executed at the time it is loaded.  This program       *)
    87:D        0 (*      initializes the static read/write memory.  This program also     *)
    88:D        0 (*      allocates the temporary storage for any card that exists -       *)
    89:D        0 (*      independent of whether there is or is not a driver for it.       *)
    90:D        0 (*                                                                       *)
    91:D        0 (*      The driver modules consist of the actual assembly or PASCAL      *)
    92:D        0 (*      routines that deal with a specific interface card.  There is     *)
    93:D        0 (*      also an executable program segment for each driver module.       *)
    94:D        0 (*      This program searches the select code table in the static r/w    *)
    95:D        0 (*      initialized by the KERNEL general_0 module for all select codes *)
    96:D        0 (*      that have the right interface card ( HPIB drivers will search    *)
    97:D        0 (*      for the 98624 interface ).  This program will then set up the    *)
    98:D        0 (*      driver tables to point to the correct drivers.                   *)
    99:D        0 (*                                                                       *)
   100:D        0 (*      The rest of the IOLIB modules are high-level modules that are    *)
   101:D        0 (*      used by an end user in his/her application program.              *)
   102:D        0 (*                                                                       *)
   103:D        0 (*      The KERNEL and some set of driver modules will exist in the      *)
   104:D        0 (*      SYSTEM.INITLIB file as object code ( not EXPORT text ).  The     *)
   105:D        0 (*      export text will reside on the SYSTEM.LIBRARY file.  The rest    *)
   106:D        0 (*      of the library will reside on the SYSTEM.LIBRARY.                *)
   107:D        0 (*                                                                       *)
   108:D        0 (*************************************************************************)
```



```
   109:D        0 $PAGE$
   110:D        0 (*************************************************************************)
   111:D        0 (*                                                                       *)
   112:D        0 (*                                                                       *)
   113:D        0 (*      BUG FIX HISTORY         - after release                          *)
   114:D        0 (*                                                                       *)
   115:D        0 (*                                                                       *)
   116:D        0 (*      BUG #   BY  / ON        LOCATION        DESCRIPTION              *)
   117:D        0 (*      -----   -----------     --------------  -----------------------  *)
   118:D        0 (*                                                                       *)
   119:D        0 (*      1250    -----------     HPIB_3          request service allows   *)
   120:D        0 (*              01/08/1982      request_service active ctl to request    *)
   121:D        0 (*                                              service.                 *)
   122:D        0 (*                                                                       *)
   123:D        0 (*      1251    -----------     HPIB_2          local(7) with sc 7 as    *)
   124:D        0 (*              01/08/1982      local           sys ctl / not act ctl    *)
   125:D        0 (*              01/26/1982                      gives error.             *)
   126:D        0 (*                                                                       *)
   127:D        0 (*      1252    -----------     HPIB_2          remote(7) with sc 7 as   *)
   128:D        0 (*              01/08/1982      remote          act ctl / not sys ctl    *)
   129:D        0 (*                                              doesn't give error.      *)
   130:D        0 (*                                                                       *)
   131:D        0 (*      1258    -----------     HPIB_2          pass control sends the   *)
   132:D        0 (*              01/08/1982      pass_control    wrong sequence to pass   *)
   133:D        0 (*                                              control to itself.       *)
   134:D        0 (*                                                                       *)
   135:D        0 (*      1269    -----------     SERIAL_3        bad check for a 98626    *)
   136:D        0 (*              01/08/1982      set_stop_bits   card.                    *)
   137:D        0 (*                                                                       *)
   138:D        0 (*      1270    -----------     SERIAL_3        make procedures for      *)
   139:D        0 (*              01/08/1982      set_baud_rate   data comm consistent     *)
   140:D        0 (*                              set_stop_bits   for buffered control.    *)
   141:D        0 (*                              set_parity                               *)
   142:D        0 (*                              set_char_length                          *)
   143:D        0 (*                                                                       *)
   144:D        0 (*      1281    -----------     GENERAL_3       Wrong message for error  *)
   145:D        0 (*              01/08/1982      ioerror_message ioe_not_dvc.             *)
   146:D        0 (*                                                                       *)
   147:D        0 (*      0082    -----------     GENERAL_3       Addition of a link for   *)
   148:D        0 (*              07/23/1982      ioerror_message the error messages.      *)
   149:D        0 (*                                              See also IODECLARATIONS. *)
   150:D        0 (*                                                                       *)
   151:D        0 (*      0083    -----------     GENERAL_4       Addition of buffer_busy  *)
   152:D        0 (*              07/23/1982      buffer_busy     and isc_busy routines.   *)
   153:D        0 (*                              isc_busy                                 *)
   154:D        0 (*                                                                       *)
   155:D        0 (*      0355    -----------     SERIAL_3        Set parity of one and    *)
   156:D        0 (*              08/20/1982      set_parity      zero parity is backwards *)
   157:D        0 (*                                              for the 98628 card.      *)
   158:D        0 (*                                                                       *)
   159:D        0 (*      0359    -----------     SERIAL_3        Changes for addition     *)
   160:D        0 (*              08/26/1982      set_parity      of 98626 drivers.        *)
   161:D        0 (*                              set_char_length                          *)
   162:D        0 (*                              set_stop_bits                            *)
   163:D        0 (*                                                                       *)
   164:D        0 (*      0364    -----------     GENERAL_3       Addition of SRM driver   *)
   165:D        0 (*              08/23/1982      ioerror_message error codes.  See also   *)
   166:D        0 (*                                              IODECLARATIONS.          *)
   167:D        0 (*                                                                       *)
   168:D        0 (*       557    -----------     GENERAL_3       Mistyped. ( typo )       *)
```

```
   169:D        0 (*              10/01/1982      set_parity                              *)
   170:D        0 (*                                                                      *)
   171:D        0 (*      jsjs    -----------     HPIB_2          BUG FIX error in Local  *)
   172:D        0 (*              03/09/1983      local           procedure for isc param *)
   173:D        0 (*                                              and not sys controller. *)
   174:D        0 (*                                                                      *)
   175:D        0 (*      tttt    ---------       HPIB_1          Use timer on CPU board  *)
   176:D        0 (*              08/03/1983                      if available for timeout*)
   177:D        0 (*                                              checking                *)
   178:D        0 (*                                                                      *)
   179:D        0 (*              ---------       serial modules  add code for 98644      *)
   180:D        0 (*              5/15/84                                                 *)
   181:D        0 (*              6/4/84                                                  *)
   182:D        0 (***************************************************************************)
```



```
   183:D        0 $PAGE$
   184:D        0 (***************************************************************************)
   185:D        0 (*                                                                         *)
   186:D        0 (*                                                                         *)
   187:D        0 (*      REFERENCES :                                                       *)
   188:D        0 (*                                                                         *)
   189:D        0 (*                                                                         *)
   190:D        0 (*      1.  9826 I/O Designers Guide            ( ----------- )          *)
   191:D        0 (*                                                                         *)
   192:D        0 (*      2.  68000 Manual                        ( Motorola )             *)
   193:D        0 (*                                                                         *)
   194:D        0 (*      3.  Pascal alpha site ERS               ( ---------- )           *)
   195:D        0 (*                                                                         *)
   196:D        0 (*      4.  Pascal I/O Library ERS              ( ------------ )         *)
   197:D        0 (*                                                                         *)
   198:D        0 (*      5.  9826 HPL EIO & IOD listings         ( ------------ )         *)
   199:D        0 (*                                                                         *)
   200:D        0 (*      6.  9826 HPL Misc. I/O Doc.             ( ------------ )         *)
   201:D        0 (*                                                                         *)
   202:D        0 (*      7.  9826 card documentation             ( ----------- )          *)
   203:D        0 (*                                                                         *)
   204:D        0 (*      8.  Pascal I/O Library IRS              ( ------------ )         *)
   205:D        0 (*                                                                         *)
   206:D        0 (*                                                                         *)
   207:D        0 (*                                                                         *)
   208:D        0 (***************************************************************************)
```

```
   209:D        0 $PAGE$
   210:D        0 (*****************************************************************************)
   211:D        0 (*                                                                           *)
   212:D        0 (*                                                                           *)
   213:D        0 (*      GENERAL GROUP                                                        *)
   214:D        0 (*                                                                           *)
   215:D        0 (*                                                                           *)
   216:D        0 (*****************************************************************************)
   217:S
   218:S
   219:S
   220:D        0 MODULE general_1 ;
   221:S
   222:S                  {
   223:S                    date    07/15/81
   224:S                    update  11/20/81
   225:S
   226:S                    purpose This module contains the LEVEL 1 GENERAL GROUP procedures.
   227:D        1           }
   228:S
   229:S
   230:D        1 IMPORT  $SEARCH 'IOLIB:KERNEL.CODE', 'IOLIB:COMASM'$
   231:D        1           iodeclarations  ;
   232:S
   233:D        1 EXPORT
   234:S
   235:D        1   PROCEDURE ioinitialize;
   236:D        1   PROCEDURE iouninitialize;
   237:D        1   PROCEDURE ioreset     ( select_code : type_isc);
   238:D        1   PROCEDURE readchar    ( select_code : type_isc ;
   239:D        2                          VAR value   : CHAR );
   240:D        1   PROCEDURE writechar   ( select_code : type_isc ;
   241:D        2                          value       : CHAR );
   242:D        1   PROCEDURE readword    ( select_code : type_isc ;
   243:D        2                          VAR num     : INTEGER);
   244:D        1   PROCEDURE writeword   ( select_code : type_isc ;
   245:D        2                          value       : INTEGER);
   246:D        1   PROCEDURE set_timeout ( select_code : type_isc ;
   247:D        2                          time        : REAL );
   248:S
   249:S
   250:D        1 IMPLEMENT
   251:S
   252:D        1   IMPORT   general_0;
   253:S
   254:S
   255:S
   256:D        1   PROCEDURE ioinitialize;
   257:C        2   BEGIN
   258:S
   259:C        2     io_system_reset;
   260:S
   261:C        2   END; { of ioinitialize }
   262:S
   263:S
   264:S
   265:D        1   PROCEDURE iouninitialize;
   266:C        2   BEGIN
   267:S
   268:C        2     io_system_reset;
```



```
   269:S
   270:C        2   END; { of iouninitialize }
   271:S
   272:S
   273:S
   274:D        1   PROCEDURE ioreset     ( select_code : type_isc);
   275:C        2   BEGIN
   276:C        2     WITH isc_table[select_code] DO
   277:C        3     CALL(io_drv_ptr^.iod_init,
   278:C        3          io_tmp_ptr);
   279:C        2   END; { of ioreset }
   280:S
   281:S
   282:D        1   PROCEDURE readchar    ( select_code : type_isc ;
   283:D        2                          VAR value   : CHAR );
   284:C        2   BEGIN
   285:C        2     WITH isc_table[select_code] DO
   286:C        3     CALL(io_drv_ptr^.iod_rdb,
   287:C        3          io_tmp_ptr,
   288:C        3          value);
   289:C        2   END; { of readchar }
   290:S
   291:S
   292:S
   293:D        1   PROCEDURE writechar   ( select_code : type_isc ;
   294:D        2                          value       : CHAR );
   295:C        2   BEGIN
   296:C        2     WITH isc_table[select_code] DO
   297:C        3     CALL(io_drv_ptr^.iod_wtb,
   298:C        3          io_tmp_ptr,
   299:C        3          value);
   300:C        2   END; { of writechar }
   301:S
   302:S
   303:S
   304:D        1   PROCEDURE readword    ( select_code : type_isc ;
   305:D        2                          VAR num     : INTEGER);
   306:D    -2  2   VAR my_num : io_word;
   307:C        2   BEGIN
   308:C        2     WITH isc_table[select_code] DO
   309:C        3     CALL(io_drv_ptr^.iod_rdw,
   310:C        3          io_tmp_ptr,
   311:C        3          my_num);
   312:C        2     num:=my_num;
   313:C        2   END; { of readword }
   314:S
   315:S
   316:S
   317:D        1   PROCEDURE writeword   ( select_code : type_isc ;
   318:D        2                          value       : INTEGER);
   319:D    -2  2   VAR my_value : io_word;
   320:C        2   BEGIN
   321:C        2     my_value:=value;
   322:C        2     WITH isc_table[select_code] DO
   323:C        3     CALL(io_drv_ptr^.iod_wtw,
   324:C        3          io_tmp_ptr,
   325:C        3          my_value);
   326:C        2   END; { of writeword }
   327:S
   328:S
```

```
   329:S
   330:D        1   PROCEDURE set_timeout ( select_code : type_isc ;
   331:D    -8  2                           time        : REAL          { in seconds ) );
   332:C        2   BEGIN
   333:C        2     IF time>8191                { 4 byte timeout  - 1 byte left for shifts }
   334:C        3       THEN BEGIN
   335:C        3         { error }
   336:C        3         io_escape(ioe_bad_tmo,select_code);
   337:C        3       END; { of IF }
   338:C        2     IF (time>0) AND (time<0.001)
   339:C        3       THEN BEGIN
   340:C        3         { error }
   341:C        3         io_escape(ioe_bad_tmo,select_code);
   342:C        3       END; { of IF }
   343:S
   344:C        2     WITH isc_table[select_code] DO BEGIN
   345:S
   346:C        3       { the table entry used by drivers is in milliseconds }
   347:C        3       user_time:=ROUND(time*1000);
   348:S
   349:C        3       IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout := user_time;
   350:S
   351:C        3     END; { of WITH DO BEGIN }
   352:S
   353:C        2   END; { of set_timeout }
   354:S
   355:S
   356:C        1 END;    { of general_1 }
```



```
   357:D        1 $PAGE$
   358:D        1 (*************************************************************************)
   359:D        1 (*                                                                       *)
   360:D        1 (*                                                                       *)
   361:D        1 (*      HPIB GROUP LEVEL 1                                               *)
   362:D        1 (*                                                                       *)
   363:D        1 (*                                                                       *)
   364:D        1 (*************************************************************************)
   365:D        1 (*                                                                       *)
   366:D        1 (*                                                                       *)
   367:D        1 (*      This level is included in the                                    *)
   368:D        1 (*      general group because HP-IB                                      *)
   369:D        1 (*      addressing is necessary for                                      *)
   370:D        1 (*      general purpose device speci-                                    *)
   371:D        1 (*      fication.                                                        *)
   372:D        1 (*                                                                       *)
   373:D        1 (*************************************************************************)
   374:S
   375:S
   376:S
   377:D        1 MODULE hpib_1 ;
   378:S
   379:S                  {
   380:S                    date    07/16/81
   381:S                    update  08/03/83
   382:S
   383:S                    purpose This module contains the LEVEL 1 HPIB GROUP procedures.
   384:S
   385:D        1           }
   386:S
   387:S
   388:D        1 IMPORT    iodeclarations ;
   389:S
   390:D        1 EXPORT
   391:S
   392:S
   393:D        1   PROCEDURE send_command( select_code : type_isc ;
   394:D        2                           command     : CHAR );
   395:D        1   FUNCTION  my_address  ( select_code : type_isc)
   396:D        2                         : type_hpib_addr ;
   397:D        1   FUNCTION  active_controller
   398:D        2                         ( select_code : type_isc)
   399:D        2                         : BOOLEAN;
   400:D        1   FUNCTION  system_controller
   401:D        2                         ( select_code : type_isc)
   402:D        2                         : BOOLEAN;
   403:D        1   FUNCTION  addr_to_talk( device      : type_device)
   404:D        2                         : type_isc;
   405:D        1   FUNCTION  addr_to_listen
   406:D        2                         ( device      : type_device)
   407:D        2                         : type_isc;
   408:D        1   FUNCTION  set_to_talk ( device      : type_device)
   409:D        2                         : type_isc;
   410:D        1   FUNCTION  set_to_listen
   411:D        2                         ( device      : type_device)
   412:D        2                         : type_isc;
   413:D        1   FUNCTION  end_set      ( select_code : type_isc )
   414:D        2                         : BOOLEAN;
   415:S
   416:S
```

```
417:D        1 IMPLEMENT
418:S
419:D        1   IMPORT  iocomasm ,
420:D        1           general_0 ;
421:S
422:D        1   TYPE    timeoutrec = record                          { tttt JS 8/3/83 }
423:D        1                           counter: integer;            { tttt JS 8/3/83 }
424:D        1                           firsttime: boolean;          { tttt JS 8/3/83 }
425:D        1                         end;                           { tttt JS 8/3/83 }
426:S
427:S
428:D        1   FUNCTION timerexists: boolean;  exterral;          { tttt JS 8/3/83 }
429:S
430:D        1   FUNCTION timed_out(var rec: timeoutrec): boolean; external; {tttt JS 8/3/83}
431:S
432:D        1   PROCEDURE send_command( select_code : type_isc ;
433:D        2                           command     : CHAR );
434:C        2   BEGIN
435:C        2     WITH isc_table[select_code] DO
436:C        3       CALL ( io_drv_ptr^.iod_send,
437:C        3              io_tmp_ptr,
438:C        3              command);
439:C        2   END; { of send_command }
440:S
441:S
442:S
443:D        1   FUNCTION  my_address  ( select_code : type_isc)
444:D        2             : type_hpib_addr ;
445:C        2   BEGIN
446:C        2     IF isc_table[select_code].io_tmp_ptr <> NIL
447:C        3       THEN BEGIN
448:C        3         WITH isc_table[select_code].io_tmp_ptr^ DO
449:C        4           IF addressed <> -1
450:C        5             THEN BEGIN
451:C        5               my_address:=addressed;
452:C        5             END
453:C        5             ELSE BEGIN
454:C        5               { error }
455:C        5               io_escape(ioe_not_hpib,select_code);
456:C        5             END; { of IF addressed }
457:C        3       END
458:C        3       ELSE BEGIN
459:C        3         { error }
460:C        3         io_escape(ioe_not_hpib,select_code);
461:C        3       END; { of IF io_tmp_ptr }
462:C        2   END; { of my_address }
463:S
464:S
465:S
466:D        1   FUNCTION  active_controller
467:D        2                           ( select_code : type_isc)
468:D        2                           : BOOLEAN;
469:C        2   BEGIN
470:C        2     IF isc_table[select_code].card_type=hpib_card
471:C        3       THEN BEGIN
472:C        3         active_controller:=bit_set(iostatus(select_code,3),6);
473:C        3       END
474:C        3       ELSE BEGIN
475:C        3         active_controller := TRUE;
476:C        3       END; { of IF }
```



```
477:C        2   END; { of active_controller }
478:S
479:S
480:S
481:D        1   FUNCTION  system_controller
482:D        2                           ( select_code : type_isc)
483:D        2                           : BOOLEAN;
484:C        2   BEGIN
485:C        2     IF isc_table[select_code].card_type=hpib_card
486:C        3       THEN BEGIN
487:C        3         system_controller:=bit_set(iostatus(select_code,3),7);
488:C        3       END
489:C        3       ELSE BEGIN
490:C        3         system_controller := TRUE;
491:C        3       END; { of IF }
492:C        2   END; { of system_controller }
493:S
494:S
495:S
496:D        1   FUNCTION  end_set     ( select_code : type_isc )
497:D        2                         : BOOLEAN ;
498:D    -1  2   VAR mybool : BOOLEAN;
499:C        2   BEGIN
500:C        2     WITH isc_table[select_code] DO
501:C        3       CALL ( io_drv_ptr^.iod_end,
502:C        3              io_tmp_ptr,
503:C        3              mybool);
504:C        2     end_set := mybool;
505:C        2   END; { of send_command }
506:S
507:S
```

```
   508:D        1   $PAGE$
   509:S
   510:D        1   FUNCTION  addr_to_talk( device      : type_device)
   511:D        2                          : type_isc;
   512:D    -2  2   VAR io_isc : type_isc;
   513:D    -6  2       timer  : INTEGER;
   514:D   -12  2       hpibrec: timeoutrec;                                    {tttt JS 8/3/83}
   515:S
   516:C        2   BEGIN
   517:S
   518:C        2     IF device>iomaxisc
   519:C        3       THEN BEGIN
   520:C        3         io_isc:=device DIV 100;
   521:S
   522:C        3         WITH isc_table[io_isc] DO BEGIN
   523:S
   524:C        4           IF io_tmp_ptr <> NIL
   525:C        5             THEN BEGIN
   526:S
   527:C        5               { set up user timeout - in case system drivers changed it }
   528:C        5               io_tmp_ptr^.timeout:=user_time;
   529:S
   530:C        5               IF io_tmp_ptr^.addressed <> -1
   531:C        6                 THEN BEGIN
   532:C        6                   IF ( card_type <> hpib_card ) AND
   533:C        7                      ( device MOD 100 > 31 )
   534:C        7                     THEN io_escape(ioe_misc,io_isc);
   535:C        6                   send_command(io_isc,CHR(talk_constant+(device MOD 100)));
   536:C        6                   send_command(io_isc,'?');
   537:C        6                   send_command(io_isc,CHR(my_address(io_isc)+listen_constant));
   538:C        6                 END
   539:C        6                 ELSE BEGIN
   540:C        6                   { error }
   541:C        6                   io_escape(ioe_not_hpib,io_isc);
   542:C        6                 END; { of IF }
   543:C        5             END
   544:C        5             ELSE BEGIN
   545:C        5             END; { of IF }
   546:C        4         END; { of WITH DO BEGIN }
   547:C        3       END
   548:C        3       ELSE BEGIN
   549:C        3         io_isc:=device;
   550:C        3
   551:C        3         WITH isc_table[io_isc] DO BEGIN
   552:S
   553:C        4           { set up user timeout - in case system drivers changed it }
   554:C        4           IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout:=user_time;
   555:S
   556:C        4           IF card_type=hpib_card THEN
   557:C        5             BEGIN
   558:C        5               IF NOT active_controller(io_isc)
   559:C        6                 THEN BEGIN
   560:C        6                   { if non controller wait until listener }
   561:C        6                   IF user_time = 0
   562:C        7                     THEN BEGIN
   563:C        7                       REPEAT
   564:C        8                         { wait forever }
   565:C        8                       UNTIL bit_set(iostatus(io_isc,6),10);
   566:C        7                     END
   567:C        7                     ELSE BEGIN
```



```
   568:C        7                       { wait for timeout value }
   569:C        7                       IF timerexists THEN BEGIN              {tttt JS 8/3/83}
   570:C        8                         hpibrec.firsttime:=true;              {tttt JS 8/3/83}
   571:C        8                         hpibrec.counter:=user_time;           {tttt JS 8/3/83}
   572:C        8                         REPEAT                                {tttt JS 8/3/83}
   573:C        9                         UNTIL timed_out(hpibrec) OR           {tttt JS 8/3/83}
   574:C        9                           bit_set(iostatus(io_isc,6),10);     {tttt JS 8/3/83}
   575:C        8                       END                                     {tttt JS 8/3/83}
   576:C        8                       ELSE BEGIN                              {tttt JS 8/3/83}
   577:C        8                         timer:=user_time*3;
   578:C        8                         REPEAT
   579:C        9                           timer:=timer-1;
   580:C        9                         UNTIL ( timer = 0 ) OR
   581:C        9                               ( bit_set(iostatus(io_isc,6),10) ) ;
   582:C        8                       END;                                    {tttt JS 8/3/83}
   583:C        7                       IF NOT bit_set(iostatus(io_isc,6),10)
   584:C        8                         THEN io_escape(ioe_timeout,io_isc);
   585:C        7                     END; { of IF user_time=0 }
   586:C        6                 END; { of IF }
   587:C        5             END; { of IF card_type = hpib_card }
   588:C        4         END; { of WITH DO BEGIN }
   589:C        3       END; { of IF }
   590:S
   591:C        2     addr_to_talk:=io_isc; { return select code }
   592:S
   593:C        2   END; { of addr_to_talk }
   594:S
```

```
595:D        1   $PAGE$
596:S
597:D        1   FUNCTION  addr_to_listen
598:D        2                        ( device      : type_device)
599:D        2                        : type_isc;
600:D    -2  2   VAR io_isc : type_isc;
601:D    -6  2       timer  : INTEGER;
602:D   -12  2       hpibrec: timeoutrec;                                  {tttt JS 8/3/83}
603:S
604:C        2   BEGIN
605:S
606:C        2     IF device>iomaxisc
607:C        3       THEN BEGIN
608:C        3         io_isc:=device DIV 100;
609:S
610:C        3         WITH isc_table[io_isc] DO BEGIN
611:S
612:C        4           IF io_tmp_ptr <> NIL
613:C        5             THEN BEGIN
614:S
615:C        5               { set up user timeout - in case system drivers changed it }
616:C        5               io_tmp_ptr^.timeout:=user_time;
617:S
618:C        5               IF io_tmp_ptr^.addressed <> -1
619:C        6                 THEN BEGIN
620:C        6                   IF ( card_type <> hpib_card ) AND
621:C        7                      ( device MOD 100 > 31 )
622:C        7                     THEN io_escape(ioe_misc,io_isc);
623:C        6                   send_command(io_isc,CHR(my_address(io_isc)+talk_constant));
624:C        6                   send_command(io_isc,'?');
625:C        6                   send_command(io_isc,CHR(listen_constant+(device MOD 100)));
626:C        6                 END
627:C        6                 ELSE BEGIN
628:C        6                   { error }
629:C        6                   io_escape(ioe_not_hpib,io_isc);
630:C        6                 END; { of IF }
631:C        5             END
632:C        5             ELSE BEGIN
633:C        5             END; { of IF }
634:C        4         END; { of WITH DO BEGIN }
635:C        3       END
636:C        3       ELSE BEGIN
637:C        3         io_isc:=device;
638:C        3
639:C        3         WITH isc_table[io_isc] DO BEGIN
640:S
641:C        4           { set up user timeout - in case system drivers changed it }
642:C        4           IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout:=user_time;
643:S
644:C        4           IF card_type=hpib_card THEN
645:C        5             BEGIN
646:C        5               IF NOT active_controller(io_isc)
647:C        6                 THEN BEGIN
648:C        6                   { if non controller wait until talker }
649:C        6                   IF user_time = 0
650:C        7                     THEN BEGIN
651:C        7                       REPEAT
652:C        8                         { wait forever }
653:C        8                       UNTIL bit_set(iostatus(io_isc,6),9);
654:C        7                     END
```



```
655:C        7                     ELSE BEGIN
656:C        7                       { wait for timeout value }
657:C        7                       IF timerexists THEN BEGIN              {tttt JS 8/3/83}
658:C        8                         hpibrec.firsttime:=true;              {tttt JS 8/3/83}
659:C        8                         hpibrec.counter:=user_time;           {tttt JS 8/3/83}
660:C        8                         REPEAT                                {tttt JS 8/3/83}
661:C        9                         UNTIL timed_out(hpibrec) OR           {tttt JS 8/3/83}
662:C        9                           bit_set(iostatus(io_isc,6), 9);     {tttt JS 8/3/83}
663:C        8                       END                                     {tttt JS 8/3/83}
664:C        8                       ELSE BEGIN                              {tttt JS 8/3/83}
665:C        8                         timer:=user_time*3;
666:C        8                         REPEAT
667:C        9                           timer:=timer-1;
668:C        9                         UNTIL ( timer = 0 ) OR
669:C        9                               ( bit_set(iostatus(io_isc,6),9) ) ;
670:C        8                       END;                                    {tttt JS 8/3/83}
671:C        7                       IF NOT bit_set(iostatus(io_isc,6),9)
672:C        8                         THEN io_escape(ioe_timeout,io_isc);
673:C        7                     END; { of IF user_time=0 }
674:C        6                 END; { of IF }
675:C        5             END; { of IF card_type = hpib_card }
676:C        4         END; { of WITH DO BEGIN }
677:C        3       END; { of IF }
678:S
679:C        2     addr_to_listen:=io_isc;
680:S
681:C        2   END; { of addr_to_listen }
682:S
```

```
683:D        1   $PAGE$
684:S
685:S
686:S            { set to talk exists because of HPIB_2/HPIB_3 -
687:S              those routines are intended to be the controller
688:S              ( active ) and should not wait for the card to be
689:S              addressed as talker.  addr_to_talk is used by
690:S              data transfer routines.  set_to_talk is used by
691:D        1     bus control routines.                          }
692:S
693:S
694:S
695:S
696:D        1   FUNCTION  set_to_talk ( device      : type_device)
697:D        2                          : type_isc;
698:D    -2  2   VAR io_isc : type_isc;
699:C        2   BEGIN
700:S
701:C        2     IF device>iomaxisc
702:C        3       THEN BEGIN
703:C        3         io_isc:=addr_to_talk(device);
704:C        3       END
705:C        3       ELSE BEGIN
706:C        3         io_isc:=device;
707:C        3
708:C        3         WITH isc_table[io_isc] DO BEGIN
709:S
710:C        4           { set up user timeout - in case system drivers changed it }
711:C        4           IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout:=user_time;
712:S
713:C        4           IF card_type=hpib_card THEN
714:C        5             BEGIN
715:C        5               IF NOT active_controller(io_isc)
716:C        6                 THEN BEGIN
717:S
718:C        6                   io_escape(ioe_not_act,io_isc);
719:S
720:C        6                 END; { of IF }
721:C        5             END; { of IF card_type = hpib_card }
722:C        4         END; { of WITH DO BEGIN }
723:C        3       END; { of IF }
724:S
725:C        2     set_to_talk:=io_isc; { return select code }
726:S
727:C        2   END; { of set_to_talk }
728:S
```



```
729:D        1   $PAGE$
730:S
731:S
732:S            { set to listen exists because of HPIB_2/HPIB_3 -
733:S              those routines are intended to be the controller
734:S              ( active ) and should not wait for the card to be
735:S              addressed as listener.  addr_to_listen is used by
736:S              data transfer routines.  set_to_listen is used by
737:D        1     bus control routines.                          }
738:S
739:S
740:S
741:S
742:S
743:D        1   FUNCTION  set_to_listen
744:D        2                          ( device      : type_device)
745:D        2                          : type_isc;
746:D    -2  2   VAR io_isc : type_isc;
747:D    -6  2       timer  : INTEGER;
748:C        2   BEGIN
749:S
750:C        2     IF device>iomaxisc
751:C        3       THEN BEGIN
752:C        3         io_isc:=addr_to_listen(device);
753:S
754:C        3       END
755:C        3       ELSE BEGIN
756:C        3         io_isc:=device;
757:C        3
758:C        3         WITH isc_table[io_isc] DO BEGIN
759:S
760:C        4           { set up user timeout - in case system drivers changed it }
761:C        4           IF io_tmp_ptr <> NIL THEN io_tmp_ptr^.timeout:=user_time;
762:S
763:C        4           IF card_type=hpib_card THEN
764:C        5             BEGIN
765:C        5               IF NOT active_controller(io_isc)
766:C        6                 THEN BEGIN
767:S
768:C        6                   io_escape(ioe_not_act,io_isc);
769:S
770:C        6                 END; { of IF }
771:C        5             END; { of IF card_type = hpib_card }
772:C        4         END; { of WITH DO BEGIN }
773:C        3       END; { of IF }
774:S
775:C        2     set_to_listen:=io_isc;
776:S
777:C        2   END; { of set_to_listen }
778:S
779:S
780:S
781:C        1 END;    { of hpib_1 }
```

```
 782:D        1 $PAGE$
 783:D        1 MODULE general_2 ;
 784:S
 785:S                  {
 786:S                    date    07/15/81
 787:S                    update  11/30/81
 788:S
 789:S                    purpose This module contains the LEVEL 2 GENERAL GROUP procedures.
 790:S
 791:D        1           }
 792:S
 793:D        1 IMPORT    iodeclarations;
 794:S
 795:D        1 EXPORT
 796:D        1
 797:D        1   PROCEDURE readnumber ( device : type_device ;
 798:D        2                           VAR num: REAL ) ;
 799:D        1   PROCEDURE writenumber( device : type_device ;
 800:D        2                           value  : REAL ) ;
 801:D        1   PROCEDURE readstring
 802:D        2                         ( device : type_device ;
 803:D        2                           VAR str: STRING ) ;
 804:D        1   PROCEDURE readstring_until
 805:D        2                         ( term   : CHAR ;
 806:D        2                           device : type_device ;
 807:D        2                           VAR str: STRING );
 808:D        1   PROCEDURE writestring( device : type_device ;
 809:D        2                           str    : io_STRING ) ;
 810:D        1   PROCEDURE readnumberln
 811:D        2                         ( device : type_device ;
 812:D        2                           VAR num: REAL );
 813:D        1   PROCEDURE writenumberln
 814:D        2                         ( device : type_device ;
 815:D        2                           value  : REAL );
 816:D        1   PROCEDURE writestringln
 817:D        2                         ( device : type_device ;
 818:D        2                           str    : io_STRING );
 819:D        1   PROCEDURE readuntil  ( term   : CHAR ;
 820:D        2                           device : type_device );
 821:D        1   PROCEDURE skipfor    ( count  : INTEGER ;
 822:D        2                           device : type_device );
 823:S
 824:S
 825:D        1 IMPLEMENT
 826:D        1 IMPORT    sysglobals,
 827:D        1           hpib_1 ,
 828:D        1           general_1 ;
 829:S
 830:S
 831:D        1
 832:D        1   PROCEDURE readnumber ( device : type_device ;
 833:D        2                           VAR num: REAL ) ;
 834:D  -256  2   VAR io_work_str : STRING[255];
 835:D  -260  2       i           : INTEGER;
 836:D  -264  2       p2          : INTEGER;
 837:D  -266  2       io_isc      : type_isc;
 838:D  -267  2       numbuilt    : BOOLEAN;
 839:S
 840:D        2     FUNCTION  numeric ( character : CHAR) : BOOLEAN;
 841:C        3     BEGIN
```



```
 842:C        3       CASE character OF
 843:C        4         '0'..'9',
 844:C        4         '+','-','.',
 845:C        4         'E','e'     : numeric:=TRUE
 846:S
 847:C        4         OTHERWISE     numeric:=FALSE
 848:C        4       END; { of CASE }
 849:C        3     END; { of numeric }
 850:C        2   BEGIN
 851:C        2     { use TRY RECOVER to build a number until I find one }
 852:S
 853:C        2     io_isc:=addr_to_talk(device);
 854:S
 855:C        2     numbuilt := FALSE;
 856:S
 857:C        2     WITH isc_table[io_isc].io_drv_ptr^,
 858:C        3          isc_table[io_isc]            DO BEGIN
 859:S
 860:C        3       REPEAT
 861:S
 862:C        4         SETSTRLEN(io_work_str,255);
 863:C        4         i:=1;
 864:S
 865:C        4         { skip over non-numeric characters }
 866:C        4         REPEAT
 867:C        5           CALL ( iod_rdb , io_tmp_ptr , io_work_str[i]);
 868:C        5           WHILE io_work_str[i]=' ' DO
 869:C        6             CALL ( iod_rdb , io_tmp_ptr , io_work_str[i]);
 870:C        5         UNTIL numeric(io_work_str[i]) ;
 871:S
 872:C        4         { read in numeric characters }
 873:C        4         REPEAT
 874:C        5           i:=i+1;
 875:C        5           CALL ( iod_rdb , io_tmp_ptr , io_work_str[i] );
 876:C        5           WHILE io_work_str[i]=' ' DO
 877:C        6             CALL ( iod_rdb , io_tmp_ptr , io_work_str[i]);
 878:C        5         UNTIL ( (NOT ( numeric(io_work_str[i]))) OR (
 879:C        5               ( i>=255) ) );
 880:S
 881:S
 882:C        4         SETSTRLEN(io_work_str,i);
 883:C        4         io_work_char:=io_work_str[i];
 884:S
 885:C        4         TRY
 886:S
 887:C        5           STRREAD(io_work_str,1,p2,num);
 888:C        5
 889:C        5           numbuilt := TRUE;
 890:S
 891:C        5         RECOVER BEGIN
 892:C        5           IF ( ESCAPECODE=-10 )                    AND
 893:C        6              (   ( IORESULT = ORD(IBADFORMAT) )  OR
 894:C        6                  ( IORESULT = ORD(ISTROVFL)   ) )
 895:C        6             THEN BEGIN
 896:C        6               { this is the strread errors - try again }
 897:C        6             END
 898:C        6             ELSE BEGIN
 899:C        6               { this means something else happened }
 900:C        6               ESCAPE(ESCAPECODE);
 901:C        6             END; { of IF my error }
```

```
 902:C        5         END; { of RECOVER }
 903:S
 904:C        4       UNTIL numbuilt;
 905:S
 906:S
 907:C        3     END; { of WITH DO BEGIN }
 908:S
 909:C        2   END; { of readnumber  }
 910:S
 911:S
 912:S
 913:D        1   PROCEDURE writenumber (device : type_device ;
 914:D    -8  2                          value  : REAL );
 915:D   -12  2   VAR i           : INTEGER;
 916:D   -16  2       p2          : INTEGER;
 917:D   -18  2       io_isc      : type_isc;
 918:D  -274  2       io_work_str : STRING[255];
 919:C        2   BEGIN
 920:C        2     io_isc:=addr_to_listen(device);
 921:S
 922:C        2     WITH isc_table[io_isc].io_drv_ptr^,
 923:C        3          isc_table[io_isc]           DO BEGIN
 924:S
 925:C        3       STRWRITE(io_work_str,1,p2,value);
 926:C        3       FOR i:=1 TO p2-1 DO
 927:C        4         CALL ( iod_wtb , io_tmp_ptr , io_work_str[i]);
 928:C        3
 929:C        3     END; { of WITH DO }
 930:S
 931:C        2   END; { of writenumber  }
 932:S
 933:S
 934:S
 935:D        1   PROCEDURE readstring
 936:D        2                          ( device : type_device ;
 937:D        2                            VAR str: STRING ) ;
 938:D    -4  2   VAR i         : INTEGER;
 939:D    -6  2       io_isc    : type_isc;
 940:C        2   BEGIN
 941:C        2     io_isc:=addr_to_talk(device);
 942:S
 943:C        2     WITH isc_table[io_isc].io_drv_ptr^,
 944:C        3          isc_table[io_isc]           DO BEGIN
 945:S
 946:C        3       SETSTRLEN(str,STRMAX(str));         { so I can do assign to empty string }
 947:C        3       i:=0;
 948:C        3       REPEAT
 949:C        4         i:=i+1;
 950:C        4         CALL ( iod_rdb , io_tmp_ptr , str[i]);
 951:C        4       UNTIL ( (i>=STRMAX(str)          )    OR
 952:C        4               ( str[i] = io_line_feed ) );
 953:C        3       IF str[i]=io_line_feed THEN i:=i-1;
 954:C        3       IF i<>0 THEN IF str[i]=io_carriage_rtn THEN i:=i-1;
 955:C        3       SETSTRLEN(str,i);
 956:S
 957:C        3     END; { of WITH DO BEGIN }
 958:S
 959:C        2   END; { of readstring }
 960:S
 961:S
```



```
 962:S
 963:S
 964:D        1   PROCEDURE readstring_until
 965:D        2                          ( term   : CHAR ;
 966:D        2                            device : type_device ;
 967:D        2                            VAR str: STRING );
 968:D    -4  2   VAR i           : INTEGER;
 969:D    -6  2       io_isc      : type_isc;
 970:D    -7  2       io_work_char: CHAR;
 971:C        2   BEGIN
 972:C        2     io_isc:=addr_to_talk(device);
 973:S
 974:C        2     WITH isc_table[io_isc].io_drv_ptr^,
 975:C        3          isc_table[io_isc]           DO BEGIN
 976:S
 977:C        3       SETSTRLEN(str,STRMAX(str));
 978:C        3       i:=0;
 979:C        3       REPEAT
 980:C        4         i:=i+1;
 981:C        4         CALL ( iod_rdb , io_tmp_ptr , str[i]);
 982:C        4       UNTIL ( (i>=STRMAX(str) )    OR
 983:C        4               ( str[i]=term ) );
 984:C        3       SETSTRLEN(str,i);
 985:S
 986:C        3     END; { of WITH DO BEGIN }
 987:S
 988:C        2   END; { of readstring_until }
 989:S
 990:S
 991:S
 992:D        1   PROCEDURE writestring( device : type_device ;
 993:D  -256  2                          str    : io_STRING ) ;
 994:D  -260  2   VAR i     : INTEGER;
 995:D  -262  2       io_isc: type_isc;
 996:C        2   BEGIN
 997:C        2     io_isc:=addr_to_listen(device);
 998:S
 999:C        2     WITH isc_table[io_isc].io_drv_ptr^,
1000:C        3          isc_table[io_isc]           DO BEGIN
1001:S
1002:C        3       FOR i:=1 TO STRLEN(str) DO
1003:C        4         CALL ( iod_wtb , io_tmp_ptr , str[i]);
1004:C        3
1005:C        3     END; { of WITH DO }
1006:S
1007:C        2   END; { of writestring }
1008:S
1009:S
1010:S
1011:D        1   PROCEDURE readnumberln
1012:D        2                          ( device : type_device ;
1013:D        2                            VAR num: REAL );
1014:D    -2  2   VAR io_isc        : type_isc;
1015:C        2   BEGIN
1016:C        2     io_isc:=addr_to_talk(device);
1017:C        2     readnumber(io_isc,num);
1018:C        2     IF io_work_char <> io_line_feed THEN
1019:C        3       readuntil(io_line_feed,io_isc);
1020:C        2   END; { of readnumberln }
1021:S
```

```
1022:S
1023:S
1024:D    -8  1   PROCEDURE writenumberln
1025:D        2                        ( device : type_device ;
1026:D    -8  2                          value  : REAL );
1027:D   -10  2   VAR io_isc       : type_isc;
1028:C        2   BEGIN
1029:C        2     io_isc:=addr_to_listen(device);
1030:C        2     writenumber(io_isc,value);
1031:C        2     writechar(io_isc,io_carriage_rtn);
1032:C        2     writechar(io_isc,io_line_feed);
1033:C        2   END; { of writenumberln }
1034:S
1035:S
1036:S
1037:D  -256  1   PROCEDURE writestringln
1038:D        2                        ( device : type_device ;
1039:D  -256  2                          str    : io_STRING );
1040:D  -258  2   VAR io_isc       : type_isc;
1041:C        2   BEGIN
1042:C        2     io_isc:=addr_to_listen(device);
1043:C        2     writestring(io_isc,str);
1044:C        2     writechar(io_isc,io_carriage_rtn);
1045:C        2     writechar(io_isc,io_line_feed);
1046:C        2   END; { of writestringln }
1047:S
1048:S
1049:S
1050:D        1   PROCEDURE readuntil  ( term   : CHAR ;
1051:D        2                          device : type_device );
1052:D    -1  2   VAR io_work_char: CHAR;
1053:D    -4  2       io_isc       : type_isc;
1054:C        2   BEGIN
1055:C        2     io_isc:=addr_to_talk(device);
1056:S
1057:C        2     WITH isc_table[io_isc].io_drv_ptr^ ,
1058:C        3          isc_table[io_isc]            DO BEGIN
1059:S
1060:C        3       REPEAT
1061:C        4         CALL ( iod_rdb , io_tmp_ptr , io_work_char);
1062:C        4       UNTIL ( io_work_char=term );
1063:C        3
1064:C        3     END; { of WITH DO BEGIN }
1065:S
1066:C        2   END; { of readuntil }
1067:S
1068:S
1069:S
1070:D        1   PROCEDURE skipfor    ( count  : INTEGER ;
1071:D        2                          device : type_device );
1072:D    -4  2   VAR i           : INTEGER;
1073:D    -6  2       io_isc      : type_isc;
1074:D    -7  2       io_work_char: CHAR;
1075:C        2   BEGIN
1076:C        2     io_isc:=addr_to_talk(device);
1077:S
1078:C        2     WITH isc_table[io_isc].io_drv_ptr^ ,
1079:C        3          isc_table[io_isc]            DO BEGIN
1080:S
1081:C        3       FOR i:=1 TO count DO
```



```
1082:C        4         CALL ( iod_rdb , io_tmp_ptr , io_work_char);
1083:C        3
1084:C        3     END; { of WITH DO BEGIN }
1085:S
1086:C        2   END; { of skipfor }
1087:S
1088:S
1089:S
1090:C        1 END;  { of general_2 }
```

```
1091:D        1 $PAGE$
1092:D        1 MODULE general_3 ;
1093:S
1094:S                  {
1095:S                    date    11/27/81
1096:S                    update  07/23/82
1097:S
1098:S                    purpose This module contains the LEVEL 3 GENERAL GROUP procedures.
1099:S
1100:D        1           }
1101:S
1102:S
1103:D        1 IMPORT    iodeclarations ;
1104:S
1105:D        1 EXPORT
1106:S
1107:D        1   FUNCTION  ioerror_message ( ioerror : INTEGER )  : io_STRING;
1108:S
1109:D        1 IMPLEMENT
1110:S
1111:D        1   FUNCTION  ioerror_message ( ioerror : INTEGER ) : io_STRING;
1112:D  -256  2   VAR my_msg : io_STRING;
1113:C        2   BEGIN
1114:C        2     my_msg:='zzzz' ;                                          { 0082 TM 7/23/82 }
1115:S
1116:C        2     IF ( ioerror <= ioe_misc )     AND
1117:C        3        ( ioerror >= ioe_no_error )
1118:C        3       THEN BEGIN
1119:C        3         CASE ioerror OF
1120:S
1121:C        4           ioe_no_error     :  my_msg := 'no error ';
1122:C        4           ioe_no_card      :  my_msg := 'no card at select code';
1123:C        4           ioe_not_hpib     :  my_msg := 'interface should be hpib';
1124:C        4           ioe_not_act      :  my_msg := 'not active controller';
1125:C        4           ioe_not_dvc      :  my_msg := 'should be device not sc';    { BUG 1281 TM 1/8/82 }
1126:C        4           ioe_no_space     :  my_msg := 'no space left in buffer';
1127:C        4           ioe_no_data      :  my_msg := 'no data left in buffer';
1128:C        4           ioe_bad_tfr      :  my_msg := 'improper transfer attempted';
1129:C        4           ioe_isc_busy     :  my_msg := 'the select code is busy';
1130:C        4           ioe_buf_busy     :  my_msg := 'the buffer is busy';
1131:C        4           ioe_bad_cnt      :  my_msg := 'improper transfer count';
1132:C        4           ioe_bad_tmo      :  my_msg := 'bad timeout value';
1133:C        4           ioe_no_driver    :  my_msg := 'no driver for this card';
1134:C        4           ioe_no_dma       :  my_msg := 'no dma';
1135:C        4           ioe_no_word      :  my_msg := 'word operations not allowed';
1136:C        4           ioe_not_talk     :  my_msg := 'not addressed as talker';
1137:C        4           ioe_not_lstn     :  my_msg := 'not addressed as listener';
1138:C        4           ioe_timeout      :  my_msg := 'a timeout has occurred';
1139:C        4           ioe_not_sctl     :  my_msg := 'not system controller';
1140:C        4           ioe_rds_wtc      :  my_msg := 'bad status or control';
1141:C        4           ioe_bad_sct      :  my_msg := 'bad set/clear/test operation';
1142:C        4           ioe_crd_dwn      :  my_msg := 'interface card is dead';
1143:C        4           ioe_eod_seen     :  my_msg := 'end/eod has occured';
1144:C        4           ioe_misc         :  my_msg := 'miscellaneous - value of param error';
1145:C        4         END; { of CASE }
1146:C        3       END; { of IF }
1147:S
1148:C        2     IF ( ioerror >= ioe_dc_fail ) AND
1149:C        3        ( ioerror <= ioe_dc_rval )
1150:C        3       THEN BEGIN
```



```
1151:C        3         CASE ioerror OF
1152:S
1153:C        4           ioe_sr_toomany  :  my_msg :=                       { 0364 TM 87/23/82 }
1154:C        4                              'too many chars w/o terminator';  { 0364 TM 87/23/82 }
1155:C        4           ioe_dc_fail     :  my_msg := 'dc interface failure';
1156:C        4           ioe_dc_usart    :  my_msg := 'USART receive buffer overflow';
1157:C        4           ioe_dc_ovfl     :  my_msg := 'receive buffer overflow';
1158:C        4           ioe_dc_clk      :  my_msg := 'missing clock';
1159:C        4           ioe_dc_cts      :  my_msg := 'CTS false too long';
1160:C        4           ioe_dc_car      :  my_msg := 'lost carrier disconnect';
1161:C        4           ioe_dc_act      :  my_msg := 'no activity disconnect';
1162:C        4           ioe_dc_conn     :  my_msg := 'connection not established';
1163:C        4           ioe_dc_conf     :  my_msg := 'bad data bits/parity combination';
1164:C        4           ioe_dc_reg      :  my_msg := 'bad status /control register';
1165:C        4           ioe_dc_rval     :  my_msg := 'control value out of range';
1166:C        4         END; { of CASE }
1167:C        3       END; { of IF }
1168:S
1169:C        2     IF ioe_result = ioe_sr_fail                             { 0364 TM 8/23/82 }
1170:C        3       THEN my_msg := 'data link failure';                   { 0364 TM 8/23/82 }
1171:S
1172:C        2     IF my_msg = 'zzzz'     { we don't let sleeping msgs lie }  { 0082 TM 7/23/82 }
1173:C        3       THEN CALL(io_error_link , ioerror , my_msg  );        { 0082 TM 7/23/82 }
1174:S
1175:C        2     ioerror_message := my_msg;
1176:S
1177:C        2   END; { ioerror_message }
1178:S
1179:C        1 END;  { of general_3 }
```

```
1180:D        1 $PAGE$
1181:D        1 MODULE general_4 ;
1182:S
1183:S                  {
1184:S                    date    07/17/81
1185:S                    update  07/23/82
1186:S
1187:S                    purpose This module contains the LEVEL 4 GENERAL GROUP procedures.
1188:S
1189:D        1           }
1190:S
1191:S
1192:D        1 IMPORT    iodeclarations ;
1193:S
1194:D        1 EXPORT
1195:D        1
1196:D        1   PROCEDURE abort_transfer
1197:D        2                         ( VAR b_info: buf_info_type );
1198:D        1   FUNCTION  transfer_setup
1199:D        2                         ( device    : type_device;
1200:D        2                           t_tfr     : user_tfr_type;
1201:D        2                           t_dir     : dir_of_tfr ;
1202:D        2                           VAR b_info: buf_info_type ;
1203:D        2                           VAR t_cnt : INTEGER )
1204:D        2                         : type_isc ;
1205:D        1   PROCEDURE transfer   ( device    : type_device;
1206:D        2                           t_tfr     : user_tfr_type;
1207:D        2                           t_dir     : dir_of_tfr ;
1208:D        2                           VAR b_info: buf_info_type;
1209:D        2                           x_count   : INTEGER ) ;
1210:D        1   PROCEDURE transfer_word
1211:D        2                         ( device    : type_device;
1212:D        2                           t_tfr     : user_tfr_type;
1213:D        2                           t_dir     : dir_of_tfr ;
1214:D        2                           VAR b_info: buf_info_type;
1215:D        2                           x_count   : INTEGER ) ;
1216:D        1   PROCEDURE transfer_until
1217:D        2                         ( term      : CHAR ;
1218:D        2                           device    : type_device;
1219:D        2                           t_tfr     : user_tfr_type;
1220:D        2                           t_dir     : dir_of_tfr;
1221:D        2                           VAR b_info: buf_info_type ) ;
1222:D        1   PROCEDURE transfer_end
1223:D        2                         ( device    : type_device;
1224:D        2                           t_tfr     : user_tfr_type;
1225:D        2                           t_dir     : dir_of_tfr;
1226:D        2                           VAR b_info: buf_info_type ) ;
1227:S
1228:D        1   PROCEDURE iobuffer   ( VAR b_info: buf_info_type ;
1229:D        2                           t_count   : INTEGER );
1230:D        1   PROCEDURE buffer_reset(VAR b_info: buf_info_type ) ;
1231:D        1   FUNCTION  buffer_space(VAR b_info: buf_info_type)
1232:D        2                         : INTEGER;
1233:D        1   FUNCTION  buffer_data( VAR b_info: buf_info_type)
1234:D        2                         : INTEGER;
1235:D        1   PROCEDURE readbuffer ( VAR b_info: buf_info_type;
1236:D        2                           VAR value : CHAR);
1237:D        1   PROCEDURE writebuffer( VAR b_info: buf_info_type;
1238:D        2                           value     : CHAR);
1239:D        1   PROCEDURE readbuffer_string
```



```
1240:D        2                         ( VAR b_info: buf_info_type;
1241:D        2                           VAR str   : STRING;
1242:D        2                           str_count : INTEGER);
1243:D        1   PROCEDURE writebuffer_string
1244:D        2                         ( VAR b_info: buf_info_type;
1245:D        2                           str       : io_STRING);
1246:S
1247:D        1   FUNCTION  buffer_busy( VAR b_info: buf_info_type )            { 0083 TM 7/23/82 }
1248:D        2                         : BOOLEAN;                             { 0083 TM 7/23/82 }
1249:D        1   FUNCTION  isc_busy    ( isc       : type_isc )               { 0083 TM 7/23/82 }
1250:D        2                         : BOOLEAN;                             { 0083 TM 7/23/82 }
1251:S
1252:D        1 IMPLEMENT
1253:S
1254:S
1255:D        1   IMPORT  hpib_1 ;
1256:S
1257:D        1   PROCEDURE iobuffer   ( VAR b_info: buf_info_type ;
1258:D        2                           t_count   : INTEGER ) ;
1259:D        2     PROCEDURE NEW $ALIAS '#SM_NEWBYTES'$
***WARNING: (line 1260): External declarations will be treated as global
1260:D        3                   (VAR p:ANYPTR;v:INTEGER);EXTERNAL;
1261:C        2   BEGIN
1262:C        2     WITH b_info DO
1263:C        3       BEGIN
1264:C        3         { what about IOBUFFER to a already existant buffer ? }
1265:C        3         { - the space will be thrown away. }
1266:S
1267:C        3         NEW(buf_ptr,t_count);
1268:C        3
1269:C        3         act_tfr   := no_tfr;
1270:C        3         active_isc:= no_isc;
1271:C        3         buf_size  := t_count;
1272:C        3         buf_empty := buf_ptr;
1273:C        3         buf_fill  := buf_ptr;
1274:S
1275:C        3         drv_tmp_ptr       := NIL;
1276:C        3         eot_proc.dummy_s1 := NIL;
1277:C        3         eot_proc.dummy_pr := NIL;
1278:C        3         eot_parm          := NIL;                 {JPC  02/22/82}
1279:C        3         dma_priority      := FALSE ;
1280:C        3       END; { of WITH DO }
1281:C        2   END; { of iobuffer }
1282:S
1283:S
1284:S
1285:D        1   FUNCTION  buffer_data(VAR b_info : buf_info_type )
1286:D        2                       : INTEGER;
1287:C        2   BEGIN
1288:C        2     WITH b_info
1289:C        3     DO BEGIN
1290:C        3       buffer_data:=INTEGER(buf_fill)-INTEGER(buf_empty);
1291:C        3     END; { of WITH DO }
1292:C        2   END; { of buffer_data }
1293:S
1294:S
1295:S
1296:D        1   PROCEDURE buffer_reset(VAR b_info: buf_info_type ) ;
1297:C        2   BEGIN
1298:C        2     WITH b_info
```

```
1299:C        3     DO BEGIN
1300:C        3       IF active_isc = no_isc
1301:C        4         THEN BEGIN
1302:C        4           buf_fill:=buf_ptr;
1303:C        4           buf_empty:=buf_ptr;
1304:C        4         END
1305:C        4         ELSE BEGIN
1306:C        4           { error }
1307:C        4           io_escape(ioe_buf_busy,no_isc);
1308:C        4         END; { of IF }
1309:C        3     END; { of WITH DO }
1310:C        2   END; { of buffer_reset }
1311:S
1312:S
1313:S
1314:D        1   FUNCTION  buffer_space(VAR b_info: buf_info_type)
1315:D        2                         : INTEGER;
1316:C        2   BEGIN
1317:C        2     WITH b_info
1318:C        3     DO BEGIN
1319:C        3       IF ( buffer_data(b_info)=0 ) AND
1320:C        4          ( active_isc = no_isc )   THEN buffer_reset(b_info);
1321:C        3       buffer_space:=buf_size+INTEGER(buf_ptr)-INTEGER(buf_fill);
1322:C        3     END; { of WITH DO }
1323:C        2   END; { of buffer_space }
1324:S
1325:S
1326:S
1327:D        1   PROCEDURE abort_transfer
1328:D        2                         ( VAR b_info: buf_info_type );
1329:C        2   BEGIN
1330:C        2     WITH b_info
1331:C        3     DO BEGIN
1332:S
1333:C        3       IF active_isc <> no_isc
1334:C        4         THEN BEGIN
1335:S
1336:C        4           WITH isc_table[active_isc] DO
1337:C        5           CALL ( io_drv_ptr^.iod_init ,
1338:C        5                  io_tmp_ptr );
1339:S
1340:S
1341:C        4         END; { of IF }
1342:S
1343:C        3     END; { of WITH b_info DO }
1344:S
1345:C        2   END; { of abort_transfer }
1346:S
1347:S
1348:S
1349:S
1350:D        1   FUNCTION  transfer_setup
1351:D        2                         ( device    : type_device;
1352:D        2                           t_tfr     : user_tfr_type;
1353:D        2                           t_dir     : dir_of_tfr ;
1354:D        2                           VAR b_info: buf_info_type ;
1355:D        2                           VAR t_cnt : INTEGER )
1356:D        2                         : type_isc ;
1357:D    -2  2   VAR io_isc : type_isc;
1358:C        2   BEGIN
```



```
1359:S
1360:C        2     IF device>iomaxisc THEN io_isc := device DIV 100
1361:C        3                        ELSE io_isc := device;
1362:S
1363:C        2     IF isc_table[io_isc].io_tmp_ptr = NIL
1364:C        3                        THEN io_escape(ioe_no_driver,io_isc);
1365:S
1366:S
1367:C        2     WITH b_info
1368:C        3     DO BEGIN
1369:S
1370:C        3       { test for tfr count }
1371:C        3       IF t_cnt=0
1372:C        4         THEN BEGIN
1373:C        4           { error }
1374:C        4           io_escape(ioe_bad_cnt,no_isc);
1375:C        4         END;
1376:S
1377:C        3       { test for another tfr on this buffer }
1378:C        3       IF active_isc <> no_isc
1379:C        4         THEN BEGIN
1380:C        4           { error }
1381:C        4           io_escape(ioe_buf_busy,no_isc);
1382:C        4         END
1383:C        4         ELSE BEGIN
1384:C        4           IF buffer_data(b_info)=0 THEN buffer_reset(b_info);
1385:C        4         END; { of IF }
1386:S
1387:S
1388:C        3       { configure card based on direction and check for available space/data }
1389:C        3       IF t_dir= to_memory
1390:C        4         THEN BEGIN
1391:C        4           IF isc_table[io_isc].io_tmp_ptr^.in_bufptr <> NIL
1392:C        5             THEN BEGIN
1393:C        5               { error }
1394:C        5               io_escape(ioe_isc_busy,io_isc);
1395:C        5             END; { of IF }
1396:C        4           IF buffer_space(b_info)<t_cnt
1397:C        5             THEN BEGIN
1398:C        5               { error }
1399:C        5               io_escape(ioe_no_space,io_isc);
1400:C        5             END; { of IF }
1401:C        4           io_isc:=addr_to_talk(device);
1402:C        4           isc_table[io_isc].io_tmp_ptr^.in_bufptr :=  ADDR( b_info );
1403:C        4         END
1404:C        4         ELSE BEGIN
1405:C        4           IF isc_table[io_isc].io_tmp_ptr^.out_bufptr <> NIL
1406:C        5             THEN BEGIN
1407:C        5               { error }
1408:C        5               io_escape(ioe_isc_busy,io_isc);
1409:C        5             END; { of IF }
1410:C        4           IF buffer_data(b_info)<t_cnt
1411:C        5             THEN BEGIN
1412:C        5               { error }
1413:C        5               io_escape(ioe_no_data,io_isc);
1414:C        5             END; { of IF }
1415:C        4           io_isc:=addr_to_listen(device);
1416:C        4           isc_table[io_isc].io_tmp_ptr^.out_bufptr :=  ADDR( b_info );
1417:C        4         END; { of IF }
1418:S
```

```
1419:C        3       drv_tmp_ptr:= isc_table[io_isc].io_tmp_ptr;
1420:C        3       act_tfr    := no_tfr;
1421:C        3       usr_tfr    := t_tfr;
1422:C        3       b_w_mode   := FALSE;                    { byte mode }
1423:C        3       end_mode   := FALSE;                    { no EOI }
1424:C        3       direction  := t_dir;
1425:C        3       term_char  := -1;                       { no termination character }
1426:C        3       term_count := t_cnt;
1427:S
1428:C        3     END; { of WITH b_info DO }
1429:S
1430:C        2     transfer_setup := io_isc;
1431:S
1432:C        2   END; { of transfer_setup }
1433:S
1434:S
1435:S
1436:D        1   PROCEDURE transfer   ( device    : type_device;
1437:D        2                          t_tfr     : user_tfr_type;
1438:D        2                          t_dir     : dir_of_tfr ;
1439:D        2                          VAR b_info: buf_info_type ;
1440:D        2                          x_count   : INTEGER ) ;
1441:D    -2  2   VAR io_isc     : type_isc;
1442:D    -6  2       t_count   : INTEGER;
1443:C        2   BEGIN
1444:C        2     t_count:=x_count;
1445:C        2     io_isc:=transfer_setup(device,t_tfr,t_dir,b_info,t_count);
1446:S
1447:C        2     { transfer temporary was set up in transfer_setup }
1448:S
1449:C        2     WITH isc_table[io_isc]
1450:C        3     DO CALL ( io_drv_ptr^.iod_tfr ,
1451:C        3               isc_table[io_isc].io_tmp_ptr,
1452:C        3               ADDR(b_info) );
1453:S
1454:C        2   END; { of transfer }
1455:S
1456:S
1457:S
1458:S
1459:D        1   PROCEDURE transfer_word
1460:D        2                          ( device    : type_device;
1461:D        2                            t_tfr     : user_tfr_type;
1462:D        2                            t_dir     : dir_of_tfr ;
1463:D        2                            VAR b_info: buf_info_type ;
1464:D        2                            x_count   : INTEGER ) ;
1465:D    -2  2   VAR io_isc     : type_isc;
1466:D    -6  2       t_count   : INTEGER;
1467:C        2   BEGIN
1468:C        2     t_count:=x_count;
1469:C        2     io_isc:=transfer_setup(device,t_tfr,t_dir,b_info,t_count);
1470:S
1471:C        2     { fix up transfer temporary }
1472:C        2     b_info.b_w_mode   := TRUE;                    { word mode }
1473:S
1474:C        2     WITH isc_table[io_isc]
1475:C        3     DO CALL ( io_drv_ptr^.iod_tfr ,
1476:C        3               isc_table[io_isc].io_tmp_ptr,
1477:C        3               ADDR(b_info) );
1478:S
```



```
1479:C        2   END; { of transfer_word }
1480:S
1481:S
1482:S
1483:S
1484:D        1   PROCEDURE transfer_until
1485:D        2                          ( term      : CHAR ;
1486:D        2                            device    : type_device;
1487:D        2                            t_tfr     : user_tfr_type;
1488:D        2                            t_dir     : dir_of_tfr;
1489:D        2                            VAR b_info: buf_info_type ) ;
1490:D    -2  2   VAR io_isc     : type_isc;
1491:D    -6  2       t_count   : INTEGER;
1492:C        2   BEGIN
1493:C        2     t_count := buffer_space(b_info);
1494:C        2     io_isc:=transfer_setup(device,t_tfr,t_dir,b_info,t_count);
1495:C        2
1496:C        2     IF t_dir = from_memory
1497:C        3       THEN BEGIN
1498:C        3         { error }
1499:C        3         io_escape(ioe_bad_tfr,io_isc);
1500:C        3       END;
1501:S
1502:C        2     WITH b_info DO BEGIN
1503:S
1504:C        3       { fix up transfer temporary }
1505:C        3       term_char  := ord(term);                  { termination character }
1506:S
1507:C        3       { check for transfer type cases }
1508:C        3       IF t_tfr = serial_FASTEST  THEN usr_tfr := serial_FHS;
1509:C        3       IF t_tfr = overlap_FASTEST THEN usr_tfr := overlap_FHS;
1510:C        3       IF t_tfr = OVERLAP         THEN usr_tfr := overlap_INTR;
1511:C        3       IF ( t_tfr = serial_DMA  )  OR  ( t_tfr = overlap_DMA )
1512:C        4         THEN BEGIN
1513:C        4           { error }
1514:C        4           io_escape(ioe_bad_tfr,io_isc);
1515:C        4         END; { of IF }
1516:S
1517:C        3     END; { of WITH b_info DO BEGIN }
1518:S
1519:S
1520:C        2     WITH isc_table[io_isc]
1521:C        3     DO CALL ( io_drv_ptr^.iod_tfr ,
1522:C        3               isc_table[io_isc].io_tmp_ptr,
1523:C        3               ADDR(b_info) );
1524:S
1525:C        2   END; { of transfer_until }
1526:S
1527:S
1528:S
1529:D        1   PROCEDURE transfer_end(device    : type_device;
1530:D        2                          t_tfr     : user_tfr_type;
1531:D        2                          t_dir     : dir_of_tfr;
1532:D        2                          VAR b_info: buf_info_type ) ;
1533:D    -2  2   VAR io_isc     : type_isc;
1534:D    -6  2       t_count   : INTEGER;
1535:C        2   BEGIN
1536:C        2     IF t_dir=from_memory
1537:C        3       THEN BEGIN
1538:C        3         t_count := buffer_data(b_info);
```

```
1539:C        3       END
1540:C        3       ELSE BEGIN
1541:C        3         t_count := buffer_space(b_info);
1542:C        3       END; { of IF }
1543:C        2     io_isc:=transfer_setup(device,t_tfr,t_dir,b_info,t_count);
1544:S
1545:C        2     { fix up transfer temporary }
1546:C        2     b_info.end_mode   := TRUE;                        { EOI }
1547:S
1548:C        2     WITH isc_table[io_isc]
1549:C        3     DO CALL ( io_drv_ptr^.iod_tfr ,
1550:C        3               isc_table[io_isc].io_tmp_ptr,
1551:C        3               ADDR(b_info) );
1552:S
1553:C        2   END; { of transfer_end }
1554:S
1555:S
1556:S
1557:D        1   PROCEDURE readbuffer ( VAR b_info: buf_info_type;
1558:D        2                          VAR value : CHAR);
1559:D    -4  2   VAR p : ^CHAR;
1560:C        2   BEGIN
1561:C        2     IF buffer_data(b_info)<1
1562:C        3       THEN BEGIN
1563:C        3         { error }
1564:C        3         io_escape(ioe_no_data,no_isc);
1565:C        3       END
1566:C        3       ELSE BEGIN
1567:C        3         WITH b_info
1568:C        4         DO BEGIN
1569:C        4           IF ( active_isc <> no_isc ) AND
1570:C        5              ( direction = from_memory )
1571:C        5             THEN BEGIN
1572:C        5               { error }
1573:C        5               io_escape( no_isc , ioe_buf_busy );
1574:C        5             END; { of IF }
1575:C        4           p:=ANYPTR(buf_empty);
1576:C        4           value:=p^;
1577:C        4           buf_empty:=ANYPTR(INTEGER(buf_empty)+1);
1578:C        4         END; { of WITH b_info DO }
1579:C        3       END; { of IF }
1580:C        2   END; { of readbuffer }
1581:S
1582:S
1583:S
1584:D        1   PROCEDURE writebuffer( VAR b_info: buf_info_type;
1585:D        2                          value     : CHAR);
1586:D    -4  2   VAR p : ^CHAR;
1587:C        2   BEGIN
1588:C        2     IF buffer_space(b_info)<1
1589:C        3       THEN BEGIN
1590:C        3         { error }
1591:C        3         io_escape(ioe_no_space,no_isc);
1592:C        3       END
1593:C        3       ELSE BEGIN
1594:C        3         WITH b_info
1595:C        4         DO BEGIN
1596:C        4           IF ( active_isc <> no_isc ) AND
1597:C        5              ( direction = to_memory )
1598:C        5             THEN BEGIN
```



```
1599:C        5               { error }
1600:C        5               io_escape( no_isc , ioe_buf_busy );
1601:C        5             END; { of IF }
1602:C        4           p:=buf_fill;
1603:C        4           p^:=value;
1604:C        4           buf_fill:=ANYPTR(INTEGER(buf_fill)+1);
1605:C        4         END; { of WITH b_info DO }
1606:C        3       END; { of IF }
1607:C        2   END; { of writebuffer }
1608:S
1609:S
1610:S
1611:D        1   PROCEDURE readbuffer_string
1612:D        2                          ( VAR b_info: buf_info_type;
1613:D        2                            VAR str   : STRING;
1614:D        2                            str_count : INTEGER);
1615:D    -4  2   VAR i         : INTEGER ;
1616:C        2   BEGIN
1617:C        2     IF STRMAX(str) < str_count
1618:C        3       THEN BEGIN
1619:C        3         { error - string too small }
1620:C        3         io_escape(ioe_misc,no_isc);
1621:C        3       END;
1622:C        2     SETSTRLEN(str,str_count);           { so I can put chars into empty string }
1623:C        2     IF buffer_data(b_info)<str_count
1624:C        3       THEN BEGIN
1625:C        3         { error - not enought data in buffer }
1626:C        3         io_escape(ioe_no_data,no_isc);
1627:C        3       END
1628:C        3       ELSE BEGIN
1629:C        3         FOR i:=1 TO str_count
1630:C        4           DO BEGIN
1631:C        4             readbuffer(b_info,str[i]);
1632:C        4           END; { of FOR BEGIN }
1633:C        3         SETSTRLEN(str,str_count);
1634:C        3       END; { of IF }
1635:C        2   END; { of readbuffer_string }
1636:S
1637:S
1638:S
1639:D  -256  1   PROCEDURE writebuffer_string
1640:D        2                         ( VAR b_info: buf_info_type;
1641:D  -256  2                           str       : io_STRING);
1642:D  -260  2   VAR i         : INTEGER;
1643:C        2   BEGIN
1644:C        2     IF buffer_space(b_info)<STRLEN(str)
1645:C        3       THEN BEGIN
1646:C        3         { error }
1647:C        3         io_escape(ioe_no_space,no_isc);
1648:C        3       END
1649:C        3       ELSE BEGIN
1650:C        3         FOR i:=1 TO STRLEN(str)
1651:C        4           DO BEGIN
1652:C        4             writebuffer(b_info,str[i]);
1653:C        4           END; { of FOR BEGIN }
1654:C        3       END; { of IF }
1655:C        2   END; { of writebuffer_string }
1656:S
1657:S
1658:D        1   FUNCTION  buffer_busy( VAR b_info: buf_info_type )          { 0083 TM 7/23/82 }
```

```
1659:D  2                       : BOOLEAN;                                  { 0083 TM 7/23/82 }
1660:C  2   BEGIN                                                           { 0083 TM 7/23/82 }
1661:C  2     WITH b_info DO BEGIN                                          { 0083 TM 7/23/82 }
1662:C  3       IF active_isc <> no_isc THEN buffer_busy := TRUE            { 0083 TM 7/23/82 }
1663:C  4                               ELSE buffer_busy := FALSE;          { 0083 TM 7/23/82 }
1664:C  3     END; { of WITH DO BEGIN }                                     { 0083 TM 7/23/82 }
1665:C  2   END; { of buffer_busy }                                         { 0083 TM 7/23/82 }
1666:S
1667:D  1   FUNCTION  isc_busy   ( isc       : type_isc )                  { 0083 TM 7/23/82 }
1668:D  2                       : BOOLEAN;                                  { 0083 TM 7/23/82 }
1669:C  2   BEGIN                                                           { 0083 TM 7/23/82 }
1670:C  2     WITH isc_table[isc].io_tmp_ptr^ DO BEGIN                      { 0083 TM 7/23/82 }
1671:C  3       IF ( in_bufptr <> NIL ) OR                                  { 0083 TM 7/23/82 }
1672:C  4          ( out_bufptr <> NIL ) THEN isc_busy := TRUE              { 0083 TM 7/23/82 }
1673:C  4                                ELSE isc_busy := FALSE;           { 0083 TM 7/23/82 }
1674:C  3     END; { of WITH DO BEGIN }                                     { 0083 TM 7/23/82 }
1675:C  2   END; { of isc_busy }                                            { 0083 TM 7/23/82 }
1676:S
1677:S
1678:S
1679:C  1 END;    { of general_4 }
```



```
1680:D  1 $PAGE$
1681:D  1 (*****************************************************************************)
1682:D  1 (*                                                                           *)
1683:D  1 (*                                                                           *)
1684:D  1 (*      HPIB GROUP                                                           *)
1685:D  1 (*                                                                           *)
1686:D  1 (*                                                                           *)
1687:D  1 (*****************************************************************************)
1688:S
1689:S
1690:S
1691:S
1692:D  1 MODULE hpib_0 ;
1693:S
1694:S                    {
1695:S                      date    07/17/81
1696:S                      update  09/17/81
1697:S
1698:S                      purpose This module contains the LEVEL 0 HPIB GROUP procedures.
1699:S
1700:D  1                   }
1701:S
1702:S
1703:D  1 IMPORT    iodeclarations ;
1704:S
1705:S
1706:D  1 EXPORT
1707:S
1708:S
1709:D  1   PROCEDURE set_hpib    ( select_code : type_isc ;
1710:D  2                            line        : type_hpib_line);
1711:D  1   PROCEDURE clear_hpib  ( select_code : type_isc ;
1712:D  2                            line        : type_hpib_line);
1713:D  1   FUNCTION  hpib_line   ( select_code : type_isc ;
1714:D  2                            line        : type_hpib_line)
1715:D  2                          : BOOLEAN;
1716:S
1717:D  1 IMPLEMENT
1718:S
1719:S
1720:S
1721:S
1722:D  1   PROCEDURE set_hpib    ( select_code : type_isc ;
1723:D  2                            line        : type_hpib_line);
1724:C  2   BEGIN
1725:C  2     WITH isc_table[select_code] DO
1726:C  3     CALL(io_drv_ptr^.iod_set,
1727:C  3          io_tmp_ptr,
1728:C  3          ORD(line));
1729:C  2   END;
1730:S
1731:S
1732:S
1733:D  1   PROCEDURE clear_hpib  ( select_code : type_isc ;
1734:D  2                            line        : type_hpib_line);
1735:C  2   BEGIN
1736:C  2     WITH isc_table[select_code] DO
1737:C  3     CALL(io_drv_ptr^.iod_clr,
1738:C  3          io_tmp_ptr,
1739:C  3          ORD(line));
```

```
1740:C        2   END;
1741:S
1742:S
1743:S
1744:D        1   FUNCTION  hpib_line   ( select_code : type_isc ;
1745:D        2                           line        : type_hpib_line)
1746:D        2                         : BOOLEAN;
1747:D    -1  2   VAR my_boolean : BOOLEAN;
1748:C        2   BEGIN
1749:C        2     WITH isc_table[select_code] DO
1750:C        3     CALL(io_drv_ptr^.iod_test,
1751:C        3          io_tmp_ptr,
1752:C        3          ORD(line),
1753:C        3          my_boolean);
1754:C        2     hpib_line:=my_boolean;
1755:C        2   END;
1756:S
1757:S
1758:S
1759:C        1 END;    { of hpib_0 }
```



```
1760:D        1 $PAGE$
1761:D        1 MODULE hpib_2 ;
1762:S
1763:S                  {
1764:S                    date    07/17/81
1765:S                    update  03/09/83
1766:S
1767:S                    purpose This module contains the LEVEL
1768:S                            2 HPIB GROUP procedures.
1769:S
1770:D        1           }
1771:S
1772:S
1773:D        1 IMPORT  iodeclarations ;
1774:S
1775:D        1 EXPORT
1776:S
1777:S
1778:D        1   PROCEDURE abort_hpib
1779:D        2                   ( select_code : type_isc);
1780:D        1   PROCEDURE clear   ( device      : type_device);
1781:D        1   PROCEDURE listen  ( select_code : type_isc ;
1782:D        2                     address     : type_hpib_addr );
1783:D        1   PROCEDURE local   ( device      : type_device);
1784:D        1   PROCEDURE local_lockout
1785:D        2                   ( select_code : type_isc);
1786:D        1   PROCEDURE pass_control
1787:D        2                   ( device      : type_device);
1788:D        1   PROCEDURE ppoll_configure
1789:D        2                   ( device      : type_device;
1790:D        2                     mask        : INTEGER );
1791:D        1   PROCEDURE ppoll_unconfigure
1792:D        2                   ( device      : type_device),
1793:D        1   PROCEDURE remote  ( device      : type_device);
1794:D        1   PROCEDURE secondary
1795:D        2                   ( select_code : type_isc ;
1796:D        2                     address     : type_hpib_addr );
1797:D        1   PROCEDURE talk    ( select_code : type_isc ;
1798:D        2                     address     : type_hpib_addr );
1799:D        1   PROCEDURE trigger ( device      : type_device);
1800:D        1   PROCEDURE unlisten( select_code : type_isc );
1801:D        1   PROCEDURE untalk  ( select_code : type_isc );
1802:S
1803:S
1804:D        1 IMPLEMENT
1805:S
1806:D        1   IMPORT  hpib_0 ,
1807:D        1           hpib_1 ;
1808:S
1809:S
1810:D        1   PROCEDURE abort_hpib
1811:D        2                   ( select_code : type_isc);
1812:C        2   BEGIN
1813:S
1814:C        2     { what about active tfrs ? }
1815:S
1816:C        2     IF system_controller(select_code)
1817:C        3       THEN BEGIN
1818:C        3         set_hpib(select_code,ifc_line);
1819:C        3           set_hpib(select_code,ren_line);
```

```
1820:C        3            clear_hpib(select_code,ifc_line);
1821:C        3            clear_hpib(select_code,atn_line);            {  all done by ifc }
1822:C        3          END
1823:C        3        ELSE BEGIN
1824:C        3          IF active_controller(select_code)
1825:C        4            THEN BEGIN
1826:C        4              send_command(select_code,
1827:C        4                   CHR(talk_constant+my_address(select_code)));
1828:C        4              send_command(select_code,'?');
1829:C        4              clear_hpib(select_code,atn_line);
1830:C        4            END
1831:C        4            ELSE BEGIN
1832:C        4              { do nothing }
1833:C        4            END; { of IF }
1834:C        3        END; { of IF }
1835:C        2    END; { of abort_hpib }
1836:S
1837:S
1838:D        1    PROCEDURE clear   ( device      : type_device);
1839:D    -2  2    VAR io_isc      : type_isc;
1840:C        2    BEGIN
1841:C        2      io_isc:=set_to_listen(device);
1842:C        2      IF device>iomaxisc
1843:C        3        THEN BEGIN
1844:C        3          send_command(io_isc,sdc_message);
1845:C        3        END
1846:C        3        ELSE BEGIN
1847:C        3          send_command(io_isc,dcl_message);
1848:C        3        END; { of IF }
1849:C        2    END; { of clear }
1850:S
1851:S
1852:S
1853:D        1    PROCEDURE listen  ( select_code : type_isc ;
1854:D        2                        address     : type_hpib_addr );
1855:C        2    BEGIN
1856:C        2      send_command(select_code,CHR(listen_constant+address));
1857:C        2    END; { of listen }
1858:S
1859:S
1860:S
1861:D        1    PROCEDURE local   ( device      : type_device);
1862:D    -2  2    VAR io_isc      : type_isc;
1863:C        2    BEGIN
1864:C        2      IF device>iomaxisc
1865:C        3        THEN BEGIN
1866:C        3          io_isc:=set_to_listen(device);                   { BUG 1251  TM 1/8/82 }
1867:C        3          send_command(io_isc,gtl_message);
1868:C        3        END
1869:C        3        ELSE BEGIN
1870:C        3          io_isc := device;                                { BUG 1251  TM 1/8/82 }
1871:C        3          IF system_controller(io_isc)                     { BUG jsjs  TM 3/9/83 }
1872:C        4            THEN BEGIN
1873:C        4              { system controller - drop REN }             { BUG jsjs  TM 3/9/83 }
1874:C        4              clear_hpib(io_isc,ren_line);
1875:C        4              IF active_controller(io_isc)                 { BUG 1251  TM 1/26/82 }
1876:C        5                THEN clear_hpib(io_isc,atn_line);          { BUG 1251  TM 1/26/82 }
1877:C        4            END                                            { BUG jsjs  TM 3/9/83 }
1878:C        4            ELSE BEGIN                                     { BUG jsjs  TM 3/9/83 }
1879:C        4              { not system controller - send GTL }         { BUG jsjs  TM 3/9/83 }
```



```
1880:C        4              send_command(io_isc,gtl_message);            { BUG jsjs  TM 3/9/83 }
1881:C        4            END; { of IF }                                 { BUG jsjs  TM 3/9/83 }
1882:C        3        END; { of IF }
1883:C        2    END; { of local }
1884:S
1885:S
1886:S
1887:S
1888:D        1    PROCEDURE local_lockout
1889:D        2                ( select_code : type_isc);
1890:C        2    BEGIN
1891:C        2      send_command(select_code,llo_message);
1892:C        2    END; { of local_lockout }
1893:S
1894:S
1895:S
1896:D        1    PROCEDURE pass_control
1897:D        2                ( device      : type_device);
1898:D    -2  2    VAR io_isc      : type_isc;
1899:C        2    BEGIN
1900:C        2      IF device>iomaxisc                                   { BUG 1258  TM 1/8/82 }
1901:C        3        THEN BEGIN                                         { BUG 1258  TM 1/8/82 }
1902:C        3          io_isc := device DIV 100;                        { BUG 1258  TM 1/8/82 }
1903:C        3          send_command(io_isc,unl_message);                { BUG 1258  TM 1/8/82 }
1904:C        3          send_command(io_isc,                             { BUG 1258  TM 1/8/82 }
1905:C        3                  chr((device MOD 100)+talk_constant));    { BUG 1258  TM 1/8/82 }
1906:C        3        END                                                { BUG 1258  TM 1/8/82 }
1907:C        3        ELSE BEGIN                                         { BUG 1258  TM 1/8/82 }
1908:C        3          io_isc := set_to_talk(device);                   { BUG 1258  TM 1/8/82 }
1909:C        3        END; { of IF device }                              { BUG 1258  TM 1/8/82 }
1910:C        2      send_command(io_isc,tct_message);
1911:C        2    END;
1912:S
1913:S
1914:S
1915:D        1    PROCEDURE ppoll_configure
1916:D        2                ( device      : type_device;
1917:D        2                  mask        : INTEGER );
1918:D    -2  2    VAR io_isc      : type_isc;
1919:C        2    BEGIN
1920:C        2      io_isc:=set_to_listen(device);
1921:C        2      IF io_isc=device
1922:C        3        THEN BEGIN
1923:C        3          { error }
1924:C        3          io_escape(ioe_not_dvc,io_isc);
1925:C        3        END
1926:C        3        ELSE BEGIN
1927:C        3          send_command(io_isc,ppc_message);
1928:C        3          send_command(io_isc,CHR(ord(ppe_message)+(mask MOD 16)));
1929:C        3        END; { of IF }
1930:C        2    END; { of ppoll_configure }
1931:S
1932:S
1933:S
1934:D        1    PROCEDURE ppoll_unconfigure
1935:D        2                ( device      : type_device);
1936:D    -2  2    VAR io_isc      : type_isc;
1937:C        2    BEGIN
1938:C        2      io_isc:=set_to_listen(device);
1939:C        2      IF device>iomaxisc
```

```
1940:C        3       THEN BEGIN
1941:C        3         send_command(io_isc,ppc_message);
1942:C        3         send_command(io_isc,ppd_message);
1943:C        3       END
1944:C        3       ELSE BEGIN
1945:C        3         send_command(io_isc,ppu_message);
1946:C        3       END; { of IF }
1947:C        2   END; { of ppoll_unconfigure }
1948:S
1949:S
1950:S
1951:D        1   PROCEDURE remote  ( device      : type_device);
1952:D    -2  2   VAR io_isc      : type_isc;
1953:C        2   BEGIN
1954:C        2     IF device>iomaxisc
1955:C        3       THEN BEGIN
1956:C        3         io_isc:=device DIV 100;
1957:C        3         IF NOT system_controller(io_isc)                { BUG 1252  TM 1/8/82 }
1958:C        4           THEN io_escape(ioe_not_sctl,io_isc);          { BUG 1252  TM 1/8/82 }
1959:C        3         set_hpib(io_isc,ren_line);
1960:C        3         io_isc:=set_to_listen(device);
1961:C        3       END
1962:C        3       ELSE BEGIN
1963:C        3         io_isc := device;                               { BUG 1252  TM 1/8/82 }
1964:C        3         IF NOT system_controller(io_isc)                { BUG 1252  TM 1/8/82 }
1965:C        4           THEN io_escape(ioe_not_sctl,io_isc);          { BUG 1252  TM 1/8/82 }
1966:C        3         set_hpib(io_isc,ren_line);
1967:C        3       END; { of IF }
1968:C        2   END; { of remote }
1969:S
1970:S
1971:S
1972:D        1   PROCEDURE secondary
1973:D        2                   ( select_code : type_isc ;
1974:D        2                     address     : type_hpib_addr );
1975:C        2   BEGIN
1976:C        2     send_command(select_code,CHR(address+96));
1977:C        2   END; { of secondary }
1978:S
1979:S
1980:S
1981:D        1   PROCEDURE talk    ( select_code : type_isc ;
1982:D        2                     address     : type_hpib_addr );
1983:C        2   BEGIN
1984:C        2     send_command(select_code,CHR(address+talk_constant));
1985:C        2   END; { of talk }
1986:S
1987:S
1988:S
1989:D        1   PROCEDURE trigger ( device      : type_device);
1990:C        2   BEGIN
1991:C        2     send_command(set_to_listen(device),get_message);
1992:C        2   END; { of trigger }
1993:S
1994:S
1995:D        1   PROCEDURE unlisten( select_code : type_isc );
1996:C        2   BEGIN
1997:C        2     send_command(select_code,unl_message);
1998:C        2   END; { of unlisten }
1999:S
```



```
2000:S
2001:D        1   PROCEDURE untalk  ( select_code : type_isc );
2002:C        2   BEGIN
2003:C        2     send_command(select_code,unt_message);
2004:C        2   END; { of untalk }
2005:S
2006:S
2007:S
2008:S
2009:C        1 END;    { of hpib_2 }
```

```
2010:D        1 $PAGE$
2011:D        1 MODULE hpib_3 ;
2012:S
2013:S                  {
2014:S                    date    07/17/81
2015:S                    update  01/08/82
2016:S
2017:S                    purpose This module contains the LEVEL
2018:S                            3 HPIB GROUP procedures.
2019:S
2020:D        1           }
2021:S
2022:S
2023:D        1 IMPORT    iodeclarations ;
2024:S
2025:D        1 EXPORT
2026:S
2027:S
2028:D        1   FUNCTION  requested
2029:D        2                       ( select_code : type_isc )
2030:D        2                       : BOOLEAN ;
2031:D        1   FUNCTION  ppoll     ( select_code : type_isc )
2032:D        2                       : INTEGER ;
2033:D        1   FUNCTION  spoll     ( device      : type_device)
2034:D        2                       : INTEGER ;
2035:S
2036:D        1   PROCEDURE request_service
2037:D        2                       ( select_code : type_isc ;
2038:D        2                         response    : INTEGER );
2039:D        1   FUNCTION  listener( select_code : type_isc )
2040:D        2                       : BOOLEAN;
2041:D        1   FUNCTION  talker   ( select_code : type_isc )
2042:D        2                       : BOOLEAN ;
2043:D        1   FUNCTION  remoted  ( select_code : type_isc )
2044:D        2                       : BOOLEAN ;
2045:D        1   FUNCTION  locked_out
2046:D        2                       ( select_code : type_isc )
2047:D        2                       : BOOLEAN ;
2048:S
2049:S
2050:D        1 IMPLEMENT
2051:S
2052:S
2053:D        1   IMPORT  iocomasm ,
2054:D        1           general_0 ,
2055:D        1           general_1 ,
2056:D        1           hpib_0 ,
2057:D        1           hpib_1 ;
2058:S
2059:S
2060:S
2061:S
2062:D        1   FUNCTION  requested
2063:D        2                       ( select_code : type_isc )
2064:D        2                       : BOOLEAN ;
2065:C        2   BEGIN
2066:C        2     IF active_controller(select_code)
2067:C        3       THEN BEGIN
2068:C        3         requested:=hpib_line(select_code,srq_line);
2069:C        3       END
```



```
2070:C        3       ELSE BEGIN
2071:C        3         { error - not active controller when look at srq }
2072:C        3         io_escape(ioe_not_act,select_code);
2073:C        3       END; { of IF }
2074:C        2   END; { of requested }
2075:S
2076:S
2077:S
2078:S
2079:S
2080:D        1   FUNCTION  ppoll   ( select_code : type_isc )
2081:D        2                     : INTEGER ;
2082:D    -1  2   VAR my_byte : CHAR;
2083:C        2   BEGIN
2084:C        2     WITH isc_table[select_code] DO
2085:C        3     CALL(io_drv_ptr^.iod_ppoll,
2086:C        3          io_tmp_ptr,
2087:C        3          my_byte);
2088:C        2     ppoll:=ORD(my_byte);
2089:C        2   END; { of ppoll }
2090:S
2091:S
2092:S
2093:S
2094:D        1   FUNCTION  spoll   ( device      : type_device)
2095:D        2                     : INTEGER ;
2096:D    -2  2   VAR io_isc      : type_isc;
2097:D    -3  2       io_work_char: CHAR;
2098:C        2   BEGIN
2099:C        2     io_isc:=set_to_talk(device);
2100:C        2     send_command(io_isc,spe_message);
2101:C        2     readchar(io_isc,io_work_char);
2102:C        2     send_command(io_isc,spd_message);
2103:C        2     send_command(io_isc,unt_message);
2104:C        2     spoll:=ord(io_work_char);
2105:C        2   END; { of spoll }
2106:S
2107:S
2108:S
2109:S
2110:S
2111:S
2112:S
2113:S
2114:S
2115:S
2116:D        1   PROCEDURE request_service
2117:D        2                       ( select_code : type_isc ;
2118:D        2                         response    : INTEGER );
2119:C        2   BEGIN
2120:C        2     IF isc_table[select_code].card_type=hpib_card
2121:C        3       THEN BEGIN
2122:C        3         IF NOT active_controller(select_code)          { BUG 1250  TM 1/8/82 }
2123:C        4           THEN iocontrol(select_code,1,response)       { BUG 1250  TM 1/8/82 }
2124:C        4           ELSE io_escape(ioe_misc,select_code);        { BUG 1250  TM 1/8/82 }
2125:C        3       END
2126:C        3       ELSE BEGIN
2127:C        3         { error }
2128:C        3         io_escape(ioe_not_hpib,select_code);
2129:C        3       END; { of IF }
```

```
2130:C        2   END; { of request_service }
2131:S
2132:S
2133:S
2134:S
2135:D        1   FUNCTION  listener( select_code : type_isc )
2136:D        2                 : BOOLEAN;
2137:C        2   BEGIN
2138:C        2     listener:=bit_set(iostatus(select_code,6),10);
2139:C        2   END; { of listener }
2140:S
2141:S
2142:S
2143:S
2144:S
2145:D        1   FUNCTION  talker  ( select_code : type_isc )
2146:D        2                 : BOOLEAN ;
2147:C        2   BEGIN
2148:C        2     talker:=bit_set(iostatus(select_code,6),9);
2149:C        2   END; { of talker }
2150:S
2151:S
2152:S
2153:S
2154:D        1   FUNCTION  remoted ( select_code : type_isc )
2155:D        2                 : BOOLEAN ;
2156:C        2   BEGIN
2157:C        2     remoted:=bit_set(iostatus(select_code,6),15);
2158:C        2   END; { of remoted }
2159:S
2160:S
2161:S
2162:S
2163:D        1   FUNCTION  locked_out
2164:D        2                 ( select_code : type_isc )
2165:D        2                 : BOOLEAN ;
2166:C        2   BEGIN
2167:C        2     locked_out:=bit_set(iostatus(select_code,6),14);
2168:C        2   END;
2169:S
2170:S
2171:S
2172:S
2173:C        1 END;    { of hpib_3 }
```



```
2174:D        1 $PAGE$
2175:D        1 (*****************************************************************************)
2176:D        1 (*                                                                           *)
2177:D        1 (*                                                                           *)
2178:D        1 (*      SERIAL GROUP                                                         *)
2179:D        1 (*                                                                           *)
2180:D        1 (*                                                                           *)
2181:D        1 (*****************************************************************************)
2182:D        1 (*                                                                           *)
2183:D        1 (*                                                                           *)
2184:D        1 (*      The 98626 code in the serial_0 and serial_3 modules has NOT          *)
2185:D        1 (*      been tested and is included in the hopes that it is correct          *)
2186:D        1 (*      and that someone will do the 98626 card drivers sometime.            *)
2187:D        1 (*                                                                           *)
2188:D        1 (*      There is a good chance that the 98626 will require a re-             *)
2189:D        1 (*      release of the IOLIB:IOLIB file ( serial modules only ).             *)
2190:D        1 (*                                                                           *)
2191:D        1 (*****************************************************************************)
2192:S
2193:S
2194:S
2195:S
2196:D        1 MODULE serial_0 ;
2197:S
2198:S                  (
2199:S                    date    07/22/81
2200:S                    update  11/06/81
2201:S
2202:S                    purpose This module contains the LEVEL
2203:S                            0 SERIAL GROUP procedures.
2204:S
2205:D        1           )
2206:S
2207:S
2208:D        1 IMPORT    iodeclarations ;
2209:S
2210:D        1 EXPORT
2211:S
2212:S
2213:S
2214:S
2215:D        1   PROCEDURE set_serial  ( select_code : type_isc ;
2216:D        2                           line        : type_serial_line);
2217:D        1   PROCEDURE clear_serial( select_code : type_isc ;
2218:D        2                           line        : type_serial_line);
2219:D        1   FUNCTION  serial_line ( select_code : type_isc ;
2220:D        2                           line        : type_serial_line)
2221:D        2                         : BOOLEAN;
2222:S
2223:S
2224:D        1 IMPLEMENT
2225:S
2226:D        1   IMPORT  iocomasm ,
2227:D        1           general_0 ;
2228:S
2229:S
2230:S
2231:D        1   PROCEDURE set_serial  ( select_code : type_isc ;
2232:D        2                           line        : type_serial_line);
2233:D    -4  2   VAR mybit  : INTEGER;
```

```
2234:D    -8  2       dummy  : INTEGER;
2235:C        2   BEGIN
2236:C        2     WITH isc_table[select_code] DO BEGIN
2237:S
2238:C        3       IF card_type <> serial_card THEN io_escape(ioe_bad_sct,select_code);
2239:C        3
2240:C        3       IF (isc_table[select_code].card_id = hp98628_async)
2241:C        4         THEN BEGIN
2242:C        4           CASE line OF
2243:S
2244:C        5             rts_line:   mybit := 1;
2245:S
2246:C        5             dtr_line:   mybit := 2;
2247:C        5
2248:C        5             drs_line:   mybit := 4;
2249:S
2250:C        5             OTHERWISE io_escape(ioe_bad_sct,select_code);
2251:S
2252:C        5           END; { of CASE line }
2253:C        4           dummy := iostatus(select_code,8);
2254:C        4           dummy := binior(dummy,mybit);
2255:C        4           iocontrol(select_code,8+256,dummy);
2256:S
2257:C        4         END
2258:C        4         ELSE BEGIN
2259:C        4
2260:C        4           IF (card_id = hp98626) OR (card_id = hp98644)
2261:C        5             THEN BEGIN
2262:C        5               CASE line OF
2263:S
2264:C        6                 rts_line:   mybit := 2;
2265:S
2266:C        6                 dtr_line:   mybit := 1;
2267:S
2268:C        6                 drs_line:   mybit := 8;
2269:S
2270:C        6                 OTHERWISE io_escape(ioe_bad_sct,select_code);
2271:S
2272:C        6               END; { of CASE line }
2273:S
2274:C        5               dummy := iostatus(select_code,5);
2275:C        5               dummy := binior(dummy,mybit);
2276:C        5               iocontrol(select_code,5,dummy);
2277:S
2278:C        5             END
2279:C        5             ELSE BEGIN
2280:C        5               CALL ( io_drv_ptr^.iod_set ,
2281:C        5                      io_tmp_ptr ,
2282:C        5                      ORD(line) );
2283:C        5               END; { of IF }
2284:S
2285:C        4         END; { of IF }
2286:S
2287:C        3     END; { of WITH isc_table BEGIN }
2288:S
2289:C        2   END; { of set_serial }
2290:S
2291:S
2292:S
2293:S
```



```
2294:D        1   PROCEDURE clear_serial( select_code : type_isc ;
2295:D        2                            line        : type_serial_line);
2296:D    -4  2   VAR mybit  : INTEGER;
2297:D    -8  2       dummy  : INTEGER;
2298:C        2   BEGIN
2299:C        2     WITH isc_table[select_code] DO BEGIN
2300:S
2301:C        3       IF card_type <> serial_card THEN io_escape(ioe_bad_sct,select_code);
2302:C        3
2303:C        3       IF isc_table[select_code].card_id = hp98628_async
2304:C        4         THEN BEGIN
2305:C        4           CASE line OF
2306:S
2307:C        5             rts_line:   mybit := 1;
2308:S
2309:C        5             dtr_line:   mybit := 2;
2310:C        5
2311:C        5             drs_line:   mybit := 4;
2312:S
2313:C        5             OTHERWISE io_escape(ioe_bad_sct,select_code);
2314:S
2315:C        5           END; { of CASE line }
2316:C        4           dummy := iostatus(select_code,8);
2317:C        4           dummy := binand(dummy,bincmp(mybit));
2318:C        4           iocontrol(select_code,8+256,dummy);
2319:S
2320:C        4         END
2321:C        4         ELSE BEGIN
2322:C        4
2323:C        4           IF (card_id = hp98626) or (card_id = hp98644)
2324:C        5             THEN BEGIN
2325:C        5               CASE line OF
2326:S
2327:C        6                 rts_line:   mybit := 2;
2328:S
2329:C        6                 dtr_line:   mybit := 1;
2330:S
2331:C        6                 drs_line:   mybit := 8;
2332:S
2333:C        6                 OTHERWISE io_escape(ioe_bad_sct,select_code);
2334:S
2335:C        6               END; { of CASE line }
2336:S
2337:C        5               dummy := iostatus(select_code,5);
2338:C        5               dummy := binand(dummy,bincmp(mybit));
2339:C        5               iocontrol(select_code,5,dummy);
2340:S
2341:C        5             END
2342:C        5             ELSE BEGIN
2343:C        5               CALL ( io_drv_ptr^.iod_clr ,
2344:C        5                      io_tmp_ptr ,
2345:C        5                      ORD(line) );
2346:C        5               END; { of IF }
2347:S
2348:C        4         END; { of IF }
2349:S
2350:C        3     END; { of WITH isc_table BEGIN }
2351:C        2   END; { of clear_serial }
2352:S
2353:S
```

```
2354:S
2355:S
2356:S
2357:D        1   FUNCTION  serial_line ( select_code : type_isc ;
2358:D        2                            line        : type_serial_line )
2359:D        2                          : BOOLEAN ;
2360:D    -4  2   VAR mybit  : INTEGER;
2361:D    -8  2       dummy  : INTEGER;
2362:D   -12  2       reg    : INTEGER;
2363:D   -13  2       mybool : BOOLEAN;
2364:C        2   BEGIN
2365:C        2     WITH isc_table[select_code] DO BEGIN
2366:S
2367:C        3       IF card_type <> serial_card THEN io_escape(ioe_bad_sct,select_code);
2368:C        3
2369:C        3       IF isc_table[select_code].card_id = hp98628_async
2370:C        4         THEN BEGIN
2371:C        4           CASE line OF
2372:S
2373:C        5             rts_line:   BEGIN
2374:C        5                           reg   := 8;
2375:C        5                           mybit := 0;
2376:C        5                         END;
2377:S
2378:C        5             dtr_line:   BEGIN
2379:C        5                           reg   := 8;
2380:C        5                           mybit := 1;
2381:C        5                         END;
2382:C        5
2383:C        5             drs_line:   BEGIN
2384:C        5                           reg   := 8;
2385:C        5                           mybit := 2;
2386:C        5                         END;
2387:S
2388:C        5             dsr_line:   BEGIN
2389:C        5                           reg   := 7;
2390:C        5                           mybit := 0;
2391:C        5                         END;
2392:S
2393:C        5             dcd_line:   BEGIN
2394:C        5                           reg   := 7;
2395:C        5                           mybit := 1;
2396:C        5                         END;
2397:S
2398:C        5             cts_line:   BEGIN
2399:C        5                           reg   := 7;
2400:C        5                           mybit := 2;
2401:C        5                         END;
2402:S
2403:C        5             ri_line:    BEGIN
2404:C        5                           reg   := 7;
2405:C        5                           mybit := 3;
2406:C        5                         END;
2407:S
2408:C        5             OTHERWISE io_escape(ioe_bad_sct,select_code);
2409:S
2410:C        5           END; { of CASE line }
2411:C        4           dummy := iostatus(select_code,reg);
2412:C        4           mybool:= bit_set(dummy,mybit);
2413:S
```



```
2414:C        4         END
2415:C        4         ELSE BEGIN
2416:C        4
2417:C        4           IF (card_id = hp98626) or (card_id = hp98644)
2418:C        5             THEN BEGIN
2419:C        5               CASE line OF
2420:S
2421:C        6                 rts_line:   BEGIN
2422:C        6                               reg   := 5;
2423:C        6                               mybit := 1;
2424:C        6                             END;
2425:S
2426:C        6                 dtr_line:   BEGIN
2427:C        6                               reg   := 5;
2428:C        6                               mybit := 0;
2429:C        6                             END;
2430:S
2431:C        6                 drs_line:   BEGIN
2432:C        6                               reg   := 5;
2433:C        6                               mybit := 3;
2434:C        6                             END;
2435:S
2436:C        6                 dsr_line:   BEGIN
2437:C        6                               reg   := 11;
2438:C        6                               mybit := 5;
2439:C        6                             END;
2440:S
2441:C        6                 dcd_line:   BEGIN
2442:C        6                               reg   := 11;
2443:C        6                               mybit := 7;
2444:C        6                             END;
2445:S
2446:C        6                 cts_line:   BEGIN
2447:C        6                               reg   := 11;
2448:C        6                               mybit := 4;
2449:C        6                             END;
2450:S
2451:C        6                 ri_line:    BEGIN
2452:C        6                               reg   := 11;
2453:C        6                               mybit := 6;
2454:C        6                             END;
2455:S
2456:C        6                 OTHERWISE io_escape(ioe_bad_sct,select_code);
2457:S
2458:C        6               END; { of CASE line }
2459:C        5               dummy := iostatus(select_code,reg);
2460:C        5               mybool:= bit_set(dummy,mybit);
2461:S
2462:C        5             END
2463:C        5             ELSE BEGIN
2464:S
2465:C        5               CALL ( io_drv_ptr^.iod_test ,
2466:C        5                  io_tmp_ptr ,
2467:C        5                  ORD(line) ,
2468:C        5                  mybool );
2469:S
2470:C        5             END; { of IF }
2471:S
2472:C        4         END; { of IF }
2473:S
```

```
2474:C        3     END; { of WITH isc_table BEGIN }
2475:C        2
2476:C        2     serial_line := mybool;
2477:S
2478:C        2   END; { of serial_line }
2479:S
2480:S
2481:S
2482:S
2483:S
2484:C        1 END; { of serial_0 }
```



```
2485:D        1 $PAGE$
2486:D        1 MODULE serial_3 ;
2487:S
2488:S                  {
2489:S                    date    07/22/81
2490:S                    update  10/01/82
2491:S
2492:S                    purpose This module contains the LEVEL
2493:S                            3 SERIAL GROUP procedures.
2494:S
2495:D        1           }
2496:S
2497:D        1 IMPORT    iodeclarations ;
2498:S
2499:D        1 EXPORT
2500:S
2501:S
2502:D        1   PROCEDURE set_baud_rate
2503:D        2                   ( select_code : type_isc ;
2504:D        2                     rate        : REAL );
2505:D        1   PROCEDURE set_stop_bits
2506:D        2                   ( select_code : type_isc ;
2507:D        2                     num_bits    : REAL );
2508:D        1   PROCEDURE set_char_length
2509:D        2                   ( select_code : type_isc ;
2510:D        2                     num_char_bit: INTEGER );
2511:D        1   PROCEDURE set_parity
2512:D        2                   ( select_code : type_isc ;
2513:D        2                     parity_mode : type_parity);
2514:D        1   PROCEDURE send_break
2515:D        2                   ( select_code : type_isc );
2516:S
2517:D        1   PROCEDURE abort_serial
2518:D        2                   ( select_code : type_isc );
2519:S
2520:S
2521:D        1 IMPLEMENT
2522:S
2523:S
2524:D        1   IMPORT  iocomasm ,
2525:D        1           general_0 ;
2526:S
2527:S
2528:D    -8  1   PROCEDURE set_baud_rate
2529:D        2                   ( select_code : type_isc ;
2530:D    -8  2                     rate        : REAL );
2531:D   -12  2   VAR dummy : INTEGER;
2532:D    -8  2     FUNCTION calc_rate ( r : REAL ) : INTEGER;
2533:D   -12  3     VAR myrate : INTEGER;
2534:C        3     BEGIN
2535:C        3       myrate := 0;
2536:C        3       IF r=50    THEN myrate := 1;
2537:C        3       IF r=75    THEN myrate := 2;
2538:C        3       IF r=110   THEN myrate := 3;
2539:C        3       IF r=134.5 THEN myrate := 4;
2540:C        3       IF r=150   THEN myrate := 5;
2541:C        3       IF r=200   THEN myrate := 6;
2542:C        3       IF r=300   THEN myrate := 7;
2543:C        3       IF r=600   THEN myrate := 8;
2544:C        3       IF r=1200  THEN myrate := 9;
```

```
2545:C        3       IF r=1800  THEN myrate :=10;
2546:C        3       IF r=2400  THEN myrate :=11;
2547:C        3       IF r=3600  THEN myrate :=12;
2548:C        3       IF r=4800  THEN myrate :=13;
2549:C        3       IF r=9600  THEN myrate :=14;
2550:C        3       IF r=19200 THEN myrate :=15;
2551:S
2552:C        3       calc_rate := myrate;
2553:S
2554:C        3     END; { of calc_rate }
2555:C        2   BEGIN
2556:C        2     WITH isc_table[select_code] DO BEGIN
2557:S
2558:C        3       IF card_type <> serial_card THEN io_escape(ioe_misc,select_code);
2559:C        3
2560:C        3       IF isc_table[select_code].card_id = hp98628_async
2561:C        4         THEN BEGIN
2562:S
2563:C        4           dummy:=calc_rate(rate);
2564:C        4           IF dummy = 0 THEN io_escape(ioe_misc,select_code);
2565:C        4           iocontrol(select_code,20,dummy);              { BUG 1270  TM 1/8/82 }
2566:C        4           iocontrol(select_code,21,dummy);              { BUG 1270  TM 1/8/82 }
2567:S
2568:C        4         END
2569:C        4         ELSE BEGIN
2570:C        4
2571:C        4           IF (isc_table[select_code].card_id = hp98626) OR
2572:C        5              (isc_table[select_code].card_id = hp98644)
2573:C        5             THEN BEGIN
2574:S
2575:C        5               dummy:=ROUND(rate);
2576:C        5               IF dummy = 0 THEN io_escape(ioe_misc,select_code);
2577:C        5               iocontrol(select_code,3,dummy);
2578:S
2579:C        5               { what about 134.5 ? }
2580:S
2581:C        5             END
2582:C        5             ELSE BEGIN
2583:S
2584:C        5               io_escape(ioe_misc,select_code);
2585:S
2586:C        5             END; { of IF 98626 }
2587:S
2588:C        4         END; { of IF 98628_async }
2589:S
2590:C        3     END; { of WITH isc_table BEGIN }
2591:C        2   END; { of set_baud_rate }
2592:S
2593:S
2594:S
2595:S
2596:S
2597:D    -8  1   PROCEDURE set_stop_bits
2598:D        2                 ( select_code : type_isc ;
2599:D    -8  2                   num_bits    : REAL );
2600:D   -12  2   VAR myval : INTEGER;
2601:D   -16  2       dummy : INTEGER;
2602:C        2   BEGIN
2603:C        2     WITH isc_table[select_code] DO BEGIN
2604:S
```



```
2605:C        3       IF card_type <> serial_card THEN io_escape(ioe_misc,select_code);
2606:C        3
2607:C        3       IF isc_table[select_code].card_id = hp98628_async
2608:C        4         THEN BEGIN
2609:S
2610:C        4           IF num_bits = 1
2611:C        5             THEN BEGIN
2612:C        5               myval := 0;
2613:C        5             END
2614:C        5             ELSE BEGIN
2615:C        5               IF num_bits = 1.5
2616:C        6                 THEN BEGIN
2617:C        6                   myval := 1;
2618:C        6                 END
2619:C        6                 ELSE BEGIN
2620:C        6                   IF num_bits = 2
2621:C        7                     THEN BEGIN
2622:C        7                       myval :=2
2623:C        7                     END
2624:C        7                     ELSE BEGIN
2625:C        7                       io_escape(ioe_misc,select_code);
2626:C        7                     END; { of IF 2 }
2627:S
2628:C        6                 END;   { of IF 1.5 }
2629:S
2630:C        5             END;     { of IF 1 }
2631:S
2632:C        4           iocontrol(select_code,35,myval);              { BUG 1270  TM 1/8/82 }
2633:S
2634:C        4         END
2635:C        4         ELSE BEGIN
2636:C        4           IF (isc_table[select_code].card_id = hp98626)   { BUG 1269  TM 1/8/82 }
2637:C        5             OR (isc_table[select_code].card_id = hp98644)
2638:C        5             THEN BEGIN
2639:C        5               IF num_bits = 1
2640:C        6                 THEN BEGIN
2641:C        6                   myval:=0;
2642:C        6                 END
2643:C        6                 ELSE BEGIN
2644:C        6                   IF num_bits = 1.5
2645:C        7                     THEN BEGIN
2646:C        7                       IF binand(iostatus(select_code,4),3)<>0
2647:C        8                         THEN io_escape(ioe_misc,select_code);
2648:C        7                       myval:=1;
2649:C        7                     END
2650:C        7                     ELSE BEGIN
2651:C        7                       IF num_bits = 2
2652:C        8                         THEN BEGIN
2653:C        8                           myval:=1;
2654:C        8                         END
2655:C        8                         ELSE BEGIN
2656:C        8                           io_escape(ioe_misc,select_code);
2657:C        8                         END; { of IF 2 }
2658:S
2659:C        7                     END;   { of IF 1.5 }
2660:S
2661:C        6                 END;     { of IF 1 }
2662:S
2663:C        5               dummy:=iostatus(select_code,4);
2664:C        5               dummy:=binand(dummy,251)+myval*4;             { 0359 TM 8/26/82 }
```

```
Pascal [Rev 3.0M  6/ 4/84] IOLIB.TEXT             26-Dec-84 21:36:08 Page 55   *** File name: IOLIB ***

2665:C        5              iocontrol(select_code,4,dummy);
2666:S
2667:C        5            END
2668:C        5            ELSE BEGIN
2669:C        5              io_escape(ioe_misc,select_code);
2670:C        5            END; { of IF 98626 }
2671:S
2672:C        4          END; { of IF 98628_async }
2673:S
2674:C        3      END; { of WITH isc_table BEGIN }
2675:S
2676:C        2    END; { set_stop_bits }
2677:S
2678:S
2679:S
2680:S
2681:D        1    PROCEDURE set_char_length
2682:D        2                    ( select_code : type_isc ;
2683:D        2                      num_char_bit: INTEGER );
2684:D    -4  2    VAR myval : INTEGER;
2685:D    -8  2        dummy : INTEGER;
2686:C        2    BEGIN
2687:C        2      WITH isc_table[select_code] DO BEGIN
2688:S
2689:C        3        IF card_type <> serial_card THEN io_escape(ioe_misc,select_code);
2690:C        3
2691:C        3        CASE num_char_bit OF
2692:S
2693:C        4          5:    myval := 0;
2694:C        4          6:    myval := 1;
2695:C        4          7:    myval := 2;
2696:C        4          8:    myval := 3;
2697:S
2698:C        4          OTHERWISE io_escape(ioe_misc,select_code);
2699:S
2700:C        4        END; { of CASE }
2701:S
2702:S
2703:C        3        IF isc_table[select_code].card_id = hp98628_async
2704:C        4          THEN BEGIN
2705:C        4
2706:C        4            iocontrol(select_code,34,myval);
2707:S
2708:C        4          END
2709:C        4          ELSE BEGIN
2710:S
2711:C        4            IF(isc_table[select_code].card_id = hp98626)
2712:C        5              or (isc_table[select_code].card_id = hp98644)
2713:C        5              THEN BEGIN
2714:S
2715:C        5                dummy:=iostatus(select_code,4);
2716:C        5                dummy:=binand(dummy,252)+myval                    { 0359 TM 8/23/82 }
2717:C        5                iocontrol(select_code,4,dummy);                   {  557 TM 10/1/82 }
2718:S
2719:C        5              END
2720:C        5              ELSE BEGIN
2721:S
2722:C        5                io_escape(ioe_misc,select_code);
2723:S
2724:C        5              END; { of IF 98626 }
```

```
Pascal [Rev 3.0M  6/ 4/84] IOLIB.TEXT             26-Dec-84 21:36:08 Page 56   *** File name: IOLIB ***

2725:S
2726:C        4          END; { of IF 98628_asnync }
2727:S
2728:C        3      END; { of WITH isc_table BEGIN }
2729:C        2    END; { set_char_length }
2730:S
2731:S
2732:S
2733:S
2734:S
2735:D        1    PROCEDURE set_parity
2736:D        2                    ( select_code : type_isc ;
2737:D        2                      parity_mode : type_parity);
2738:D    -4  2    VAR myval : INTEGER;
2739:D    -8  2        dummy : INTEGER;
2740:C        2    BEGIN
2741:C        2      WITH isc_table[select_code] DO BEGIN
2742:S
2743:C        3        IF card_type <> serial_card THEN io_escape(ioe_misc,select_code);
2744:C        3
2745:C        3        IF isc_table[select_code].card_id = hp98628_async
2746:C        4          THEN BEGIN
2747:C        4
2748:C        4            CASE parity_mode OF
2749:S
2750:C        5              no_parity:    myval := 0;
2751:C        5              odd_parity:   myval := 1;
2752:C        5              even_parity:  myval := 2;
2753:C        5              zero_parity:  myval := 3;                         { 0355 TM 8/20/82 }
2754:C        5              one_parity:   myval := 4;                         { 0355 TM 8/20/82 }
2755:S
2756:C        5              OTHERWISE io_escape(ioe_misc,select_code);
2757:S
2758:C        5            END; { of CASE }
2759:S
2760:C        4            iocontrol(select_code,36,myval);
2761:S
2762:C        4          END
2763:C        4          ELSE BEGIN
2764:S
2765:C        4            IF (isc_table[select_code].card_id = hp98626)
2766:C        5              or (isc_table[select_code].card_id = hp98644)
2767:C        5              THEN BEGIN
2768:S
2769:C        5                CASE parity_mode OF
2770:S
2771:C        6                  no_parity:    myval := 0;
2772:C        6                  odd_parity:   myval := 1;
2773:C        6                  even_parity:  myval := 3;
2774:C        6                  one_parity:   myval := 5;
2775:C        6                  zero_parity:  myval := 7;
2776:S
2777:C        6                  OTHERWISE io_escape(ioe_misc,select_code);
2778:S
2779:C        6                END; { of CASE }
2780:S
2781:C        5                dummy:=iostatus(select_code,4);
2782:C        5                dummy:=binand(dummy,193)+myval*8;                 { 0359 TM 8/23/82 }
2783:C        5                iocontrol(select_code,4,dummy);
2784:S
```

```
2785:C        5             END
2786:C        5             ELSE BEGIN
2787:S
2788:C        5               io_escape(ioe_misc,select_code);
2789:S
2790:C        5             END; { of IF 98626 }
2791:S
2792:C        4         END; { of IF 98628_asnync }
2793:S
2794:C        3     END; { of WITH isc_table BEGIN }
2795:C        2   END; { set_parity }
2796:S
2797:S
2798:S
2799:S
2800:D        1   PROCEDURE send_break
2801:D        2                     ( select_code : type_isc );
2802:C        2   BEGIN
2803:C        2
2804:C        2     { what about active tfrs }
2805:S
2806:C        2     WITH isc_table[select_code] DO BEGIN
2807:S
2808:C        3       IF card_type <> serial_card THEN io_escape(ioe_misc,select_code);
2809:C        3
2810:C        3       IF isc_table[select_code].card_id = hp98628_async
2811:C        4         THEN BEGIN
2812:S
2813:C        4           iocontrol(select_code,6,1);
2814:S
2815:C        4         END
2816:C        4         ELSE BEGIN
2817:C        4           IF (card_id = hp98626) or (card_id = hp98644)
2818:C        5             THEN BEGIN
2819:S
2820:C        5               iocontrol(select_code,1,1);
2821:S
2822:C        5             END
2823:C        5             ELSE BEGIN
2824:S
2825:C        5               io_escape(ioe_misc,select_code);
2826:C        5
2827:C        5             END; { of IF }
2828:S
2829:C        4         END; { of IF }
2830:S
2831:C        3     END; { of WITH isc_table BEGIN }
2832:C        2   END; { of send_break }
2833:S
2834:S
2835:D        1   PROCEDURE abort_serial
2836:D        2                     ( select_code : type_isc );
2837:C        2   BEGIN
2838:S
2839:C        2     { what about active tfrs }
2840:S
2841:C        2     WITH isc_table[select_code] DO BEGIN
2842:S
2843:C        3       IF card_type <> serial_card THEN io_escape(ioe_misc,select_code);
2844:C        3
```



```
2845:C        3       IF isc_table[select_code].card_id = hp98628_async
2846:C        4         THEN BEGIN
2847:S
2848:C        4           iocontrol(select_code,256+125,1);            { BUG xxxx TM 1/26/82 }
2849:S
2850:C        4         END
2851:C        4         ELSE BEGIN
2852:C        4           IF (card_id = hp98626) or (card_id = hp98644)  { BUG FIX 6/4/84 }
2853:C        5             THEN BEGIN
2854:S
2855:C        5               iocontrol(select_code,0,1);
2856:S
2857:C        5             END
2858:C        5             ELSE BEGIN
2859:S
2860:C        5               io_escape(ioe_misc,select_code);
2861:C        5
2862:C        5             END; { of IF }
2863:S
2864:C        4         END; { of IF }
2865:S
2866:C        3     END; { of WITH isc_table BEGIN }
2867:C        2   END; { of abort_serial }
2868:S
2869:S
2870:S
2871:C        1 END.    { of serial_3 }
2872:S

No errors. 1 warnings.
            ***** Nonstandard language features enabled *****
```

Description

This is the I/O library kernel.

Usage

The modules in KERNEL provide the framework for the rest of the I/O library and I/O drivers.

```
     1:D        0
     2:S                                                    (*
     3:S
     4:S           (c) Copyright Hewlett-Packard Company, 1984.
     5:S         All rights are reserved.  Copying or other
     6:S         reproduction of this program except for archival
     7:S         purposes is prohibited without the prior
     8:S         written consent of Hewlett-Packard Company.
     9:S
    10:S
    11:S                        RESTRICTED RIGHTS LEGEND
    12:S
    13:S         Use, duplication, or disclosure by the Government
    14:S         is subject to restrictions as set forth in
    15:S         paragraph (b) (3) (B) of the Rights in Technical
    16:S         Data and Computer Software clause in
    17:S         DAR 7-104.9(a).
    18:S
    19:S         HEWLETT-PACKARD COMPANY
    20:D        0 Fort Collins, Colorado                        *)
    21:S
    22:S
    23:D        0 $MODCAL ON$
    24:D        0 $PARTIAL_EVAL ON$
    25:D        0 $RANGE OFF$
    26:D        0 $DEBUG OFF$
    27:D        0 $UVFLCHECK OFF$
    28:D        0 $STACKCHECK OFF$
```



```
    29:D        0 $PAGE$
    30:D        0 (*************************************************************************)
    31:D        0 (*                                                                       *)
    32:D        0 (*       RELEASED        VERSION         3.0                             *)
    33:D        0 (*                                                                       *)
    34:D        0 (*************************************************************************)
    35:D        0 (*                                                                       *)
    36:D        0 (*                                                                       *)
    37:D        0 (*       IOLIB           KERNEL                                          *)
    38:D        0 (*                                                                       *)
    39:D        0 (*                                                                       *)
    40:D        0 (*************************************************************************)
    41:D        0 (*                                                                       *)
    42:D        0 (*                                                                       *)
    43:D        0 (*       library       -  IOLIB                                          *)
    44:D        0 (*       name          -  KERNEL                                         *)
    45:D        0 (*       module(s)     -  iodeclarations                                 *)
    46:D        0 (*                     -  iocomasm                                       *)
    47:D        0 (*                     -  general_0                                      *)
    48:D        0 (*                                                                       *)
    49:D        0 (*       author        -                                                 *)
    50:D        0 (*       phone         -                                                 *)
    51:D        0 (*                                                                       *)
    52:D        0 (*       date          -  June  1 , 1981                                 *)
    53:D        0 (*       update        -  Feb   9 , 1984                                 *)
    54:D        0 (*       release       -  xxxxxxxxxxxxxx     (3.0 release date)          *)
    55:D        0 (*                                                                       *)
    56:D        0 (*       source        -  IOLIB:KERNEL.TEXT                              *)
    57:D        0 (*       object        -  IOLIB:KERNEL.CODE                              *)
    58:D        0 (*                                                                       *)
    59:D        0 (*************************************************************************)
```



```
    60:D        0 $PAGE$
    61:D        0 (*************************************************************************)
    62:D        0 (*                                                                       *)
    63:D        0 (*                                                                       *)
    64:D        0 (*       This is the source code for an external procedures library      *)
    65:D        0 (*       to be used for general purpose interfacing on the HP 9826.      *)
    66:D        0 (*                                                                       *)
    67:D        0 (*       The library consists of 3 primary sets of modules -             *)
    68:D        0 (*                                                                       *)
    69:D        0 (*               1.      KERNEL modules                                  *)
    70:D        0 (*               2.      driver modules                                  *)
    71:D        0 (*               3.      IOLIB  modules                                  *)
    72:D        0 (*                                                                       *)
    73:D        0 (*       The KERNEL modules consist of the following modules -           *)
    74:D        0 (*                                                                       *)
    75:D        0 (*               1.      iodeclarations  ( contains static r/w space )   *)
    76:D        0 (*               2.      iocomasm                                        *)
    77:D        0 (*               3.      general_0       ( initialization & low level    *)
    78:D        0 (*                                         routines like ioread/iowrite) *)
    79:D        0 (*       The KERNEL modules also have an executable program segement     *)
    80:D        0 (*       that gets executed at the time it is loaded.  This program      *)
    81:D        0 (*       initializes the static read/write memory.  This program also    *)
    82:D        0 (*       allocates the temporary storage for any card that exists -      *)
    83:D        0 (*       independent of whether there is or is not a driver for it.      *)
    84:D        0 (*                                                                       *)
    85:D        0 (*       The driver modules consist of the actual assembly or PASCAL     *)
    86:D        0 (*       routines that deal with a specific interface card.  There is    *)
    87:D        0 (*       also an executable program segment for each driver module.      *)
    88:D        0 (*       This program searches the select code table for the select      *)
```

```
    89:D        0 (*      initialized by the KERNEL general_0 module for all select codes *)
    90:D        0 (*      that have the right interface card ( HPIB drivers will search   *)
    91:D        0 (*      for the 98624 interface ).  This program will then set up the   *)
    92:D        0 (*      driver tables to point to the correct drivers.                  *)
    93:D        0 (*                                                                      *)
    94:D        0 (*      The rest of the IOLIB modules are high-level modules that are   *)
    95:D        0 (*      used by an end user in his/her application program.             *)
    96:D        0 (*                                                                      *)
    97:D        0 (*      The KERNEL and some set of driver modules will exist in the     *)
    98:D        0 (*      INITLIB file as object code ( not EXPORT text ).  The           *)
    99:D        0 (*      export text will reside on the IO file.  The rest               *)
   100:D        0 (*      of the library will reside on the IO file.                      *)
   101:D        0 (*                                                                      *)
   102:D        0 (************************************************************************)
```



```
   103:D        0 $PAGE$
   104:D        0 (************************************************************************)
   105:D        0 (*                                                                      *)
   106:D        0 (*                                                                      *)
   107:D        0 (*      BUG FIX HISTORY         - after release 1.0                     *)
   108:D        0 (*                                                                      *)
   109:D        0 (*                                                                      *)
   110:D        0 (*      BUG #   BY  / ON        LOC             DESCRIPTION             *)
   111:D        0 (*      -----   ----------      --------------  ----------------------  *)
   112:D        0 (*                                                                      *)
   113:D        0 (*      1281    ----------      IODECLARATIONS  bad error message.      *)
   114:D        0 (*              01/08/82                        No code change in this  *)
   115:D        0 (*                                              module - code change in *)
   116:D        0 (*                                              GENERAL_3.              *)
   117:D        0 (*                                                                      *)
   118:D        0 (*      aaaa    ----------      kernel          No bug sheet.           *)
   119:D        0 (*              05/24/82        initialize      The CRT and keyboard    *)
   120:D        0 (*                                              select codes were       *)
   121:D        0 (*                                              interchanged.  Should   *)
   122:D        0 (*                                              have no effect.         *)
   123:D        0 (*                                                                      *)
   124:D        0 (*      bbbb    ----------      kernel          No bug sheet.           *)
   125:D        0 (*              07/09/82        initialize      Adding machine id info  *)
   126:D        0 (*                              iodeclarations  and the                 *)
   127:D        0 (*                                              Ganglia drivers.        *)
   128:D        0 (*                                                                      *)
   129:D        0 (*      jw      -------         kernel          No bug sheet.           *)
   130:D        0 (*              07/12/82        initialize      Adding temp type for    *)
   131:D        0 (*                              iodeclarations  626 drivers.            *)
   132:D        0 (*                                                                      *)
   133:D        0 (*      0082    ----------      kernel          Adding a link for the   *)
   134:D        0 (*              07/23/82        initialize      error message function. *)
   135:D        0 (*                              iodeclarations  See also GENERAL_3.     *)
   136:D        0 (*                              end_error_link                          *)
   137:D        0 (*                                                                      *)
   138:D        0 (*      cccc    ----------      iodeclarations  Allowing the 98629 card *)
   139:D        0 (*              08/16/82                        and the 98628_dsndl as  *)
   140:D        0 (*                                              identifiable cards.     *)
   141:D        0 (*                                              See DC_DRV modules also.*)
   142:D        0 (*                                                                      *)
   143:D        0 (*      0350    ----------      general_0       Fundamental flaw with   *)
   144:D        0 (*              08/19/82                        the dummy drivers.      *)
   145:D        0 (*                                                                      *)
   146:D        0 (*      0364    ----------      iodeclarations  Addition of SRM errors. *)
   147:D        0 (*              08/23/82                        See also GENERAL_3.     *)
   148:D        0 (*                                                                      *)
   149:D        0 (*      0367    ----------      iolibrary_kernel                        *)
   150:D        0 (*              09/22/82                        Allow kernel to install *)
   151:D        0 (*                                              itself - probably will  *)
   152:D        0 (*                                              not be used much.       *)
   153:D        0 (*                                                                      *)
   154:D        0 (*      tttt    ----------      kbd_rdb         Change in the handling  *)
   155:D        0 (*              09/22/82                        of the keyboard end of  *)
   156:D        0 (*                                              line condition from 1.0.*)
   157:D        0 (*                                                                      *)
   158:D        0 (*      uuuu    ----------      iodeclarations  Differentiate between a *)
   159:D        0 (*              09/28/82                        628 async and 629 srm   *)
   160:D        0 (*                                              card type (srm/serial). *)
   161:D        0 (*                                              See also DC_DRV.        *)
   162:D        0 (*                                                                      *)
```

```
   163:D        0 (*       jws     ---------       iodeclarations  Add card id's and types *)
   164:D        0 (*               03/25/83        general_0       for EPROM programmer    *)
   165:D        0 (*                                               and bubble cards.       *)
   166:D        0 (*                                               Add initialization code.*)
   167:D        0 (*                                                                       *)
   168:D        0 (*       jws2    ---------       general_0       Add test for internal   *)
   169:D        0 (*               06/28/83                         HP-IB present.         *)
   170:D        0 (*                                                                       *)
   171:D        0 (*       jws3    ---------       iodeclarations  Deleted io_model_number *)
   172:D        0 (*               02/09/84        general_0       and io_model_name stuff *)
   173:D        0 (*                                               Added GATOR id, init.   *)
   174:D        0 (*       jws4    ---------       iodeclarations  Added id and init for   *)
   175:D        0 (*               03/05/84        general_0       98644 card              *)
   176:D        0 (*****************************************************************************)
   177:S
```



```
   178:D        0 $PAGE$
   179:D        0 (*****************************************************************************)
   180:D        0 (*                                                                           *)
   181:D        0 (*                                                                           *)
   182:D        0 (*       REFERENCES :                                                        *)
   183:D        0 (*                                                                           *)
   184:D        0 (*                                                                           *)
   185:D        0 (*       1.  9826 I/O Designers Guide              ( ----------- )         *)
   186:D        0 (*                                                                           *)
   187:D        0 (*       2.  68000 Manual                          ( Motorola )            *)
   188:D        0 (*                                                                           *)
   189:D        0 (*       3.  Pascal alpha site ERS                 ( ---------- )          *)
   190:D        0 (*                                                                           *)
   191:D        0 (*       4.  Pascal I/O Library ERS                ( ------------ )        *)
   192:D        0 (*                                                                           *)
   193:D        0 (*       5.  9826 HPL EIO & IOD listings           ( ------------ )        *)
   194:D        0 (*                                                                           *)
   195:D        0 (*       6.  9826 HPL Misc. I/O Doc.               ( ------------ )        *)
   196:D        0 (*                                                                           *)
   197:D        0 (*       7.  9826 card documentation               ( ----------- )         *)
   198:D        0 (*                                                                           *)
   199:D        0 (*                                                                           *)
   200:D        0 (*****************************************************************************)
```

```
   201:D        0 $PAGE$
   202:D        0 PROGRAM iolibrary_kernel ( INPUT , OUTPUT );
   203:S
   204:S
   205:S
   206:S                  { the PROGRAM surrounds the following modules because there needs
   207:S                    to be a start address for this set of modules to allow
   208:D        1           initialization to occur }
   209:S
```



```
   210:D        1 $PAGE$
   211:D        1 (*****************************************************************************)
   212:D        1 (*                                                                           *)
   213:D        1 (*                                                                           *)
   214:D        1 (*      GENERAL GROUP           IODECLARATIONS                               *)
   215:D        1 (*                                                                           *)
   216:D        1 (*                                                                           *)
   217:D        1 (*****************************************************************************)
   218:S
   219:S
   220:S
   221:S
   222:D        1 MODULE iodeclarations ;
   223:S
   224:S                  {
   225:S                    date    07/15/81
   226:S                    update  03/05/84
   227:S
   228:S                    purpose This module contains the common
   229:S                            declarations to be used by the
   230:S                            rest of the I/O library.
   231:S
   232:D        1           }
   233:S
   234:S
   235:S
   236:D        1 IMPORT  sysglobals;
   237:S
   238:D        1 EXPORT
   239:D        1
   240:D        1   CONST
   241:D        1           iominisc   = 0 ;                { 0 - 7  internal }
   242:D        1           iomaxisc   = 31;                { 8 - 31 external 8..31 }
   243:D        1           minrealisc = 7 ;
   244:D        1           maxrealisc = 31;                { 7..31 are real isc.s with temps }
   245:S
   246:D        1           io_line_feed    = CHR(10);
   247:D        1           io_carriage_rtn = CHR(13);
   248:S
```

```
   249:D        1          $PAGE$
   250:S
   251:D        1          { escape code constants }
   252:S
   253:D        1          ioescapecode     =  -26;
   254:S
   255:D        1          ioe_no_error     = 0;            { no error                        }
   256:D        1          ioe_no_card      = 1;            { no card at select code          }
   257:D        1          ioe_not_hpib     = 2;            { interface should be hpib        }
   258:D        1          ioe_not_act      = 3;            { not active controller           }
   259:D        1          ioe_not_dvc      = 4;            { should be device not isc        } { BUG 1281 TM 1/8/82}
   260:D        1          ioe_no_space     = 5;            { no space left in buffer         }
   261:D        1          ioe_no_data      = 6;            { no data left in buffer          }
   262:D        1          ioe_bad_tfr      = 7;            { improper transfer attempted     }
   263:D        1          ioe_isc_busy     = 8;            { the select code is busy         }
   264:D        1          ioe_buf_busy     = 9;            { the buffer is busy              }
   265:D        1          ioe_bad_cnt      = 10;           { improper transfer count         }
   266:D        1          ioe_bad_tmo      = 11;           { bad timeout value               }
   267:D        1          ioe_no_driver    = 12;           { no driver for this card         }
   268:D        1          ioe_no_dma       = 13;           { no dma                          }
   269:D        1          ioe_no_word      = 14;           { word operations not allowed     }
   270:D        1          ioe_not_talk     = 15;           { not addressed as talker         }
   271:D        1          ioe_not_lstn     = 16;           { not addressed as listener       }
   272:D        1          ioe_timeout      = 17;           { a timeout has occurred          }
   273:D        1          ioe_not_sctl     = 18;           { not system controller           }
   274:D        1          ioe_bad_stl      = 19;           { bad status or control           }
   275:D        1          ioe_bad_sct      = 20;           { bad set/clear/test operation    }
   276:D        1          ioe_crd_dwn      = 21;           { interface card is dead          }
   277:D        1          ioe_eod_seen     = 22;           { end/eod has occured             }
   278:D        1          ioe_misc         = 23;           { miscellaneous -                 }
   279:D        1                                           { value of param error            }
   280:S
   281:D        1          ioe_sr_toomany   = 304;          { too many chars w/o terminator   } { 0364 TM  8/23/82 }
   282:D        1          ioe_dc_fail      = 306;          { dc interface failure            }
   283:D        1          ioe_dc_usart     = 313;          { USART receive buffer overflow   }
   284:D        1          ioe_dc_ovfl      = 314;          { receive buffer overflow         }
   285:D        1          ioe_dc_clk       = 315;          { missing clock                   }
   286:D        1          ioe_dc_cts       = 316;          { CTS false too long              }
   287:D        1          ioe_dc_car       = 317;          { lost carrier disconnect         }
   288:D        1          ioe_dc_act       = 318;          { no activity disconnect          }
   289:D        1          ioe_dc_conn      = 319;          { connection not established      }
   290:D        1          ioe_dc_conf      = 325;          { bad data bits/par combination   }
   291:D        1          ioe_dc_reg       = 326;          { bad status /control register    }
   292:D        1          ioe_dc_rval      = 327;          { control value out of range      }
   293:D        1          ioe_sr_fail      = 353;          { data link failure               } { 0364 TM  8/23/82 }
   294:S
   295:D        1          no_isc           = 255;          { used for ioe_isc within io errors }
   296:S
```



```
   297:D        1          $PAGE$
   298:D        1          { hpib message constants }
   299:S
   300:D        1          gtl_message      = CHR( 1  );
   301:D        1          sdc_message      = CHR( 4  );
   302:D        1          ppc_message      = CHR( 5  );
   303:D        1          get_message      = CHR( 8  );
   304:D        1          tct_message      = CHR( 9  );
   305:D        1          llo_message      = CHR( 17 );
   306:D        1          dcl_message      = CHR( 20 );
   307:D        1          ppu_message      = CHR( 21 );
   308:D        1          spe_message      = CHR( 24 );
   309:D        1          spd_message      = CHR( 25 );
   310:D        1          unl_message      = CHR( 63 );
   311:D        1          unt_message      = CHR( 95 );
   312:D        1          ppe_message      = CHR( 96 );
   313:D        1          ppd_message      = CHR(112 );
   314:S
   315:S
   316:D        1          talk_constant    = 64;
   317:D        1          listen_constant  = 32;
   318:S
```

```
   319:D        1 $PAGE$
   320:S
   321:D        1         { card type constants }
   322:S
   323:D        1         no_card          = 0 ;
   324:D        1         other_card       = 1 ;
   325:D        1         system_card      = 2 ;
   326:D        1         hpib_card        = 3 ;
   327:D        1         gpio_card        = 4 ;
   328:D        1         serial_card      = 5 ;
   329:D        1         graphics_card    = 6 ;
   330:D        1         srm_card         = 7 ;           { shared resource mgr } { uuuu TM 9/28/82 }
   331:D        1         bubble_card      = 8 ;           { bubble memory       } { jws 3/25/83}
   332:D        1         eprom_prgmr      = 9 ;           { eprom programmer    } { jws 3/25/83}
   333:S
   334:S
   335:D        1         { card id constants             }
   336:D        1         {                               }
   337:D        1         { positive id's are the actual  }
   338:D        1         { card id's                     }
   339:S
   340:D        1         hp98628_dsndl    = -7;          { DSN/DL }              { cccc TM 8/16/82 }
   341:D        1         hp98629          = -6;          { shared resource mgr } { cccc TM 8/16/82 }
   342:D        1         hp_datacomm      = -5;
   343:D        1         hp98620          = -4;
   344:D        1         internal_kbd     = -3;
   345:D        1         internal_crt     = -2;
   346:D        1         internal_hpib    = -1;
   347:S
   348:D        1         no_id            =  0;
   349:S
   350:D        1         hp98624          =  1;          { hpib }
   351:D        1         hp98626          =  2;          { serial }
   352:D        1         hp98622          =  3;          { gpio }
   353:D        1         hp98623          =  4;          { bcd }
   354:D        1         hp98625          =  8;          { disk }
   355:D        1         hp98628_async    = 20;          { hp98628 }
   356:D        1         hpGATOR          = 25;          { bitmap display -- need number jws3 }
   357:D        1         hp98253          = 27;          { EPROM programmer }   { jws 3/25/83 }
   358:D        1         hp98627          = 28;          { graphics }
   359:D        1         hp98259          = 30;          { bubble memory }      { jws 3/25/83 }
   360:D        1         hp98644          = 66;          { serial -- pri. id=2, sec. id=2  jws4 }
   361:S
```



```
   362:D        1   $PAGE$
   363:S
   364:D        1   TYPE
   365:S
   366:D        1         { general declarations }
   367:S
   368:D        1         type_isc         = iominisc..iomaxisc ;
   369:D        1         type_device      = iominisc..iomaxisc*100+99;
   370:D        1         io_bit           = 0..15 ;
   371:D        1         io_byte          = 0..255 ;
   372:D        1         io_word          = -32768..32767 ;
   373:D        1         io_string        = STRING[255];
   374:D        1         io_proc_type     = PROCEDURE;
   375:S
   376:D        1         errlnk_type      = PROCEDURE ( errorcode : INTEGER ;     { 0082 TM 7/23/82 }
   377:D        1                                        VAR s     : io_string );  { 0082 TM 7/23/82 }
   378:S
   379:D        1         { card id and type declarations }
   380:S
   381:D        1         type_of_card     = io_word;
   382:D        1         type_card_id     = io_word;
   383:S
   384:S
   385:D        1         { hpib type declarations }
   386:S
   387:D        1         type_hpib_addr   = 0..31 ;
   388:D        1         type_hpib_line   = ( ren_line ,
   389:D        1                              ifc_line ,
   390:D        1                              srq_line ,
   391:D        1                              eoi_line ,
   392:D        1                              nrfd_line ,
   393:D        1                              ndac_line ,
   394:D        1                              dav_line ,
   395:D        1                              atn_line
   396:D        1                            ) ;
   397:S
   398:S
   399:D        1         { serial type declarations }
   400:S
   401:D        1         type_parity      = ( no_parity ,
   402:D        1                              odd_parity ,
   403:D        1                              even_parity ,
   404:D        1                              zero_parity ,
   405:D        1                              one_parity
   406:D        1                            );
   407:D        1         type_serial_line= ( rts_line ,
   408:D        1                              cts_line ,
   409:D        1                              dcd_line ,
   410:D        1                              dsr_line ,
   411:D        1                              drs_line ,
   412:D        1                              ri_line  ,
   413:D        1                              dtr_line
   414:D        1                            );
```

```
Pascal [Rev 3.0M  6/ 4/84] KERNEL.TEXT            26-Dec-84 21:24:50 Page 13   *** File name: KERNEL ***

 415:D        1          $PAGE$
 416:D        1          { driver declarations }
 417:S
 418:D        1          io_proc          = PROCEDURE ( temp  : ANYPTR );
 419:D        1          io_proc_c        = PROCEDURE ( temp  : ANYPTR;
 420:D        1                                         v     : CHAR );
 421:D        1          io_proc_vc       = PROCEDURE ( temp  : ANYPTR;
 422:D        1                                         VAR v : CHAR);
 423:D        1          io_proc_w        = PROCEDURE ( temp  : ANYPTR;
 424:D        1                                         v     : io_word );
 425:D        1          io_proc_vw       = PROCEDURE ( temp  : ANYPTR;
 426:D        1                                         VAR v : io_word );
 427:D        1          io_proc_s        = PROCEDURE ( temp  : ANYPTR;
 428:D        1                                         reg   : io_word ;
 429:D        1                                         v     : io_word );
 430:D        1          io_proc_vs       = PROCEDURE ( temp  : ANYPTR;
 431:D        1                                         reg   : io_word ;
 432:D        1                                         VAR v : io_word );
 433:D        1          io_proc_l        = PROCEDURE ( temp  : ANYPTR;
 434:D        1                                         line  : io_bit );
 435:D        1          io_proc_vl       = PROCEDURE ( temp  : ANYPTR;
 436:D        1                                         line  : io_bit ;
 437:D        1                                         VAR v : BOOLEAN );
 438:D        1          io_proc_vb       = PROCEDURE ( temp  : ANYPTR;
 439:D        1                                         VAR v : BOOLEAN );
 440:D        1          io_proc_ptr      = PROCEDURE ( temp  : ANYPTR;
 441:D        1                                         v     : ANYPTR );
 442:S
 443:D        1          drv_table_type   = RECORD
 444:D        1                               iod_init  : io_proc ;
 445:D        1                               iod_isr   : ISRPROCTYPE ;
 446:D        1                               iod_rdb   : io_proc_vc ;
 447:D        1                               iod_wtb   : io_proc_c ;
 448:D        1                               iod_rdw   : io_proc_vw ;
 449:D        1                               iod_wtw   : io_proc_w ;
 450:D        1                               iod_rds   : io_proc_vs ;
 451:D        1                               iod_wtc   : io_proc_s ;
 452:D        1                               iod_end   : io_proc_vb ;
 453:D        1                               iod_tfr   : io_proc_ptr ;
 454:D        1                               iod_send  : io_proc_c;
 455:D        1                               iod_ppoll : io_proc_vc ;
 456:D        1                               iod_set   : io_proc_l ;
 457:D        1                               iod_clr   : io_proc_l ;
 458:D        1                               iod_test  : io_proc_vl ;
 459:D        1                             END;
 460:S
 461:S
 462:D        1          { procedure definition for DMA termination procedure }
 463:D        1          io_funny_proc    = RECORD
 464:D        1                               CASE BOOLEAN OF
 465:D        1                                 TRUE:
 466:D        1                                   ( real_proc : io_proc );
 467:D        1                                 FALSE:
 468:D        1                                   ( dummy_pr  : ANYPTR ;
 469:D        1                                     dummy_sl  : ANYPTR )
 470:D        1                             END;
 471:S
 472:D        1          { procedure definition for user EOT/ISR procedures }
 473:S
 474:S                   { Note that the current definition of EOT / ISR procedures is


Pascal [Rev 3.0M  6/ 4/84] KERNEL.TEXT            26-Dec-84 21:24:50 Page 14   *** File name: KERNEL ***

 475:S                     that they will execute ( probably ) inside an ISR and will
 476:S                     therefore be at that interrupt level.  This has several side
 477:S                     effects - such as READ from the keyboard not working- since they
 478:S                     depend on interrupts occuring.
 479:S
 480:S                     The ISR / EOT procedures will work best if they set a flag that
 481:S                     the user program will periodically check.  Basically this means
 482:S                     the user needs to implement his own end-of-line searching.
 483:S
 484:S                     The ISR / EOT procedure is allowed to have a NON-ZERO static link.
 485:S                     This means it can be a nested procedure.  This is potentially
 486:S                     dangerous if the user program is no longer in that chain.
 487:D        1            CAVEAT EMPTOR.                                                  }
 488:S
 489:S
 490:D        1          { interface driver space }
 491:S
 492:D        1          io_temp_type     = PACKED RECORD
 493:D        1                               myisrib   : ISRIB ;
 494:D        1                               user_isr  : io_funny_proc;          {JPC 2/22/82}
 495:D        1                               user_parm : ANYPTR ;                {JPC 2/22/82}
 496:D        1                               card_addr : ANYPTR ;
 497:D        1                               in_bufptr : ANYPTR ;
 498:D        1                               out_bufptr: ANYPTR ;
 499:D        1                               eirbyte   : CHAR ;
 500:D        1                               my_isc    : io_byte ;
 501:D        1                               timeout   : INTEGER ;                { in milliseconds }
 502:D        1                               addressed : io_word ;
 503:D        1                               drv_misc  : ARRAY[1..32] OF CHAR ;
 504:D        1                             END;
 505:S
 506:D        1          io_temp_type2    = PACKED RECORD
 507:D        1                               myisrib   : ISRIB ;
 508:D        1                               user_isr  : io_funny_proc;          {JPC 2/22/82}
 509:D        1                               user_parm : ANYPTR ;                {JPC 2/22/82}
 510:D        1                               card_addr : ANYPTR ;
 511:D        1                               in_bufptr : ANYPTR ;
 512:D        1                               out_bufptr: ANYPTR ;
 513:D        1                               eirbyte   : CHAR ;
 514:D        1                               my_isc    : io_byte ;
 515:D        1                               timeout   : INTEGER ;                { in milliseconds }
 516:D        1                               addressed : io_word ;
 517:D        1                               drv_misc  : ARRAY[1..128] OF CHAR ;
 518:D        1                             END;
 519:S
 520:D        1          io_temp_type3    = PACKED RECORD                         {jw 7/12/82}
 521:D        1                               myisrib   : ISRIB ;                 {jw 7/12/82}
 522:D        1                               user_isr  : io_funny_proc;          {JPC 2/22/82}
 523:D        1                               user_parm : ANYPTR ;                {JPC 2/22/82}
 524:D        1                               card_addr : ANYPTR ;                {jw 7/12/82}
 525:D        1                               in_bufptr : ANYPTR ;                {jw 7/12/82}
 526:D        1                               out_bufptr: ANYPTR ;                {jw 7/12/82}
 527:D        1                               eirbyte   : CHAR ;                  {jw 7/12/82}
 528:D        1                               my_isc    : io_byte ;               {jw 7/12/82}
 529:D        1                               timeout   : INTEGER ;               { in milliseconds }
 530:D        1                               addressed : io_word ;               {jw 7/12/82}
 531:D        1                               drv_misc  : ARRAY[1..160] OF CHAR ; {jw 7/12/82}
 532:D        1                             END;                                  {jw 7/12/82}
 533:S
 534:S
```

```
   535:D        1          pio_tmp_ptr      = ^io_temp_type;
   536:S
   537:S
   538:D        1          isc_table_type   = RECORD
   539:D        1                               io_drv_ptr: ^drv_table_type;
   540:D        1                               io_tmp_ptr: pio_tmp_ptr;
   541:D        1                               card_type : type_of_card;
   542:D        1                               user_time : INTEGER;
   543:D        1                               card_id   : type_card_id;
   544:D        1                               card_ptr  : ANYPTR;
   545:D        1                             END;
```



```
   546:D        1          $PAGE$
   547:D        1          { transfer declarations }
   548:S
   549:D        1          user_tfr_type    = ( dummy_tfr_1 ,                { serial INTR }
   550:D        1                               serial_DMA ,
   551:D        1                               serial_FHS ,
   552:D        1                               serial_FASTEST ,
   553:D        1                               dummy_tfr_2 ,                { serial OVERLAP }
   554:D        1                               overlap_INTR ,
   555:D        1                               overlap_DMA ,
   556:D        1                               overlap_FHS ,
   557:D        1                               overlap_FASTEST ,
   558:D        1                               OVERLAP ) ;
   559:D        1          actual_tfr_type = ( no_tfr ,
   560:D        1                               INTR_tfr ,
   561:D        1                               DMA_tfr ,
   562:D        1                               BURST_tfr ,
   563:D        1                               FHS_tfr ) ;
   564:D        1          dir_of_tfr       = ( to_memory,                   { input = BOOLEAN false }
   565:D        1                               from_memory                  { output= BOOLEAN true }
   566:D        1                             ) ;
   567:S
   568:D        1          buf_type         = PACKED ARRAY[0..maxint] OF CHAR;
   569:S
   570:D        1          buf_info_type    = RECORD
   571:D        1                               drv_tmp_ptr : pio_tmp_ptr;
   572:D        1                               active_isc  : io_byte;
   573:D        1                               act_tfr     : actual_tfr_type ;
   574:D        1                               usr_tfr     : user_tfr_type ;
   575:D        1                               b_w_mode    : BOOLEAN ;              { word = BOOLEAN true }
   576:D        1                               end_mode    : BOOLEAN ;              { eoi  = BOOLEAN true }
   577:D        1                               direction   : dir_of_tfr ;
   578:D        1                               term_char   : -1..255 ;              { -1 = no termination char }
   579:D        1                               term_count  : INTEGER ;
   580:D        1                               buf_ptr     : ^buf_type ;
   581:D        1                               buf_size    : INTEGER ;
   582:D        1                               buf_empty   : ANYPTR ;
   583:D        1                               buf_fill    : ANYPTR ;
   584:D        1                               eot_proc    : io_funny_proc;         {JPC 2/22/82}
   585:D        1                               eot_parm    : ANYPTR ;               {JPC 2/22/82}
   586:D        1                               dma_priority: BOOLEAN;
   587:D        1                             END;
```

```
   588:D        1    $PAGE$
   589:D        1    VAR
   590:S
   591:S                  { dma driver variables - used by DMA_DRV module
   592:S                    and by assembly language drivers.
   593:D        1           ---------- DON'T MOVE ---------- }
   594:S
   595:D    -8  1          dma_ch_0        : io_funny_proc ;
   596:D   -10  1          dma_isc_0       : io_byte ;
   597:D   -18  1          dma_ch_1        : io_funny_proc ;
   598:D   -20  1          dma_isc_1       : io_byte ;
   599:D   -40  1          dma_isrib0      : ISRIB ;
   600:D   -60  1          dma_isrib1      : ISRIB ;
   601:D   -61  1          dma_here        : BOOLEAN;
   602:S
   603:S
   604:S
   605:D   -62  1          io_work_char    : CHAR;
   606:S
   607:S
   608:D   -62  1          { io escape access variables }
   609:S
   610:D   -66  1          ioe_result      : INTEGER;
   611:S                   ioe_isc         : INTEGER;      { must be integer because the sc could
   612:S                                                     be no_sc ( 255 ) or a device
   613:D   -70  1                                            like 701 etc. }
   614:S
   615:S
   616:D   -70  1          isc_table       : PACKED ARRAY [type_isc]
   617:D  -710  1                              OF isc_table_type;
   618:S
   619:S                   io_revid        : STRING[96];   { revision string - added 2/5/82 - TM
   620:S
   621:S                                                     meaning - 'IO 1.0 refers to the IO
   622:S                                                     library ( in system.library ) and
   623:S                                                     the kernel code.  Each driver
   624:S                                                     module will append an indication of
   625:S                                                     its revision like 'G1.0' for GPIO.
   626:S                                                     So - a typical system would have a
   627:S                                                     io_revid of 'IO 1.0 : D1.0 H1.0
   628:S                                                     G1.0 S1.0'.
   629:S
   630:S                                                     known ids -
   631:S
   632:S                                                     main io lib   'IO 1.0 : '
   633:S                                                     dma driver    ' D1.0'
   634:S                                                     hpib driver   ' H1.0'
   635:S                                                     gpio driver   ' G1.0'
   636:S                                                     628 driver    ' S1.0'
   637:S                                                     626 driver    ' R1.0'
   638:D  -808  1                                           }
   639:S
   640:S
   641:D  -816  1          io_error_link   : errlnk_type;  { error msg extension } { 0082 TM 7/23/82 }
   642:S
   643:S                                                    { Will be initialized to a call
   644:S                                                      to a proc in kernal_initialize.
   645:S                                                      To extend error messages, copy old
   646:S                                                      proc value into local var and put in
   647:S                                                      your new proc ( type errlnk_type ).
```



```
   648:S                                                    Your proc should see if it can handle
   649:S                                                    code and return the error msg string,
   650:S                                                    if it can't it should call your
   651:S                                                    local link.
   652:D  -816  1                                           }
   653:S
   654:S
   655:D        1  PROCEDURE io_escape   ( my_code    : INTEGER ;
   656:D  -816  2                           select_code: INTEGER);
   657:D  -816  1  FUNCTION  io_find_isc ( iod_temp   : ANYPTR ): io_byte;
   658:S
   659:S
```

```
   660:D  -816  1 $PAGE$
   661:D  -816  1 IMPLEMENT
   662:S
   663:D        1   PROCEDURE io_escape   ( my_code    : INTEGER ;
   664:D        2                          select_code: INTEGER);
   665:C        2   BEGIN
   666:C        2     ioe_isc    := select_code;
   667:C        2     ioe_result := my_code;
   668:C        2     ESCAPE(ioescapecode);
   669:C        2   END; { of io_escape }
   670:S
   671:S
   672:D        1   FUNCTION  io_find_isc ( iod_temp    : ANYPTR ): io_byte;
   673:D    -4  2   VAR my_ptr     : pio_tmp_ptr;
   674:C        2   BEGIN
   675:C        2     IF iod_temp = NIL
   676:C        3       THEN BEGIN
   677:C        3         io_find_isc := no_isc ;
   678:C        3       END
   679:C        3       ELSE BEGIN
   680:C        3         my_ptr := iod_temp;
   681:C        3         io_find_isc := my_ptr^.my_isc;
   682:C        3       END; { of IF }
   683:C        2   END; { of io_find_isc }
   684:S
   685:C        1 END;    { of iodeclarations }
```



```
   686:D        1 $PAGE$
   687:D        1 EXTERNAL MODULE iocomasm ;
   688:S
   689:S                {
   690:S                  date    10/29/81
   691:S                  update  10/29/81
   692:S
   693:S                  purpose This module contains the iocomasm code
   694:S                          { binary functions & dma control }
   695:D        1         }
   696:S
   697:D        1 IMPORT iodeclarations;
   698:S
   699:D        1 EXPORT
   700:S
   701:D        1   FUNCTION  dma_request ( temp   : ANYPTR ) : INTEGER;
   702:D        1   PROCEDURE dma_release ( temp   : ANYPTR );
   703:D        1   FUNCTION  bit_set     ( v : INTEGER ;
   704:D        2                          b : INTEGER ) : BOOLEAN ;
   705:D        1   FUNCTION  binand      ( x : INTEGER ;
   706:D        2                          y : INTEGER ) : INTEGER ;
   707:D        1   FUNCTION  binior      ( x : INTEGER ;
   708:D        2                          y : INTEGER ) : INTEGER ;
   709:D        1   FUNCTION  bineor      ( x : INTEGER ;
   710:D        2                          y : INTEGER ) : INTEGER ;
   711:D        1   FUNCTION  bincmp      ( x : INTEGER ) : INTEGER ;
   712:S
   713:S
   714:D        1 END; { of iocomasm }
```

```
   715:D        1 $PAGE$
   716:D        1 {*************************************************************************}
   717:D        1 {*                                                                       *}
   718:D        1 {*                                                                       *}
   719:D        1 {*      GENERAL GROUP           GENERAL_0                                *}
   720:D        1 {*                                                                       *}
   721:D        1 {*                                                                       *}
   722:D        1 {*************************************************************************}
   723:S
   724:S
   725:S
   726:D        1 MODULE general_0 ;
   727:S
   728:S                  {
   729:S                    date    07/15/81
   730:S                    update  03/05/84
   731:S
   732:S                    purpose This module contains the LEVEL 0 GENERAL GROUP procedures.
   733:D        1           }
   734:S
   735:S
   736:D        1 IMPORT  iodeclarations;
   737:S
   738:D        1 EXPORT
   739:S
   740:D        1   VAR
   741:D        1     { driver tables }
   742:D  -120  1     kbd_crt_drivers : drv_table_type;
   743:D  -240  1     dummy_drivers   : drv_table_type;
   744:S
   745:S
   746:S
   747:D        1   FUNCTION  ioread_word ( select_code: type_isc ;
   748:D        2                            register   : io_word )
   749:D  -240  2                          : io_word ;
   750:D        1   PROCEDURE iowrite_word( select_code: type_isc ;
   751:D        2                            register   : io_word ;
   752:D  -240  2                            value      : io_word);
   753:D        1   FUNCTION  ioread_byte ( select_code: type_isc ;
   754:D        2                            register   : io_word )
   755:D  -240  2                          : io_byte ;
   756:D        1   PROCEDURE iowrite_byte( select_code: type_isc ;
   757:D        2                            register   : io_word ;
   758:D  -240  2                            value      : io_byte);
   759:D        1   FUNCTION  iostatus    ( select_code: type_isc ;
   760:D        2                            register   : io_word )
   761:D  -240  2                          : io_word ;
   762:D        1   PROCEDURE iocontrol   ( select_code: type_isc ;
   763:D        2                            register   : io_word ;
   764:D  -240  2                            value      : io_word);
   765:D  -240  1   PROCEDURE kernel_initialize;
   766:D  -240  1   PROCEDURE io_system_reset;
   767:S
   768:S
   769:D  -240  1 IMPLEMENT
   770:S
   771:D  -240  1   IMPORT sysglobals ,
   772:D  -240  1          iocomasm  ;
   773:S
   774:S
```



```
   775:S
```

```
   776:D  -240  1   $PAGE$
   777:S
   778:D  -240  1   { these are dummy driver procedures used by the kernel }
   779:S
   780:D        1   PROCEDURE kbd_rdb     ( iod_temp   : ANYPTR ;
   781:D        2                          VAR value  : CHAR );
   782:C        2   BEGIN                                                    { tttt TM 9/22/82 }
   783:C        2     IF EOLN(input)
   784:C        3       THEN BEGIN
   785:C        3         READ(value);
   786:C        3         value:=io_carriage_rtn;
   787:C        3       END
   788:C        3       ELSE BEGIN
   789:C        3         READ(value);
   790:C        3       END; { of IF EOLN }
   791:C        2   END; { of kbd_rdb }                                      { tttt TM 9/22/82 }
   792:S
   793:S
   794:S
   795:D        1   PROCEDURE crt_wtb     ( iod_temp   : ANYPTR ;
   796:D        2                          value      : CHAR );
   797:C        2   BEGIN
   798:C        2     WRITE(value);
   799:C        2   END; { of crt_wtb }
   800:S
   801:S
   802:S
   803:D        1   PROCEDURE simple_init ( temp : ANYPTR );
   804:C        2   BEGIN
   805:C        2     { this initialization will do nothing }
   806:C        2   END; { of simple_init }
   807:S
   808:D  -240  1   { addition of new dummy drivers                            0350 TM 8/19/82 }
   809:D        1   PROCEDURE dummy_driver( temp : ANYPTR );                 { 0350 TM 8/19/82 }
   810:C        2   BEGIN                                                    { 0350 TM 8/19/82 }
   811:C        2     io_escape(ioe_no_driver,io_find_isc(temp));
   812:C        2   END;
   813:S
   814:D        1   PROCEDURE dummy_driver_c( temp : ANYPTR ; dummy : CHAR );
   815:C        2   BEGIN                                                    { 0350 TM 8/19/82 }
   816:C        2     dummy_driver(temp);
   817:C        2   END;
   818:S
   819:D        1   PROCEDURE dummy_driver_a( temp : ANYPTR ; ANYVAR dummy : ANYPTR );
   820:C        2   BEGIN                                                    { 0350 TM 8/19/82 }
   821:C        2     dummy_driver(temp);
   822:C        2   END;
   823:S
   824:D        1   PROCEDURE dummy_driver_w( temp : ANYPTR ; dummy : io_word );
   825:C        2   BEGIN                                                    { 0350 TM 8/19/82 }
   826:C        2     dummy_driver(temp);
   827:C        2   END;
   828:S
   829:D        1   PROCEDURE dummy_driver_dw(temp : ANYPTR ; dummy,d2  : io_word );
   830:C        2   BEGIN                                                    { 0350 TM 8/19/82 }
   831:C        2     dummy_driver(temp);
   832:C        2   END;
   833:S
   834:D        1   PROCEDURE dummy_driver_wa(temp : ANYPTR ; dummy : io_word ; VAR d2 : io_word );
   835:C        2   BEGIN                                                    { 0350 TM 8/19/82 }
```



```
   836:C        2     dummy_driver(temp);
   837:C        2   END;
   838:S
   839:D        1   PROCEDURE dummy_driver_b( temp : ANYPTR ; dummy : io_bit );
   840:C        2   BEGIN                                                    { 0350 TM 8/19/82 }
   841:C        2     dummy_driver(temp);
   842:C        2   END;
   843:S
   844:D        1   PROCEDURE dummy_driver_ba(temp : ANYPTR ; dummy : io_bit  ; VAR d2 : BOOLEAN );
   845:C        2   BEGIN                                                    { 0350 TM 8/19/82 }
   846:C        2     dummy_driver(temp);
   847:C        2   END;
   848:S
   849:S
```

```
850:D  -240  1   $PAGE$
851:S
852:D        1   FUNCTION  ioread_word ( select_code: type_isc ;
853:D        2                            register   : io_word )
854:D        2                          : io_word ;
855:D    -4  2   VAR p : ^io_word;
856:C        2   BEGIN
857:C        2     p := ANYPTR(isc_table[select_code].card_ptr);
858:C        2     IF p = NIL
859:C        3       THEN BEGIN
860:C        3         { error }
861:C        3         io_escape(ioe_no_card,select_code);
862:C        3       END
863:C        3       ELSE BEGIN
864:C        3         p:=ANYPTR(INTEGER(p)+register);
865:C        3         ioread_word:=p^;
866:C        3       END; { of IF }
867:C        2   END; { of ioread_word }
868:S
869:S
870:S
871:D        1   PROCEDURE iowrite_word( select_code: type_isc ;
872:D        2                           register   : io_word ;
873:D        2                           value      : io_word);
874:D    -4  2   VAR p : ^io_word;
875:C        2   BEGIN
876:C        2     p := ANYPTR(isc_table[select_code].card_ptr);
877:C        2     IF p = NIL
878:C        3       THEN BEGIN
879:C        3         { error }
880:C        3         io_escape(ioe_no_card,select_code);
881:C        3       END
882:C        3       ELSE BEGIN
883:C        3         p:=ANYPTR(INTEGER(p)+register);
884:C        3         p^:=value;
885:C        3       END; { of IF }
886:C        2   END; { of iowrite_word }
887:S
888:S
889:S
890:D        1   FUNCTION  ioread_byte ( select_code: type_isc ;
891:D        2                           register   : io_word )
892:D        2                         : io_byte ;
893:D        2   TYPE mycharptr = ^CHAR;
894:D    -4  2   VAR  p          : ^CHAR;
895:C        2   BEGIN
896:C        2     p := ANYPTR(isc_table[select_code].card_ptr);
897:C        2     IF p = NIL
898:C        3       THEN BEGIN
899:C        3         { error }
900:C        3         io_escape(ioe_no_card,select_code);
901:C        3       END
902:C        3       ELSE BEGIN
903:C        3         ioread_byte:=ORD(mycharptr(ADDR(p^,register))^);
904:C        3       END; { of IF }
905:C        2   END; { of ioread_word }
906:S
907:S
908:S
909:D        1   PROCEDURE iowrite_byte( select_code: type_isc ;
```



```
910:D        2                           register   : io_word ;
911:D        2                           value      : io_byte);
912:D        2   TYPE mycharptr = ^CHAR;
913:D    -4  2   VAR  p          : ^CHAR;
914:C        2   BEGIN
915:C        2     p := ANYPTR(isc_table[select_code].card_ptr);
916:C        2     IF p = NIL
917:C        3       THEN BEGIN
918:C        3         { error }
919:C        3         io_escape(ioe_no_card,select_code);
920:C        3       END
921:C        3       ELSE BEGIN
922:C        3         p:=ANYPTR(INTEGER(p)+register);
923:C        3         p^:=CHR(value);
924:C        3       END; { of IF }
925:C        2   END; { of iowrite_byte }
926:S
927:S
928:S
929:D        1   FUNCTION  iostatus    ( select_code: type_isc ;
930:D        2                           register   : io_word )
931:D        2                         : io_word ;
932:D    -2  2   VAR value : io_word;
933:C        2   BEGIN
934:C        2     WITH isc_table[select_code] DO
935:C        3     CALL(io_drv_ptr^.iod_rds,
936:C        3          io_tmp_ptr,
937:C        3          register,
938:C        3          value);
939:C        2     iostatus:=value;
940:C        2   END; { of iostatus }
941:S
942:S
943:S
944:D        1   PROCEDURE iocontrol   ( select_code: type_isc ;
945:D        2                           register   : io_word ;
946:D        2                           value      : io_word);
947:D    -2  2   VAR my_value : io_word;
948:C        2   BEGIN
949:C        2     my_value:=value;
950:C        2     WITH isc_table[select_code] DO
951:C        3     CALL(io_drv_ptr^.iod_wtc,
952:C        3          io_tmp_ptr,
953:C        3          register,
954:C        3          my_value);
955:C        2   END; { of iocontrol }
956:S
```

```
   957:D  -240  1   $PAGE$
   958:D        1   PROCEDURE end_error_link ( errorcode : INTEGER ;              { 0082 TM 7/23/82 }
   959:D        2                             VAR s     : io_string );           { 0082 TM 7/23/82 }
   960:C        2   BEGIN                                                         { 0082 TM 7/23/82 }
   961:C        2     s := 'unrecognized error';                                  { 0082 TM 7/23/82 }
   962:C        2   END; { of end_error_link }                                   { 0082 TM 7/23/82 }
   963:S
   964:D        1   PROCEDURE kernel_initialize;
   965:D    -2  2     VAR io_isc   : type_isc;
   966:D    -6  2         dc_dummy : INTEGER;
   967:D   -10  2         dummy    : INTEGER;
   968:D   -11  2         double   : BOOLEAN;       { indicates an int. that takes 2 s.c. }
   969:D   -16  2         bigtemp  : ^io_temp_type2;
   970:D   -20  2         bigtemp3 : ^io_temp_type3;
   971:D   -20  2
   972:S
   973:C        2   BEGIN
   974:S
   975:C        2     io_revid := 'IO 3.0: ';         { io library revision/id -   jws 8/03/83  }
   976:S
   977:C        2     io_error_link := end_error_link;  { error msg extension }   { 0082 TM 7/23/82 }
   978:S
   979:C        2     double := FALSE;
   980:S
   981:S
   982:C        2     { determine what interfaces are present }
   983:S
   984:C        2     FOR io_isc:=iominisc TO iomaxisc DO WITH isc_table[io_isc] DO BEGIN
   985:C        4       user_time :=0;
   986:C        4       card_id   := no_id;
   987:C        4       IF io_isc<minrealisc
   988:C        5         THEN BEGIN
   989:C        5           card_type := system_card;
   990:C        5           io_tmp_ptr:= NIL;
   991:C        5           CASE io_isc OF
   992:S
   993:C        6             1: BEGIN
   994:C        6                  card_id:=internal_crt;                 { BUG aaaa TM 5/24/82 }
   995:C        6                END;
   996:S
   997:C        6             2: BEGIN
   998:C        6                  card_id:=internal_kbd;                 { BUG aaaa TM 5/24/82 }
   999:C        6                END;
  1000:S
  1001:C        6             3: BEGIN
  1002:C        6                  card_ptr := ANYPTR(5242880);           { dma address $500000 }
  1003:C        6                  { let DMA driver module hunt for dma }
  1004:C        6                  dma_here := FALSE;
  1005:C        6                END;          { of BEGIN }
  1006:S
  1007:S
  1008:C        6             OTHERWISE  BEGIN END;                     { other internal interfaces }
  1009:C        6
  1010:C        6           END; { of CASE io_isc }
  1011:S
  1012:C        5         END
  1013:C        5         ELSE BEGIN
  1014:C        5           card_type:=no_card;
  1015:C        5           IF (io_isc=7) and not(sysflag.nointhpib)     { jws2  6/28/83 }
  1016:C        6             THEN BEGIN
```



```
  1017:C        6               NEW(io_tmp_ptr);                        { get temp space }
  1018:C        6               card_type:=hpib_card;
  1019:C        6               card_id  :=internal_hpib;
  1020:C        6               card_ptr :=ANYPTR(HEX('478000'));
  1021:C        6             END
  1022:C        6             ELSE BEGIN
  1023:C        6               IF double
  1024:C        7                 THEN BEGIN

  1025:S
  1026:C        7                   double := FALSE;

  1027:S
  1028:C        7                 END
  1029:C        7                 ELSE BEGIN
  1030:C        7                   card_ptr  := ANYPTR(HEX('600000')+((io_isc MOD 32)*65536));
  1031:C        7                   TRY
  1032:C        8                     dummy := ioread_byte(io_isc,1);
  1033:C        8                     dummy := dummy MOD 128;       { mask off remote id bit }
  1034:S
  1035:C        8                     IF dummy <= 8 THEN CASE dummy OF
  1036:S
  1037:C       10                       1: BEGIN
  1038:C       10                            card_type:=hpib_card;
  1039:C       10                            card_id  :=hp98624;
  1040:C       10                            NEW(io_tmp_ptr);                  { get temp space }
  1041:C       10                          END;
  1042:S
  1043:C       10                       2: BEGIN
  1044:C       10                            card_type:=serial_card;
  1045:C       10                            card_id  :=hp98626;
  1046:C       10                            NEW(bigtemp3);                    {jw 7/12/82}
  1047:C       10                                                              { get temp space }
  1048:C       10                            io_tmp_ptr := ANYPTR(bigtemp3);   {jw 7/12/82}
  1049:C       10                          END;
  1050:S
  1051:C       10                       3: BEGIN
  1052:C       10                            card_type:=gpio_card;
  1053:C       10                            card_id  :=hp98622;
  1054:C       10                            NEW(io_tmp_ptr);                  { get temp space }
  1055:C       10                          END;
  1056:S
  1057:S
  1058:C       10                       8: BEGIN
  1059:C       10                            card_type:=hpib_card;
  1060:C       10                            card_id  :=hp98625;
  1061:C       10                            NEW(io_tmp_ptr);                  { get temp space }
  1062:C       10                          END;
  1063:S
  1064:C       10                       OTHERWISE BEGIN
  1065:C       10                            NEW(io_tmp_ptr);                  { get temp space }
  1066:C       10                            card_type:=other_card;
  1067:C       10                            { note - this card will get TEMP space }
  1068:C       10                          END;
  1069:S
  1070:C       10                     END { of CASE }
  1071:C       10                     ELSE BEGIN
  1072:S
  1073:C        9                       IF dummy <= 30 THEN                   { jws 3/25/83 }
  1074:S
  1075:C       10                         CASE dummy of                        { jws 3/25/83 }
  1076:S
```

```
 1077:C       11                          25: BEGIN                                { jws3 2/09/84 }
 1078:C       11                                card_id:=hpGATOR;                  { jws3 2/09/84 }
 1079:C       11                                card_type:=graphics_card;          { jws3 2/09/84 }
 1080:C       11                                NEW(io_tmp_ptr);                   { jws3 2/09/84 }
 1081:C       11                              END;                                 { jws3 2/09/84 }
 1082:S
 1083:C       11                          27: BEGIN                                { jws 3/25/83 }
 1084:C       11                                card_id:=hp98253;                  { jws 3/25/83 }
 1085:C       11                                card_type:=eprom_prgmr;            { jws 3/25/83 }
 1086:C       11                                NEW(io_tmp_ptr);                   { jws 3/25/83 }
 1087:C       11                              END;                                 { jws 3/25/83 }
 1088:S
 1089:C       11                          28: BEGIN
 1090:C       11                                card_id := hp98627;
 1091:C       11                                card_type := graphics_card;
 1092:C       11                                double := TRUE;
 1093:C       11                                NEW(io_tmp_ptr);                   { get temp space }
 1094:C       11                              END;
 1095:S
 1096:C       11                          29: BEGIN
 1097:C       11                           { id=29 is also set aside for double wide cards }
 1098:C       11                                double := TRUE;
 1099:C       11                                NEW(bigtemp3);                     { jws 3/25/83}
 1100:C       11                                io_tmp_ptr:=ANYPTR(bigtemp3);      { jws 3/25/83}
 1101:C       11                                card_type:=other_card;             { jws 3/25/83}
 1102:C       11                              END;
 1103:S
 1104:C       11                          30: BEGIN                                { jws 3/25/83 }
 1105:C       11                                card_id:=hp98259;                  { jws 3/25/83 }
 1106:C       11                                card_type:=bubble_card;            { jws 3/25/83 }
 1107:C       11                                NEW(bigtemp);                      { jws 3/25/83 }
 1108:C       11                                io_tmp_ptr:=ANYPTR(bigtemp);       { jws 3/25/83 }
 1109:C       11                              END;
 1110:S
 1111:C       11                          OTHERWISE                                { jws 3/25/83 }
 1112:C       11                              BEGIN                                { jws 3/25/83 }
 1113:C       11                               NEW(bigtemp3);                      { jws 3/25/83 }
 1114:C       11                                io_tmp_ptr:=ANYPTR(bigtemp3);      { jws 3/25/83 }
 1115:C       11                                card_type:=other_card;             { jws 3/25/83 }
 1116:C       11                              END;                                 { jws 3/25/83 }
 1117:S
 1118:C       11                        END;  { of case  }                         { jws 3/25/83 }
 1119:S
 1120:S
 1121:S
 1122:C        9                    IF dummy=52
 1123:C       10                      THEN BEGIN
 1124:C       10                        dc_dummy := ioread_byte(io_isc,16395)*256+ioread_byte(io_isc,16393);
 1125:C       10                        IF dc_dummy < 32768
 1126:C       11                          THEN BEGIN
 1127:C       11                            dc_dummy := ioread_byte(io_isc,dc_dummy*2+1);
 1128:C       11                            IF (dc_dummy MOD 128) = 1
 1129:C       12                              THEN BEGIN
 1130:C       12                                card_id := hp_datacomm;          { since I don't know yet }
 1131:C       12                                card_type := serial_card;
 1132:C       12                             END; { of IF }
 1133:C       11                          END; { of IF }
 1134:C       10                        NEW(bigtemp);
 1135:C       10                        io_tmp_ptr:=ANYPTR(bigtemp);
 1136:C       10                      END; { of BEGIN }
```



```
 1137:S
 1138:C        9                    IF dummy=66                                    { jws4  3/5/84 }
 1139:C       10                      THEN BEGIN                                   { jws4  3/5/84 }
 1140:C       10                        card_id:=hp98644;                          { jws4  3/5/84 }
 1141:C       10                        card_type:=serial_card;                    { jws4  3/5/84 }
 1142:C       10                        NEW(bigtemp3);                             { jws4  3/5/84 }
 1143:C       10                        io_tmp_ptr:=ANYPTR(bigtemp3);              { jws4  3/5/84 }
 1144:C       10                      END;                                         { jws4  3/5/84 }
 1145:S
 1146:C        9                    IF ( dummy > 30 ) AND                          { jws4 3/5/84 }
 1147:C       10                       ( dummy <> 52 ) AND ( dummy<>66 )
 1148:C       10                      THEN BEGIN
 1149:C       10                        NEW(bigtemp3);                             {      TM 8/20/82 }
 1150:C       10                        io_tmp_ptr := ANYPTR(bigtemp3);            {      TM 8/20/82 }
 1151:C       10                        { note - this card WILL get LARGE temp space }
 1152:C       10                        card_type:=other_card;
 1153:C       10                      END;
 1154:S
 1155:C        9                    END; { of IF dummy <= 8 }
 1156:S
 1157:C        8                  RECOVER
 1158:C        8                    BEGIN
 1159:C        8                      IF ( escapecode=-11 ) OR ( escapecode=-12 )
 1160:C        9                        THEN BEGIN
 1161:C        9                          { no card at this address }
 1162:C        9                        END
 1163:C        9                        ELSE BEGIN
 1164:C        9                          { some other problem }
 1165:C        9                          ESCAPE(escapecode);
 1166:C        9                        END;
 1167:C        8                    END; { of RECOVER BEGIN }
 1168:S
 1169:S
 1170:C        7                END; { of IF double }
 1171:C        6              END;   { of IF io_isc=7 }
 1172:C        5            END;     { of IF io_isc<minrealisc THEN/ELSE }
 1173:C        4          END;       { of FOR WITH isc_table[io_isc] DO BEGIN }
 1174:S
 1175:S
 1176:C        2          { set up the driver tables }                             { 0350 TM 8/19/82 }
 1177:C        2                                                                   { 0350 TM 8/19/82 }
 1178:C        2          WITH dummy_drivers DO BEGIN                              { 0350 TM 8/19/82 }
 1179:C        3            iod_init  :=              dummy_driver ;
 1180:C        3            iod_isr   := ISRPROCTYPE (dummy_driver);
 1181:C        3            iod_rdb   := io_proc_vc  (dummy_driver_a);
 1182:C        3            iod_wtb   :=              dummy_driver_c ;
 1183:C        3            iod_rdw   := io_proc_vw  (dummy_driver_a);
 1184:C        3            iod_wtw   :=              dummy_driver_w ;
 1185:C        3            iod_rds   := io_proc_vs  (dummy_driver_wa);
 1186:C        3            iod_wtc   :=              dummy_driver_dw ;
 1187:C        3            iod_end   := io_proc_vb  (dummy_driver_a);
 1188:C        3            iod_ifr   := io_proc_ptr (dummy_driver_a);
 1189:C        3            iod_send  :=              dummy_driver_c ;
 1190:C        3            iod_ppoll := io_proc_vc  (dummy_driver_a);
 1191:C        3            iod_set   :=              dummy_driver_b ;
 1192:C        3            iod_clr   :=              dummy_driver_b ;
 1193:C        3            iod_test  :=              dummy_driver_ba ;              { 0350 TM 8/19/82 }
 1194:C        3          END; { of WITH }                                         { 0350 TM 8/19/82 }
 1195:S
 1196:S
```

```
1197:S
1198:C          2     { initialize the temp space }
1199:C          2     FOR io_isc := iominisc TO iomaxisc DO BEGIN
1200:S
1201:C          3       { set up dummy drivers for the interfaces }
1202:C          3       isc_table[io_isc].io_drv_ptr:=ADDR(dummy_drivers);
1203:S
1204:C          3       IF isc_table[io_isc].io_tmp_ptr <> NIL
1205:C          4         THEN WITH isc_table[io_isc].io_tmp_ptr^ DO BEGIN
1206:C          5           card_addr := isc_table[io_isc].card_ptr;
1207:C          5           eirbyte   := CHR( 0 );
1208:C          5           my_isc    := io_isc ;
1209:C          5           timeout   := 0 ;
1210:C          5           addressed := -1;
1211:C          5           in_bufptr := NIL ;
1212:C          5           out_bufptr:= NIL ;
1213:C          5           user_isr.dummy_sl := NIL;
1214:C          5           user_isr.dummy_pr := NIL;
1215:C          5           user_parm         := NIL;                    {JPC 2/22/82}
1216:C          5           myisrib.INTREGADDR:= NIL;
1217:C          5         END; { of IF  WITH io_tmp_ptr^ DO }
1218:C          3     END; { of FOR DO BEGIN }
1219:S
1220:C          2     dma_isrib0.INTREGADDR := NIL;
1221:C          2     dma_isrib1.INTREGADDR := NIL;
1222:S
1223:S
1224:S                 { note - because of the ISRs - this routine can only be called
1225:S                          once - at INITLIB time.  If it is called again after that
1226:S                          invocation the ISR structure will be in very bad shape and
1227:C          2               will probably hang the machine }
1228:S
1229:S
1230:S
1231:C          2     kbd_crt_drivers:=dummy_drivers;
1232:C          2     WITH kbd_crt_drivers DO BEGIN
1233:C          3       iod_rdb   := kbd_rdb;
1234:C          3       iod_wtb   := crt_wtb;
1235:C          3       iod_init  := simple_init;
1236:C          3     END; { of WITH }
1237:S
1238:S
1239:C          2     isc_table[1].io_drv_ptr := ADDR(kbd_crt_drivers);
1240:C          2     isc_table[2].io_drv_ptr := ADDR(kbd_crt_drivers);
1241:S
1242:S
1243:C          2     { set up clear i/o hook to do an io system reset }
1244:C          2     cleariohook := io_system_reset;
1245:S
1246:S
1247:S
1248:C          2   END; { of kernel_initialize }
1249:S
```



```
1250:D  -240  1   $PAGE$
1251:S
1252:D        1 PROCEDURE io_system_reset;
1253:D    -2  2   VAR io_isc      : type_isc;
1254:C        2   BEGIN
1255:S
1256:C        2     { initialize the interfaces }
1257:S
1258:C        2     FOR io_isc:=iominisc TO iomaxisc DO
1259:C        3       WITH isc_table[io_isc] DO BEGIN
1260:S
1261:C        4         user_time := 0;                                  { user timeout }
1262:S
1263:C        4         IF io_tmp_ptr <> NIL
1264:C        5           THEN WITH io_tmp_ptr^ DO BEGIN
1265:C        6             eirbyte   := CHR( 0 );
1266:C        6             my_isc    := io_isc ;
1267:C        6             timeout   := 0 ;                             { driver timeout }
1268:C        6             user_isr.dummy_sl := NIL;
1269:C        6             user_isr.dummy_pr := NIL;
1270:C        6             user_parm         := NIL;                    {JPC 2/22/82}
1271:C        6           END; { of IF THEN WITH BEGIN  }
1272:S
1273:C        4       END; { of FOR WITH BEGIN }
1274:S
1275:S                 { these two FOR blocks are seperate in case two HPIB interfaces are
1276:C        2         connected in one machine - no funny user isr's will happen.        }
1277:S
1278:C        2     FOR io_isc:=iominisc TO iomaxisc DO
1279:C        3       WITH isc_table[io_isc] DO BEGIN
1280:S
1281:C        4         IF io_drv_ptr^.iod_init <> dummy_driver
1282:C        5           THEN BEGIN
1283:C        5             CALL ( io_drv_ptr^.iod_init ,
1284:C        5                    io_tmp_ptr );
1285:C        5           END; { of IF }
1286:S
1287:C        4       END; { of FOR WITH BEGIN }
1288:S
1289:S
1290:S                 { In case - for some messed up reason ( typically this is when the user
1291:S                   doesn't call ioinitialize/iouninitialize ) - the dma resources are not
1292:C        2         relinquished by the init routines then this will free the resources.  }
1293:S
1294:C        2     IF dma_isc_0 <> no_isc
1295:C        3       THEN BEGIN
1296:C        3         dma_release(isc_table[dma_isc_0].io_tmp_ptr);
1297:C        3       END; { of IF }
1298:S
1299:C        2     IF dma_isc_1 <> no_isc
1300:C        3       THEN BEGIN
1301:C        3         dma_release(isc_table[dma_isc_1].io_tmp_ptr);
1302:C        3       END; { of IF }
1303:S
1304:S
1305:S
1306:C        2   END; { of io_system_reset }
1307:S
1308:S
1309:S
```

```
1310:S
1311:C        1 END;    { of general_0 }
```



```
1312:D        1 $PAGE$
1313:D        1 (*************************************************************************)
1314:D        1 (*                                                                       *)
1315:D        1 (*                                                                       *)
1316:D        1 (*      IOLIBRARY_KERNEL                                                 *)
1317:D        1 (*                                                                       *)
1318:D        1 (*                                                                       *)
1319:D        1 (*************************************************************************)
1320:S
1321:S
1322:S
1323:D        1 IMPORT  general_0 ,
1324:D        1         LOADER ;                                          { 367 TM 9/22/82 }
1325:S
1326:S
1327:C        1 BEGIN
1328:S
1329:C        1   kernel_initialize;
1330:C        1   MARKUSER;                                               { 367 TM 9/22/82 }
1331:C        1 END. { of iolibrary_kernel }
1332:S

No errors. No warnings.
          ***** Nonstandard language features enabled *****
```

KEYS

Description

KEYS contains the initialization routine for the keyboard, the keyboard TM, the general translation routine (executed via KBDTRANSHOOK in A804XDVR), keyboard and RPG ISRs.

Requirements

SYSGLOBALS, ASM, MISC, and SYSDEVS.

Notes

Some of the quoted strings in the following listing (Roman Extension characters, etc.) are printed incorrectly due to the fact that the CRT character set does not match the printer's character set.

See also A804XDVR for calls to some of these routines.

```
     1:S                                                    (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1984.
     4:S           All rights are reserved.  Copying or other
     5:S           reproduction of this program except for archival
     6:S           purposes is prohibited without the prior
     7:S           written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                       RESTRICTED RIGHTS LEGEND
    11:S
    12:S           Use, duplication, or disclosure by the Government
    13:S           is subject to restrictions as set forth in
    14:S           paragraph (b) (3) (B) of the Rights in Technical
    15:S           Data and Computer Software clause in
    16:S           DAR 7-104.9(a).
    17:S
    18:S           HEWLETT-PACKARD COMPANY
    19:D         0 Fort Collins, Colorado                        *)
    20:S
    21:D         0 $MODCAL $
    22:D         0 $heap_dispose off$
    23:D         0 $iocheck off$
    24:D         0 $range off$ $ovflcheck off$
    25:D         0 $debug off$
    26:D         0 $STACKCHECK OFF$
    27:S
    28:D         0 $SEARCH 'INITLOAD','ASM','INIT', 'SYSDEVS'$
    29:S
    30:D         0 program keysinit;
    31:S
    32:D         1 module keys;
    33:D         1 import sysglobals, asm, misc, sysdevs;
    34:D         1 export
    35:D         1   procedure initkeys;
    36:S
    37:D         1 implement
    38:D         1 const
    39:D         1 sysnorm =
    40:D         1     string80['|  k0   |RECALL |CLR-END| CONT  | | STEP  | ALPHA | GRAPH |  k9   |'];
    41:D         1 sysshft =
    42:D         1     string80['|  k0   |RECALL |CLR-END| CONT  | |ANYCHAR| DMP A | DMP G |  k9   |'];
    43:D         1 type
    44:D         1     menu1 = array[boolean] of menutype;
    45:D         1     menu2 = array[m_none..m_sysshift] of menu1;
    46:D         1   const
    47:D         1     mstates = menu2[
    48:D         1                 { no menu } menu1[m_sysnorm,m_sysshift],
    49:D         1                 { normal  } menu1[m_none,m_sysshift],
    50:D         1                 { shifted } menu1[m_sysnorm,m_none]];
    51:D         1 var
    52:D    -1   1   anychar        : boolean;
    53:D    -4   1   buildchar      : shortint;
    54:D    -6   1   buildcount     : 0..4;
    55:D   -14   1   anycharsavehook : kbdhooktype;
    56:S
    57:S
    58:S
    59:D         1 procedure stopaction;
    60:C         2 begin
```



```
    61:C         2   actionspending := 0; escape(-20);
    62:C         2 end;
    63:S
    64:D         1 procedure cntrlpausekey;
    65:D         2 type strin = string[1];
    66:D         2 const qm = strin['?'];
***WARNING: (line   67): 'ADDR' of a constant may not be supported on other implementations
    67:C         2 begin call(debugger,4,integer(addr(qm)),0) end;
    68:D   -14   1
    69:D         1 procedure pausekey;
    70:C         2 begin call(debugger,6,0,0) end;
    71:D   -14   1
    72:D         1 procedure dokbdio(fp: fibp; request: amrequesttype; anyvar buffer: window;
    73:D         2                                                     length, position: integer);
    74:D    -4   2 var   interruptlevel: integer;
    75:D    -5   2       commandinprogress: char;
    76:D    -6   2       zchr: char;
    77:D   -10   2       buf: charptr;
    78:C         2 begin
    79:C         2  ioresult := ord(inoerror);
    80:C         2  buf := addr(buffer);
    81:C         2  case request of
    82:C         3   flush:  {do nothing};
    83:C         3   unitstatus:  fp^.fbusy := keybuffer^.size = 0;
    84:C         3   clearunit:  begin
    85:C         3                 interruptlevel := intlevel;
    86:C         3                 if interruptlevel <1 then setintlevel(1);
    87:C         3                 keybufops(kclear,zchr); { clear typeahead buffer }
    88:C         3                 setintlevel(interruptlevel);
    89:C         3               end;
    90:C         3
    91:S             $if false$
    92:S             writeeol,
    93:S             startwrite,
    94:S               writebytes: crtio(fp, request, buffer, length, 0);
    95:C         3   $end$
    96:S
    97:C         3   readtoeol,
    98:C         3   readbytes,
    99:C         3   startread:
   100:C         3    begin
   101:C         3    if request = readtoeol then
   102:C         4    begin
   103:C         4      buf := addr(buf^, 1);      { format buffer as a string }
   104:C         4      buffer[0] := chr(0);
   105:C         4    end;
   106:C         3    while length>0 do
   107:C         4    begin
   108:C         4      if runlight<>chr(idle) then commandinprogress := runlight;
   109:C         4      interruptlevel := intlevel;
   110:C         4      with keybuffer^ do
   111:C         5      repeat
   112:C         6        while size = 0 do call(kbdwaithook);     { wait for keys }
   113:C         6        setintlevel(7);          {disable interrupts}
   114:C         6        if size = 0 then setintlevel(interruptlevel);
   115:C         6      until size > 0;
   116:C         4      setrunlight(commandinprogress);
   117:S
   118:C         4      with keybuffer^ do buf^:=buffer^[outp];    { BUG FIX #64 }
   119:S
```

```
   120:C        4      if buf^ = chr(etx) then length := 0
   121:C        5                         else length := length-1;
   122:C        4      if (buf^=eol) and (request=readtoeol) then length := 0
   123:C        5      else
   124:C        5      begin
   125:C        5        keybufops(kgetchar,buf^);  { get next character from input }
   126:C        5                                   { BUG FIX #64 }
   127:C        5        fp^.feoln := false;        { set not end of line }
   128:C        5        buf := addr(buf^, 1);      { increment output buffer address }
   129:C        5        if request = readtoeol then buffer[0] := chr(ord(buffer[0])+1);
   130:C        5                                   { increment string size }
   131:C        5      end;
   132:C        4      setintlevel(interruptlevel);  { restore interrupt level}
   133:C        4                                   { BUG FIX #64 }
   134:C        4    end; { while }
   135:C        3
   136:C        3    if request = startread then call(fp^.feot, fp);
   137:C        3    end;
   138:C        3   otherwise ioresult := ord(ibadrequest);
   139:C        3  end;
   140:C        2 end;    { dokbdio }
   141:S
   142:D        1 procedure nonadvkeys;
   143:D        2 type
   144:D        2   aa = packed array['(' .. ','] of char;
   145:D        2 const
   146:D        2   ala = aa['DH@Lb'];    { tables modified/widened 5/7/84 SFB/RQ }
   147:D        2   ale = aa['EIAMe'];
   148:D        2   ali = aa['UYQ]i'];
   149:D        2   alo = aa['FJBNj'];
   150:D        2   alu = aa['GKCOu'];
   151:D        2   aua = aa['`!"Xa'];
   152:D        2   aue = aa['\#$%E'];
   153:D        2   aui = aa['ef&'I'];
   154:D        2   auo = aa['gh_Zi'];
   155:D        2   auu = aa['m-.[U'];
   156:D        2 var
   157:D    -1  2   nc : char;
   158:C        2 begin
   159:C        2   nc := keybuffer^.non_char;
   160:C        2   if langtable[langindex]^.can_nonadv and (nc<>' ') then
   161:C        3   begin
   162:C        3     if (langcom.key=rightchar) or (langcom.key=' ') then langcom.key:=nc
   163:C        4     else
   164:C        4     case langcom.key of
   165:C        5       'a': langcom.key := ala[nc];
   166:C        5       'e': langcom.key := ale[nc];
   167:C        5       'i': langcom.key := ali[nc];
   168:C        5       'o': langcom.key := alo[nc];
   169:C        5       'u': langcom.key := alu[nc];
   170:C        5       'n': if nc = ',' then langcom.key := '7';
   171:C        5       'y': if nc = '+' then langcom.key := 'o';  { SFB/RQ 5/7/84 }
   172:C        5       'A': langcom.key := aua[nc];
   173:C        5       'E': langcom.key := aue[nc];
   174:C        5       'I': langcom.key := aui[nc];               { SFB/RQ 5/7/84 }
   175:C        5       'O': langcom.key := auo[nc];
   176:C        5       'U': langcom.key := auu[nc];
   177:C        5       'N': if nc = ',' then langcom.key := '6';
   178:C        5       'Y': if nc = '+' then langcom.key := 'n';  { SFB/RQ 5/7/84 }
   179:C        5       otherwise
```



```
   180:C        5     end;  {case key}
   181:C        3     keybuffer^.non_char:= ' ';
   182:C        3   end;
   183:C        2 end; {nonadvkeys}
   184:S
   185:D        1 procedure generaltrans;
   186:D        2   { multi lingual translate procedure }
   187:D        2   { all languages except katakana     }
   188:D        2 label 1;
   189:D    -1  2 var tshift : boolean;
   190:C        2 begin
   191:C        2   if langcom.status=0 then
   192:C        3   begin { check key for non advanced key involvement }
   193:C        3     if (langcom.key>=#168)and(langcom.key<=#172) then
   194:C        4      langcom.result:=nonadv_key
   195:C        4     else
   196:C        4     begin langcom.result:=alpha_key; nonadvkeys; end;
   197:C        3   end
   198:C        3   else  { process key code }
   199:C        3   if langcom.data>127 then langcom.result:=ignored_key
   200:C        4   else
   201:C        4   with langcom, langtable[langindex]^, keytable[data] do
   202:C        5   begin
   203:C        5     result := keyclass;
   204:C        5     if result=ignored_key then goto 1;
   205:C        5     extension := extension and not no_extension;
   206:C        5     if extension then
   207:C        6     begin { call alternate semantics routine }
   208:C        6       extension:=false; langindex:=1-langindex; { 0->1 or 1->0 }
   209:C        6       call(langtable[langindex]^.semantics);
   210:C        6       langindex:=1-langindex;
   211:C        6     end
   212:C        6     else
   213:C        6     begin { normal processing }
   214:C        6       control:= control and not no_control;
   215:C        6       shift  := shift and not no_shift;
   216:C        6       tshift := (kbdcapslock and not no_capslock) <> shift;
   217:C        6       key := keys[tshift];
   218:S
   219:C        6       if (result=alpha_key) and control then key := chr(ord(key) mod 32);
   220:S
   221:C        6       if result=alpha_key then nonadvkeys ;
   222:S                { 5/21/84 SFB
   223:S                else
   224:S                if result=special_key then
   225:S                  if (not tshift and not control and (key=rightchar)) then
   226:C        6         begin result:=alpha_key; nonadvkeys; end;  { 5/9/84 RQ/SFB }
   227:C        6     end;
   228:C        5   end;  { with langcom etc }
   229:C        2 1:end;    { generaltrans }
   230:S
   231:D        1 procedure initlang;
   232:D        2 type
   233:D        2   alphatype     = packed array[boolean, 100..125] of char;
   234:D        2   aspecialtype  = packed array[boolean, 80..99] of char;
   235:D        2   b26           = packed array [100..125] of byte;
   236:D        2   bytealphabettype = packed array [boolean] of b26;
   237:S
   238:D        2 const
   239:D        2   alphabet = alphatype['opklqwertyuiasdfghjmzxcvbn',
```

```
240:D        2                        'OPKLQWERTYUIASDFGHJMZXCVBN'];   {shifted}
241:D        2   us    = aspecialtype['0.,+123-456*789/E()^1234567890-={};''.:/ ',
242:D        2                        '0.,+123-456*789/`|\~!@#$%^&*()_+{}:"<>? '];   {shifted}
243:D        2   romanspecials = aspecialtype[
244:D        2 '0.,+123-456*789/E()^'#184#64#35#247#248'^\[]9'#246#254'3|/'#96'<>_ ',
245:D        2 '0.,+123-456*789/`|\~'#184#64#35#247#248'^\{}9'#176#254'3|/'#39'<>_ '];
246:D        2 { ABOVE LINE MODIFIED TO GIVE SHIFT-EXTEND 6 AS ^, NOT | 5/7/84 SFB }
247:D        2   romanalphabet = bytealphabettype
248:D        2                       [b26[214,241,191,187,000,126,215,168,169,170,171,172,212,
249:D        2                            222,228,190,186,188,036,250,000,236,181,189,252,249],
250:D        2                        b26[210,240,191,187,000,126,211,168,169,170,171,172,208,
251:D        2                            222,227,190,186,188,036,250,000,235,180,189,252,249]];
252:D        2 var
253:D    -2  2   i    : shortint;
254:D   -10  2   krec : langkeyrec;
255:C        2 begin
256:C        2   if langtable[0]=nil then new(langtable[0]);
257:C        2   with langtable[0]^ do
258:C        3   begin
259:C        3     langcode  := us_kbd;
260:C        3     semantics := generaltrans;  { default translator }
261:C        3     can_nonadv:= true;
262:C        3     kbdcapslock := true;
263:C        3     kbdaltlock  := false;
264:C        3     transmode:=kshift_extc;
265:C        3     krec.no_capslock:=true;
266:C        3     krec.no_shift   :=false;
267:C        3     krec.no_control :=false;
268:C        3     krec.no_extension:=false;
269:S
270:C        3     krec.keyclass   :=alpha_key;
271:C        3     { key codes 1..3 }
272:C        3     krec.keys[false]:='`'; krec.keys[true]:='~';
273:C        3     keytable[1]:=krec;
274:C        3     krec.keys[false]:='\'; krec.keys[true]:='|';
275:C        3     keytable[2]:=krec;
276:C        3     krec.keys[false]:=#27; krec.keys[true]:=#127; {esc, del}
277:C        3     keytable[3]:=krec;
278:C        3     keytable[3].no_extension:=true; keytable[2].no_control:=true;
279:S
280:C        3     for i:=60 to 125 do
281:C        4     begin
282:C        4       if i>=100 then
283:C        5       begin     { normal alpha keys }
284:C        5         if i=100 then krec.no_capslock:=false;
285:C        5         krec.keys[false]:=alphabet[false,i]; krec.keys[true]:=alphabet[true,i];
286:C        5       end
287:C        5       else
288:C        5       begin     { special alpha keys }
289:C        5         krec.keys[false]:=us[false,i]; krec.keys[true]:=us[true,i];
290:C        5       end;
291:C        4       keytable[i]:=krec;
292:C        4     end;
293:S
294:C        3     { un_implemented key codes }
295:C        3     keytable[0].keyclass:=ignored_key; keytable[4].keyclass:=ignored_key;
296:C        3     keytable[126].keyclass:=ignored_key; keytable[127].keyclass:=ignored_key;
297:S
298:C        3     krec.keyclass:=special_key;
299:C        3     krec.no_shift:=false;     krec.no_control:=false;
```



```
300:C        3     krec.no_capslock:=true;  krec.no_extension:=true;
301:C        3     for i:=5 to 59 do
302:C        4     begin       { special function keys }
303:C        4       krec.keys[false]:=chr(i); krec.keys[true]:=chr(i);
304:C        4       keytable[i]:=krec;
305:C        4     end;
306:C        3     { rightchar(fsp) involved in non advanceing keys }
307:C        3     keytable[39].keys[false]:=rightchar; keytable[39].keys[true]:=rightchar;
308:C        3   end; { with langtable[0] }
309:S
310:C        2   if kbdtype=itfkbd then
311:C        3   begin
312:C        3     if langtable[1]=nil then new(langtable[1]);
313:C        3     langtable[1]^:=langtable[0]^;    { copy US to alternate }
314:C        3     with langtable[1]^ do
315:C        4     begin       { load roman keyboard }
316:C        4       langcode:=roman8_kbd;
317:C        4       keytable[1].keys[false]:=#251; keytable[1].keys[true]:=#253;{<<,>>}
318:C        4       for i:=60 to 125 do
319:C        5       begin
320:C        5         if i>=80 then keytable[i].no_capslock:=true;
321:C        5         with keytable[i] DO
322:C        6         if i>=100 then
323:C        7         begin   { normal alpha keys }
324:C        7           keys[false]:=chr(romanalphabet[false,i]);
325:C        7           keys[true]:= chr(romanalphabet[true,i]);
326:C        7         end
327:C        7         else
328:C        7         begin   { special alpha keys }
329:C        7           keys[false]:=romanspecials[false,i];
330:C        7           keys[true]:= romanspecials[true,i];
331:C        7         end;
332:C        5       end;
333:S
334:C        4       { mark non advanceing keys }
335:C        4       for i:=107 to 111 do keytable[i].keyclass:=nonadv_key;
336:S
337:C        4       { ignore some keys }
338:C        4       keytable[002].keyclass:=ignored_key;
339:C        4       keytable[104].keyclass:=ignored_key;
340:C        4       keytable[120].keyclass:=ignored_key;
341:S
342:C        4       { cancel extension interpretation on all keys }
343:C        4       for i:=0 to 127 do keytable[i].no_extension:=true;
344:S
345:C        4     end; { with langtable[1] }
346:C        3   end;  { if ITF keyboard }
347:C        2 end;    {initlang}
348:S
349:D        1 procedure clearanychar;
350:C        2 begin
351:C        2   if anychar then
352:C        3   begin anychar := false; kbdisrhook := anycharsavehook; end;
353:C        2 end;    { clearanychar }
354:S
355:D        1 procedure rpghandler(var kbdstatus, kbddata: byte; var dokey: boolean);
356:D    -1  2 var key: char;
357:C        2 begin       { only store the key if the typeahead buffer is empty }
358:C        2   case kbdstatus div 16 of
359:C        3     14: {shifted}
```

```
360:C        3          if kbddata >= 128 then key:=chr(1f) else key:=chr(us);
361:C        3      15: {unshifted)
362:C        3          if kbddata >= 128 then key:=chr(fsp) else key:=chr(bs);
363:C        3      otherwise   { ignore if control key down }
364:C        3          beep;  key := ' ';
365:C        3    end;
366:C        2    if (keybuffer^.size=0) and (key<>' ') then
367:C        3    begin
368:C        3      keybufops(kappend,key);  clearanychar;
369:C        3      call(kbdreleasehook);    { signal non empty buffer }
370:C        3    end;
371:C        2  end; { rpghandler }
372:S
373:D        1  procedure keyservice(var kbdstatus, kbddata: byte; var dokey: boolean);
374:D        2  label 1;
375:D    -1  2  var done:  boolean;
376:D    -2  2      small: boolean;
377:D    -4  2      i    : shortint;
378:D    -5  2      c    : char;
379:S
380:D        2      procedure morealpha;
381:C        3      begin
382:C        3        done := true;
383:C        3        if not alphastate then call(togglealphahook)
384:C        4        else if graphicstate then call(togglegraphicshook);
385:C        3      end;
386:D    -5  2
387:D        2      procedure moregraphics;
388:C        3      begin
389:C        3        done := true;
390:C        3        if not graphicstate then call(togglegraphicshook)
391:C        4        else if alphastate then call(togglealphahook);
392:C        3      end;
393:D    -5  2
394:D        2      procedure dumpalpha;
395:C        3      begin done := true; lockedaction(dumpalphahook);
396:C        3      end;
397:D    -5  2
398:D        2      procedure dumpgraphics;
399:C        3      begin done := true; lockedaction(dumpgraphicshook);
400:C        3      end;
401:D    -5  2
402:D        2      procedure  unrecognized;
403:C        3      begin done := true; beep;
404:C        3      end;
405:S
406:D        2      procedure remove(all:boolean);
407:D    -1  3      var dummykey: char;
408:C        3      begin
409:C        3        if keybuffer^.size>0 then
410:C        4        begin
411:C        4          if all then keybufops(kclear,dummykey)
412:C        5                 else keybufops(kgetlast,dummykey); { pop last char }
413:C        4        end
414:C        4        else beep;
415:C        3        done:=true;
416:C        3      end; { remove }
417:S
418:D        2      procedure debugkey;
419:C        3      begin call(debugger,3,langcom.status,langcom.data); end;
```



```
420:S
421:C        2  begin   { keyservice }
422:C        2    if dokey then
423:C        3    with langcom do
424:C        4    begin
425:C        4      done  := false;{done indicates that key is handled immediately)
426:C        4      status  := kbdstatus;
427:C        4      data    := kbddata;
428:C        4      extension:= not odd(kbdstatus div 8);
429:C        4      shift   := not odd(kbdstatus div 16);
430:C        4      control := not odd(kbdstatus div 32);
431:S
432:C        4      if not odd(status) then
433:C        5      begin result:=alpha_key; key:=chr(data); end
434:C        5      else  call(langtable[langindex]^.semantics);
435:S
436:C        4      case result of
437:C        5      alpha_key,    { don't do anything yet }
438:C        5      nonadv_key,{ have non advanceing key }
439:C        5      nona_alpha_key: (* 3.01 BUG FIX  9/19/84 SFB-JWS *);
440:C        5      special_key:
441:C        5        begin { decode special function keys }
442:C        5          small := kbdtype=smallkbd;
443:C        5          case data of
444:C        6            17: if control and not shift then unrecognized   {enter/print}
445:C        7                else if shift and control then dumpgraphics
446:C        8                     else if shift then dumpalpha
447:C        9                          else key:=chr(cr); {enter}
448:C        6            20:begin {system/user}
449:C        6                 if shift then key:='U' else key:='S';
450:C        6                 kbdsysmode:=not shift;
451:C        6                 setstatus(6,key);
452:C        6                 if key='U' then
453:C        7                   if (menustate=m_sysnorm) or (menustate=m_sysshift) then
454:C        8                   begin
455:C        8                     menustate:=m_none;
456:C        8                     keybuffer^.echo:=true;
457:C        8                     keybufops(kdisplay,c);
458:C        8                   end;
459:C        6                 done:=true;
460:C        6               end;
461:C        6            21:begin {menu}
462:C        6                 if kbdsysmode then
463:C        7                 begin
464:C        7                   call(crtllhook,cllclear,i,c); { clear display }
465:C        7                   if menustate>m_sysshift then menustate:=m_none;
466:C        7                   menustate:=mstates[menustate,shift];
467:C        7                   keybuffer^.echo:=(menustate=m_none);
468:C        7                   case menustate of
469:C        8                     m_none    : keybufops(kdisplay,c);
470:C        8                     m_sysnorm : call(crtllhook,clldisplay,sysmenu^,c);
471:C        8                     m_sysshift: call(crtllhook,clldisplay,sysmenushift^,c);
472:C        8                     otherwise
473:C        8                   end;
474:C        7                   done := true;
475:C        7                 end
476:C        7                 else unrecognized;
477:C        6               end;
478:C        6            22:if control then remove(true) else key:=chr(del);{ clear line }
479:C        6            23:if control then remove(true) else key:=chr(ff); { clear display }
```

```
   480:C        6           24:begin KBDCAPSLOCK := not KBDCAPSLOCK; done := true end;
   481:C        6           9,25:                    key := chr(tab);       {tab}
   482:C        6           34:                      key := chr(lf);        {down arrow}
   483:C        6           35:                      key := chr(us);        {up arrow}
   484:C        6                                                           {left arrow, backspace}
   485:C        6           38,46: if control then remove(false) else key := chr(bs);
   486:C        6           39:BEGIN {SFB 5/21/84}  key := chr(fsp); STATUS:=0;
   487:C        6               CALL(LANGTABLE[LANGINDEX]^.SEMANTICS);END; {right arrow}
   488:C        6           40,43:                   key := 'I';            {insert mode}
   489:C        6           41: if small then moregraphics else
   490:C        7                                    key := 'D';            {delete mode}
   491:C        6           42: if shift and small then morealpha else unrecognized;
   492:C        6                                                           {recall}
   493:C        6           44:                      key := 'D';            {delete mode}
   494:C        6           47:                      key := 'R';            {RUN  key}
   495:C        6           48:                      key := 'E';            {EDIT key}
   496:C        6           49: if shift then dumpalpha else morealpha;
   497:C        6           50: if shift then dumpgraphics else moregraphics;
   498:C        6           51: begin                                       {STEP} {ANYCHAR}
   499:C        6               done := true;
   500:C        6               if shift then begin anychar:=true;
   501:C        7                                    anycharsavehook := kbdisrhook;
   502:C        7                                    kbdisrhook := keyservice;
   503:C        7                                    buildcount := 1; buildchar:= 0;
   504:C        7                               end
   505:C        7               else debugkey;                              {STEP key}
   506:C        6               end;
   507:C        6           52: if control then remove(true)
   508:C        7               else if shift then  key := chr(ff)          {clear screen}
   509:C        8                              else  key := chr(del);       {clear line}
   510:C        6           53: if shift and small then dumpalpha else unrecognized;
   511:C        6                                                           {result, set tab}
   512:C        6           54: if shift and small then dumpgraphics else unrecognized;
   513:C        6                                                           {prt all, clr tab}
   514:C        6           6,55: begin                                     {stop} {clear I/O}
   515:C        6                   done := true;
   516:C        6                   clearanychar; lockedaction(stopaction);
   517:C        6                 end;
   518:C        6           5,56: begin                                     {pause}
   519:C        6                   done := true; data:=56;       { change 5 to 56 }
   520:C        6                   if locklevel = 0 then debugkey
   521:C        7                   else if control then lockedaction(cntrlpausekey)
   522:C        8                                   else lockedaction(pausekey);
   523:C        6                 end;
   524:C        6           8,57:                    key := chr(cr);        {ENTER}
   525:C        6           58: begin done := true; debugkey; end;
   526:C        6           7,59:                                           {EXECUTE key}
   527:C        6               if    control then  key := chr(cntrl)       {'control' char}
   528:C        7               else if shift then  key := chr(esc)         {escape}
   529:C        8               else                key := chr(etx);        {EXECUTE}
   530:C        6           otherwise unrecognized; {no such code}
   531:C        6         end;{ case data }
   532:C        5       end;{ decode special function keys }
   533:C        5     ignored_key: goto 1;{ ignore this key }
   534:C        5     end;{ case langcom.result }
   535:S
   536:C        4     if anychar then
   537:C        5     begin
   538:C        5       if done then
   539:C        6       begin     { special key except ANYCHAR terminates ANYCHAR }
```



```
   540:C        6         if not (shift and (data = 51)) then clearanychar;
   541:C        6       end
   542:C        6       else
   543:C        6       begin
   544:C        6         if (key < '0') or (key > '9') then
   545:C        7         begin clearanychar; unrecognized; end
   546:C        7         else
   547:C        7         begin
   548:C        7           buildchar:=buildchar*10+(ord(key)-ORD('0'));
   549:C        7           buildcount:=buildcount+1;
   550:C        7           done := buildcount <= 3;
   551:C        7           if not done then
   552:C        8           begin
   553:C        8             clearanychar; key:=chr(buildchar mod 256);
   554:C        8             status:=0;
   555:C        8             call(langtable[langindex]^.semantics);
   556:C        8           end;
   557:C        7         end;
   558:C        6       end;
   559:C        5     end; { anychar }
   560:S
   561:C        4     if not done then
   562:C        5     begin
   563:C        5       if keybuffer^.size>=keybuffer^.maxsize then beep { no room in buffer }
   564:C        6       else
   565:C        6       if (result=nonadv_key) OR ((result=nona_alpha_key) and not shift)
   566:C        7          then keybufops(knonadvance,key)   { 3.01 BUG FIX 9/19/84 SFB/JWS}
   567:C        7       else
   568:C        7       begin keybufops(kappend,langcom.key); call(kbdreleasehook); end;
   569:C        5     end; { not done }
   570:C        4   end;  { with langcom }
   571:S
   572:C        2 1:end;    { keyservice }
   573:S
   574:D        1 procedure dummykbdwait;
   575:C        2 begin setrunlight(chr(idle)); end;
   576:S
   577:D        1 procedure initkeys;
   578:C        2 begin
   579:C        2   kbdwaithook := dummykbdwait;
   580:C        2   kbdisrhook := keyservice;
   581:C        2   rpgisrhook := rpghandler;
   582:C        2   kbdiohook  := dokbdio;
   583:C        2   anychar := false;
***WARNING: (line  584): 'ADDR' of a constant may not be supported on other implementations
   584:C        2   sysmenu := addr(sysnorm);
***WARNING: (line  585): 'ADDR' of a constant may not be supported on other implementations
   585:C        2   sysmenushift := addr(sysshft);
   586:C        2   initlang;
   587:S
   588:C        2   KBDSETUP(SET_AUTO_REPEAT,4);          {auto repeat period = 40 ms}
   589:C        2   KBDSETUP(SET_AUTO_DELAY,30);          {auto repeat delay  = 300 ms}
   590:C        2   SETRPGRATE(1);                        {rpg interupt rate  = 10 ms}
   591:C        2 end;    { initkeys }
   592:S
   593:C        1 end;    { module keys }
   594:S
   595:D        1 import keys, loader;
   596:S
   597:C        1 begin
```

```
   598:C        1   initkeys; markuser;
   599:C        1 end.
   600:S

No errors. 3 warnings.
             ***** Nonstandard language features enabled *****
```

Description

LIB provides the normal user-interface-level routines for DGL.

Requirements

DGL_TYPES, SYSGLOBALS, SYSDEVS, DGL_VARS, DGL_AUTL, DGL_GEN, ASM, GLE_GEN, GLE_TYPES, GLE_GENI, DGL_CONFG_OUT, DGL_CONFG_IN, DGL_TOOLS, DGL_IBODY, and IODECLARATIONS.

```
     1:D        0 {                                                                    }
     2:D        0 { Pascal work station graphics library                               }
     3:D        0 {                                                                    }
     4:D        0 { Module    = DGL_LIB                                                }
     5:D        0 { Programer = BJS                                                    }
     6:D        0 { Date      = 2/1/81                                                 }
     7:S
     8:D        0 { Rev history:                                                       }
     9:D        0 {  5/21/82  BJS  Set display and locator names to '      ' on term  }
    10:D        0 {  5/21/82  BJS  Fixed inverted window/set_echo_pos bug              }
    11:D        0 {  8/25/82  BJS  Major mods for GLE                                  }
    12:D        0 {  2/17/84  BDS  Changed dynamic to global allocation for Pascal 3.0 }
    13:D        0 { Purpose: Hold normal user interface routines                       }
    14:S
    15:S          {     (c) Copyright Hewlett-Packard Company, 1983.
    16:S                All rights are reserved.  Copying or other
    17:S                reproduction of this program except for archival
    18:S                purposes is prohibited without the prior
    19:S                written consent of Hewlett-Packard Company.
    20:S
    21:S
    22:S                            RESTRICTED RIGHTS LEGEND
    23:S
    24:S                Use, duplication, or disclosure by the Government
    25:S                is subject to restrictions as set forth in
    26:S                paragraph (b) (3) (B) of the Rights in Technical
    27:S                Data and Computer Software clause in
    28:S                DAR 7-104.9(a).
    29:S
    30:S                HEWLETT-PACKARD COMPANY
    31:D        0       Fort Collins, Colorado                                 }
    32:S
    33:D        0 $modcal$
    33:D        0 $include 'OPTIONS'$
    34:S
    35:S           { This include file specifies range checking, debug and other compiler
    36:D        0    options for the graphics library                                  }
    37:S
    38:D        0 $debug OFF$
    39:D        0 $range OFF$
    40:D        0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    41:D        0 $FLOAT_HDW TEST$
    42:S
    43:S
    44:S
    45:S
    46:D        0 $include 'OPTIONS'$
 20000:D        0 $linenum 20000$
 20001:D        0 $search 'TYPES',
 20002:D        0         'DGL_VARS',
 20003:D        0         'GEN',
 20004:D        0         'GLE_LIB',
 20005:D        0         'DGL_C_OUT',
 20006:D        0         'DGL_C_IN',
 20007:D        0         'DGL_TOOLS',
 20008:D        0         'DGL_AUTL',
 20009:D        0         'DGL_IBODY'$
 20010:S
 20011:S
 20012:D        0 module DGL_LIB;
```



```
 20013:S
 20014:D        1 import dgl_types;
 20015:S
 20016:D        1 export
 20017:D        1   procedure graphics_init;
 20018:D        1   procedure set_aspect (x, y : real);
 20019:D        1   procedure set_viewport (vxmin, vxmax, vymin, vymax : real );
 20020:D        1   procedure set_window   (wxmin, wxmax, wymin, wymax : real );
 20021:D        1   procedure display_finit (   fname : gstring255;
 20022:D        2                               device_name : gstring255;
 20023:D        2                               control : integer;
 20024:D        2                             var ierr : integer );
 20025:D        1   procedure display_init (    dev_adr : integer;
 20026:D        2                               control : integer;
 20027:D        2                             var ierr : integer );
 20028:D        1   procedure display_term;
 20029:D        1   procedure make_pic_current;
 20030:D        1   procedure set_timing ( opcode : integer );
 20031:D        1   procedure graphics_term;
 20032:D        1   procedure locator_term;
 20033:S
 20034:D        1   procedure set_color_model ( model : integer);
 20035:D        1   procedure set_line_style ( index : integer);
 20036:D        1   procedure set_color ( index : integer);
 20037:D        1   procedure set_line_width ( index : integer);
 20038:D        1   procedure locator_init (    dev_adr : integer;
 20039:D        2                             var ierr : integer );
 20040:S
 20041:D        1   function graphicserror : integer;
 20042:S
 20043:D        1   procedure move (x,y:real );
 20044:D        1   procedure line (x,y:real );
 20045:D        1   procedure int_polyline ( num_points : integer;
 20046:D        2                             anyvar xvec, yvec : gshortint_list );
 20047:D        1   procedure polyline ( num_points : integer;
 20048:D        2                           anyvar xvec, yvec : greal_list );
 20049:D        1   procedure int_move (ix,iy : gshortint);
 20050:D        1   procedure int_line (ix,iy : gshortint);
 20051:D        1   procedure marker (marker_number : integer);
 20052:D        1   procedure set_display_lim (    dxmin, dxmax,
 20053:D        2                                  dymin, dymax : real;
 20054:D        2                               var ierr : integer);
 20055:D        1   procedure clear_display;
 20056:D        1   procedure input_esc (       opcode : integer;
 20057:D        2                               isize  : integer;
 20058:D        2                               rsize  : integer;
 20059:D        2                           anyvar ilist  : gint_list;
 20060:D        2                           anyvar rlist  : greal_list;
 20061:D        2                           var    ierr   : integer  );
 20062:D        1
 20063:D        1   procedure output_esc (       opcode : integer;
 20064:D        2                                 isize  : integer;
 20065:D        2                                 rsize  : integer;
 20066:D        2                            anyvar ilist  : gint_list;
 20067:D        2                            anyvar rlist  : greal_list;
 20068:D        2                            var    ierr   : integer  );
 20069:S
 20070:D        1   procedure set_text_rot (dx, dy : real);
 20071:D        1   procedure set_char_size (width,height : real);
 20072:D        1   procedure gtext( s : gstring255 );
```

```
20073:S
20074:D         1   procedure set_echo_pos (wx,wy : real);
20075:S
20076:D         1   procedure await_locator (     echo   : integer;
20077:D         2                            var button : integer;
20078:D         2                            var rx, ry : real );
20079:S
20080:D         1   procedure sample_locator (     echo   : integer;
20081:D         2                              var rx, ry : real );
20082:S
20083:D         1   procedure set_locator_lim (    lxmin,lxmax,lymin,lymax : real;
20084:D         2                              var ierr : integer              );
20085:S
20086:D         1   procedure set_color_table ( index : integer;
20087:D         2                               parm1 : real;
20088:D         2                               parm2 : real;
20089:D         2                               parm3 : real);
20090:S
20091:D         1   procedure convert_wtodmm ( wx, wy : real; var mmx,mmy : real);
20092:D         1   procedure convert_wtolmm ( wx, wy : real; var mmx,mmy : real);
20093:S
20094:D         1 implement
20095:S
20096:D         1 import sysglobals,
20097:D         1        sysdevs,
20098:S                 {kbd,
20099:D         1        keys,}
20100:D         1        {asm,}
20101:D         1        dgl_vars,
20102:D         1        dgl_autl,
20103:D         1        dgl_gen,
20104:D         1        gle_types,
20105:D         1        gle_GEN,
20106:D         1        gle_geni,
20107:D         1        dgl_confg_out,
20108:D         1        dgl_confg_in,
20109:D         1        dgl_tools,
20110:D         1        dgl_ibody,
20111:D         1        iodeclarations;
20112:S
20113:D         1 {procedure hpm_dispose ( var object : anyptr; bytesize : integer ); external;}
20114:S
20115:D         1 function graphicserror : integer;
20116:S
20117:D         2 { Purpose:  To return the most resent graphics error number                }
20118:S
20119:C         2 begin
20120:C         2   graphicserror := graphics_error;
20121:C         2 end; { graphicserror }
20122:S
20123:D         1 procedure display_move ( x,y : integer );
20124:S
20125:C         2 begin
20126:C         2   with gle_gcb^ do
20127:C         3     begin
20128:C         3       end_x := x;
20129:C         3       end_y := y;
20130:C         3       call (move,gle_gcb);
20131:C         3     end;
20132:C         2 end;
```



```
20133:S
20134:D         1 procedure display_draw ( x,y : integer );
20135:S
20136:C         2 begin
20137:C         2   with gle_gcb^ do
20138:C         3     begin
20139:C         3       end_x := x;
20140:C         3       end_y := y;
20141:C         3       call (draw,gle_gcb);
20142:C         3     end;
20143:C         2 end;
20144:S
20145:D         1 procedure adjust_echo (var dx,dy : integer);
20146:S
20147:D         2 { Purpose : To adjust echo for rubber band line effects }
20148:S
20149:C         2 begin
20150:C         2   case current_echo_type of
20151:C         3     5 : dy:=d_loc_echo_y;          { horz rubber band line }
20152:C         3     6 : dx:=d_loc_echo_x;          { vert rubber band line }
20153:C         3     7 :                            { snap horz / vert rubber band line }
20154:C         3       if abs(dx-d_loc_echo_x) >= abs(dy-d_loc_echo_y) then
20155:C         4         dy:=d_loc_echo_y
20156:C         4       else
20157:C         4         dx:=d_loc_echo_x;
20158:C         3     otherwise      ;               { all other echos are ok }
20159:C         3   end;  { of  case }
20160:C         2 end; { adjust_echo }
20161:S
20162:D         1 procedure cursor ( x,y : integer);
20163:S
20164:C         2 begin
20165:C         2   display_move(x-8,y);
20166:C         2   display_draw(x+8,y);
20167:S
20168:C         2   display_move(x,y+8);
20169:C         2   display_draw(x,y-8);
20170:C         2 end;
20171:S
20172:D         1 procedure echo_cursor (dx,dy : integer);
20173:S
20174:D         2 { Purpose : To perform the current echo on a raster display }
20175:S
20176:C         2 begin
20177:S
20178:C         2   case current_echo_type of
20179:S
20180:C         3     1,2 : cursor(dx,dy);
20181:S
20182:C         3     3 :
20183:C         3         with gcb^,gcb^.max_disp_lim do
20184:C         4           begin {full screen}
20185:C         4             display_move(trunc(xmin),dy);
20186:C         4             display_draw(trunc(xmax),dy);
20187:C         4             display_move(dx,trunc(ymin));
20188:C         4             display_draw(dx,trunc(ymax));
20189:C         4             display_move(dx,dy);          { set cp to cursor center }
20190:C         4           end;
20191:S
20192:C         3     4,5,6,7 :
```

```
20193:C        3          begin {rubber bands}
20194:C        3            adjust_echo (dx,dy);          { are dx, dy correct for this echo? }
20195:C        3            display_move(d_loc_echo_x,d_loc_echo_y);
20196:C        3            display_draw(dx,dy);
20197:C        3            cursor(dx,dy);
20198:C        3          end;
20199:S
20200:C        3      8 :
20201:C        3          begin {rubber band box}
20202:C        3            display_move(d_loc_echo_x,d_loc_echo_y);
20203:C        3            display_draw(d_loc_echo_x,dy);
20204:C        3            display_draw(dx,dy);
20205:C        3            display_draw(dx,d_loc_echo_y);
20206:C        3            display_draw(d_loc_echo_x,d_loc_echo_y);
20207:C        3            cursor(dx,dy);
20208:C        3          end;
20209:S
20210:C        3       otherwise ;    { no echo }
20211:S
20212:C        3    end;  { of case }
20213:C        2
20214:C        2 end; { display_echo }
20215:S
20216:D        1 procedure DGL_CURSOR (  gle_gcb : graphics_control_block_ptr );
20217:S
20218:D        2 VAR
20219:D    -8  2   x,y : integer;
20220:D   -12  2   CPX : INTEGER;
20221:D   -16  2   CPY : INTEGER;
20222:S
20223:C        2 begin
20224:C        2   with gle_gcb^ do
20225:C        3     begin
20226:C        3       x := info1;
20227:C        3       y := info2;
20228:C        3       CPX := CURRENT_POS_X; { SAVE CP }
20229:C        3       CPY := CURRENT_POS_Y;
20230:C        3       gle_await_blanking(gle_gcb);
20231:C        3       if current_cursor_state = 1 then  { remove old cursor }
20232:C        4         echo_cursor (current_cursor_x, current_cursor_y);
20233:C        3       if (info3 = 1) then               { draw new cursor }
20234:C        4         echo_cursor (x,y);
20235:S
20236:C        3       current_cursor_state := info3;
20237:C        3       current_cursor_x := x;
20238:C        3       current_cursor_y := y;
20239:S
20240:C        3       END_X := CPX;        { RESTORE CP }
20241:C        3       END_Y := CPY;
20242:C        3       GLE_MOVE(GLE_GCB);
20243:C        3     end;
20244:C        2 end;
20245:S
20246:D   -16  1 procedure convert_wtodmm ( wx, wy : real; var mmx,mmy : real);
20247:S
20248:D   -16  2 var
20249:D   -32  2   dx,dy : real;
20250:S
20251:C        2 begin
20252:C        2   ck_system_init;
```



```
20253:C        2   ck_display_init;
20254:C        2   dx := wx * xwtod_scale + xwtod_offset;
20255:C        2   dy := wy * ywtod_scale + ywtod_offset;
20256:C        2   with gcb^,gcb^.max_disp_lim,gle_gcb^ do
20257:C        3     begin
20258:C        3       mmx := (dx - xmin) / display_res_x;
20259:C        3       mmy := (dy - ymin) / display_res_y;
20260:C        3     end;
20261:C        2 end;
20262:S
20263:D   -16  1 procedure convert_wtolmm ( wx, wy : real; var mmx,mmy : real);
20264:S
20265:D   -16  2 var
20266:D   -32  2   dx,dy : real;
20267:D   -48  2   tx,ty : real;
20268:S
20269:C        2 begin
20270:C        2   ck_system_init;
20271:C        2   ck_locator_init;
20272:C        2   dx := wx * xwtod_scale + xwtod_offset;
20273:C        2   dy := wy * ywtod_scale + ywtod_offset;
20274:C        2   with gcb^,gle_gcb1^ do
20275:C        3     begin
20276:C        3       { convert display to locator }
20277:C        3       tx := ((dx-cur_disp_lim.xmin) / xltod_scale) + log_loc_lim.xmin;
20278:C        3       ty := ((dy-cur_disp_lim.ymin) / yltod_scale) + log_loc_lim.ymin;
20279:S
20280:C        3       { convert to mm }
20281:C        3       mmx := (tx - max_loc_lim.xmin) / input_res_x;
20282:C        3       mmy := (ty - max_loc_lim.ymin) / input_res_y;
20283:C        3     end;
20284:C        2 end;
20285:S
20286:D   -32  1 procedure set_viewport (vxmin, vxmax, vymin, vymax : real );
20287:S
20288:D   -32  2 { Purpose:  To set the viewport                                          }
20289:S
20290:C        2 begin
20291:C        2   ck_system_init;
20292:C        2
20293:C        2   {ck parms}
20294:S
20295:C        2   if (vxmin >= vxmax) or (vymin >= vymax) then error (err_bad_parms);
20296:S
20297:C        2   with gcb^ do
20298:C        3     begin
20299:C        3       if (vxmin < 0.0) or                {ck with vir limits}
20300:C        4          (vymin < 0.0) or
20301:C        4          (vxmax > cur_vir_lim.xlim) or
20302:C        4          (vymax > cur_vir_lim.ylim) then
20303:C        4         error (err_out_virt);
20304:S
20305:C        3       with viewport_lim do
20306:C        4         begin
20307:C        4           xmin := vxmin;                 { set the new limits }
20308:C        4           xmax := vxmax;
20309:C        4           ymin := vymin;
20310:C        4           ymax := vymax;
20311:C        4         end;
20312:S
```

```
20313:C        3       calculate_viewing;
20314:S
20315:C        3       { set flag so character size will be recalculated }
20316:S
20317:C        3       calc_text_xform := true;
20318:C        3     end
20319:C        3 end; { set_viewport }
20320:S
20321:D   -32  1 procedure set_window (wxmin, wxmax, wymin, wymax : real );
20322:S
20323:D   -32  2 { Purpose:  To set the window                                          }
20324:S
20325:C        2 begin
20326:C        2   ck_system_init;
20327:S
20328:C        2   {ck parms}
20329:S
20330:C        2   if (wxmin = wxmax) or (wymin = wymax) then error (err_bad_parms);
20331:S
20332:C        2   with gcb^ do
20333:C        3     with window_lim do
20334:C        4       begin
20335:C        4         xmin := wxmin;              { set the new window }
20336:C        4         xmax := wxmax;
20337:C        4         ymin := wymin;
20338:C        4         ymax := wymax;
20339:C        4       end;
20340:S
20341:C        2   calculate_viewing;
20342:S
20343:C        2   { set flag so character xform will be recalculated }
20344:S
20345:C        2   calc_text_xform := true;
20346:C        2 end; { set_window }
20347:S
20348:D   -16  1 procedure set_aspect (x, y : real);
20349:S
20350:D   -16  2 { Purpose:  To set the aspect ratio                                    }
20351:S
20352:C        2 begin
20353:C        2   ck_system_init;
20354:S
20355:C        2   {ck parms }
20356:C        2   if (x <= 0.0) or (y <= 0.0) then error (err_aspect);
20357:S
20358:C        2   with gcb^ do
20359:C        3     { calc new limits }
20360:C        3     with cur_vir_lim do
20361:C        4       begin
20362:C        4         aspect_ratio := y / x;
20363:C        4         if aspect_ratio <= 1.0 then
20364:C        5           begin
20365:C        5             xlim := 1.0;
20366:C        5             ylim := aspect_ratio;
20367:C        5           end
20368:C        5         else
20369:C        5           begin
20370:C        5             xlim := 1.0 / aspect_ratio;
20371:C        5             ylim := 1.0;
20372:C        5           end;
```



```
20373:C        4         { set viewport to new limits }
20374:C        4         set_viewport( 0.0, xlim, 0.0, ylim);
20375:C        4       end;
20376:C        2 end; { set_aspect }
20377:S
20378:D        1 procedure make_pic_current;
20379:S
20380:C        2 begin
20381:C        2   ck_system_init;
20382:C        2   ck_display_init;
20383:C        2   gle_flush_buffer ( gle_gcb );
20384:C        2 end;
20385:S
20386:D        1 procedure set_timing ( opcode : integer );
20387:S
20388:C        2 begin
20389:C        2   ck_system_init;
20390:C        2   if (opcode < 0) or (opcode >1) then error(err_bad_parms);
20391:C        2   gcb^.dgl_current_timming_mode := opcode;
20392:C        2   if disp_init then
20393:C        3     begin
20394:C        3       gle_gcb^.info1 := opcode;
20395:C        3       gle_buffer_mode ( gle_gcb );
20396:C        3     end;
20397:C        2 end;
20398:S
20399:D        1 procedure set_color_model ( model : integer);
20400:S
20401:C        2 begin
20402:C        2   ck_system_init;
20403:S
20404:C        2   if (model<1) or (model>2) then error(err_bad_parms);
20405:C        2   gcb^.dgl_current_color_model := model;
20406:C        2 end;
20407:S
20408:D        1 procedure set_color_table ( index : integer;
20409:D        2                             parm1 : real;
20410:D        2                             parm2 : real;
20411:D   -24  2                             parm3 : real);
20412:S
20413:C        2 begin
20414:C        2   ck_system_init;
20415:C        2   ck_display_init;
20416:C        2   with gcb^,gle_gcb^ do
20417:C        3     begin
20418:C        3       if (index >= 0) and  (index <= color_table_size) then
20419:C        4         begin
20420:C        4           dgl_polygon_color_current := false;  { dither pattern is wrong }
20421:C        4           if ((0 > parm1) or (parm1 > 1)) or
20422:C        5              ((0 > parm2) or (parm2 > 1)) or
20423:C        5              ((0 > parm3) or (parm3 > 1)) then error (err_bad_parms);
20424:C        4           call (proc_color_table,index,parm1,parm2,parm3);
20425:C        4           { always recalculate line color (2.1 buug fix) }
20426:C        4           call (proc_color,dgl_current_color);
20427:C        4         end;
20428:C        3     end;
20429:C        2 end;
20430:S
20431:D        1 procedure set_line_width (index : integer);
20432:S
```

```
20433:D        2 { Purpose:  To set the line width primitives will be drawn with             }
20434:S
20435:C        2 begin
20436:C        2   ck_system_init;
20437:C        2   ck_display_init;
20438:C        2   with gle_gcb^ do
20439:C        3     begin
20440:C        3       if (index < 1 ) or ( index > linewidths ) then index := 1;
20441:C        3       gcb^.dgl_current_linewidth := index;
20442:C        3       info1 := index;
20443:C        3       gle_linewidth ( gle_gcb );
20444:C        3     end;
20445:C        2 end; { set_linewidth }
20446:S
20447:D        1 procedure set_color (index : integer);
20448:S
20449:D        2 { Purpose:  To set the color primitives will be drawn with                  }
20450:S
20451:C        2 begin
20452:C        2   ck_system_init;
20453:C        2   ck_display_init;
20454:S
20455:C        2   with gcb^,gle_gcb^ do
20456:C        3     begin
20457:C        3       if  (index < 0) or
20458:C        4          ((index > gamut) and
20459:C        4          ((color_table_size = 0) or (index > color_table_size))) then index := 1;
20460:C        3       { optimize changing color on raster devices (2.1 bug fix) }
20461:C        3       if ((dgl_current_color <> index) or (complement_support <> 1))  then
20462:C        4         begin
20463:C        4           call (proc_color,index);
20464:C        4           dgl_current_color := index;
20465:C        4         end;
20466:C        3     end;
20467:C        2 end; { set_color }
20468:S
20469:D        1 procedure set_line_style ( index : integer);
20470:S
20471:D        2 { Purpose:  To set the linestyle that primitives are drawn with             }
20472:S
20473:C        2 begin
20474:C        2   ck_system_init;
20475:C        2   ck_display_init;
20476:S
20477:C        2   with gcb^ do
20478:C        3     begin
20479:C        3       if (index < 1) or (index > number_dgl_linestyles) then index := 1;
20480:S                 (if dgl_current_linestyle <> index then
20481:C        3         begin)
20482:C        3           dgl_current_linestyle := index;
20483:C        3           call (proc_linestyle,index);
20484:C        3         (end;)
20485:C        3     end;
20486:C        2 end; { set_line_style }
20487:S
20488:D        1 procedure set_display_lim (    dxmin, dxmax, dymin, dymax : real;
20489:D   -32  2                               var ierr : integer);
20490:S
20491:D   -32  2 { Purpose : To set the logical display limits                          }
20492:S
```



```
20493:D   -32  2 var
20494:D   -40  2   txmin : real;
20495:D   -48  2   txmax : real;
20496:D   -56  2   tymin : real;
20497:D   -64  2   tymax : real;
20498:S
20499:C        2 begin
20500:C        2   ck_system_init;
20501:C        2   ck_display_init;
20502:S
20503:C        2   {WRITELN(dxmin:19:18,dxmax:19:18,dymin:19:18,dymax:19:18);}
20504:C        2   { ck parms }
20505:C        2   if ((dxmin < dxmax) and (dymin < dymax)) then
20506:C        3     with gcb^ do
20507:C        4       with max_disp_lim do
20508:C        5         begin
20509:C        5           with gle_gcb^ do
20510:C        6             begin
20511:C        6               txmin := (dxmin * display_res_x) + xmin;
20512:C        6               txmax := (dxmax * display_res_x) + xmin;
20513:S
20514:C        6               tymin := (dymin * display_res_y) + ymin;
20515:C        6               tymax := (dymax * display_res_y) + ymin;
20516:C        6             end;
20517:S
20518:C        5           { make sure new logical imits are within the physical limits       }
20519:C        5           if (txmin >= xmin) and (txmax <= xmax + eight_diget_epsilon) and
20520:C        6              (tymin >= ymin) and (tymax <= ymax + eight_diget_epsilon) then
20521:C        6             begin
20522:C        6               display_limits (txmin,txmax,tymin,tymax);
20523:S
20524:S                         { set flag indicating that the char size needs to be
20525:S                           recalculated.  This is done since the physical character size
20526:C        6                  may change due to this procedure                          }
20527:S
20528:C        6               calc_text_xform := true;
20529:S
20530:C        6               ierr := 0;
20531:C        6             end
20532:C        6           else ierr := 2;
20533:C        5         end
20534:C        5   else
20535:C        3     ierr := 1;
20536:C        2 end; { set_display_lim }
20537:S
20538:S
20539:D        1 procedure clear_display;
20540:S
20541:D        2 { Purpose : To clear to display                                          }
20542:S
20543:C        2 begin
20544:C        2   ck_system_init;
20545:C        2   ck_display_init;
20546:S
20547:C        2   with gle_gcb^ do
20548:C        3     begin
20549:C        3       info1 := -1;        { clear all planes }
20550:C        3       info2 := gcb^.dgl_background_index;
20551:C        3       gle_clear ( gle_gcb );
20552:C        3     end;
```

```
20553:C        2 end; { Clear_display }
20554:S
20555:S
20556:D        1 $stackcheck off$
20557:D   -16  1 procedure move (x, y : real);
20558:S
20559:D   -16  2 { Purpose : To change the current position                          }
20560:S
20561:C        2 begin
20562:C        2   if disp_init then
20563:C        3     begin
20564:C        3       { save world cp }
20565:C        3       int_cp := false; { cp saved as real value }
20566:C        3       world_real_cpx := x;
20567:C        3       world_real_cpy := y;
20568:S
20569:C        3       { calc new device dependent cp }
20570:C        3       cpx := trunc ( x * xwtod_scale + xwtod_offset );
20571:C        3       cpy := trunc ( y * ywtod_scale + ywtod_offset );
20572:S
20573:C        3       {WRITELN('MOVE (',CPX:5,',',CPY:5,')');}
20574:C        3       with gle_gcb^ do
20575:C        4         begin
20576:C        4           end_x := cpx;
20577:C        4           end_y := cpy;
20578:C        4           call (move,gle_gcb);
20579:C        4         end;
20580:C        3     end
20581:C        3   else
20582:C        3     if system_init then error (err_dis_int)
20583:C        4     else error (err_sys_int);
20584:C        2 end; { move }
20585:S
20586:S
20587:D   -16  1 procedure line (x, y : real);
20588:S
20589:D   -16  2 { Purpose : To draw a line                                           }
20590:S
20591:C        2 begin
20592:C        2   if disp_init then
20593:C        3     begin
20594:C        3       { save world cp }
20595:C        3       int_cp := false; { cp saved as real value }
20596:C        3       world_real_cpx := x;
20597:C        3       world_real_cpy := y;
20598:S
20599:C        3       { calc new device dependent cp }
20600:C        3       cpx := trunc ( x * xwtod_scale + xwtod_offset );
20601:C        3       cpy := trunc ( y * ywtod_scale + ywtod_offset );
20602:S
20603:C        3       {WRITELN('LINE (',CPX:5,',',CPY:5,')');}
20604:C        3       with gle_gcb^ do
20605:C        4         begin
20606:C        4           end_x := cpx;
20607:C        4           end_y := cpy;
20608:C        4           call (draw,gle_gcb);
20609:C        4         end;
20610:C        3     end
20611:C        3   else
20612:C        3     if system_init then error (err_dis_int)
```



```
20613:C        4     else error (err_sys_int);
20614:C        2 end; { line }
20615:S
20616:D        1 procedure int_move (ix, iy : gshortint);
20617:S
20618:D        2 { Purpose : To move the current position                            }
20619:S
20620:C        2 begin
20621:C        2   if disp_init then
20622:C        3     begin
20623:C        3       { use the normal move routine unless the short flag is set }
20624:C        3       if short_flag then
20625:C        4         begin
20626:C        4           { save world cp }
20627:C        4           int_cp := true;      { cp saved as gshortint }
20628:C        4           world_int_cpx := ix;
20629:C        4           world_int_cpy := iy;
20630:S
20631:C        4           with gle_gcb^ do
20632:C        5             begin
20633:C        5               end_x := ix;
20634:C        5               end_y := iy;
20635:C        5               dgl_scaled_move; { perform a scaled move }
20636:C        5             end;
20637:C        4         end
20638:C        4       else move(ix,iy);
20639:C        3     end
20640:C        3   else
20641:C        3     if system_init then error (err_dis_int)
20642:C        4     else error (err_sys_int);
20643:C        2 end; { int_move }
20644:S
20645:S
20646:D        1 procedure int_line (ix, iy : gshortint);
20647:S
20648:S
20649:D        2 { Purpose : To set the logical display limits                        }
20650:S
20651:C        2 begin
20652:C        2   if disp_init then
20653:C        3     begin
20654:C        3       { use normal line unless short_flag is set }
20655:C        3       if short_flag then
20656:C        4         begin
20657:C        4           { save world cp }
20658:C        4           int_cp := true;      { cp saved as gshortint }
20659:C        4           world_int_cpx := ix;
20660:C        4           world_int_cpy := iy;
20661:C        4           with gle_gcb^ do
20662:C        5             begin
20663:C        5               end_x := ix;
20664:C        5               end_y := iy;
20665:C        5               dgl_scaled_draw;   { perform a scaled draw }
20666:C        5             end;
20667:C        4         end
20668:C        4       else
20669:C        4         line(ix,iy);
20670:C        3     end
20671:C        3   else
20672:C        3     if system_init then error (err_dis_int)
```

```
20673:C        4     else error (err_sys_int);
20674:C        2 end; { int_line }
20675:S
20676:D        1 procedure int_polyline ( num_points : integer;
20677:D        2                          anyvar xvec, yvec : gshortint_list );
20678:D        2 var
20679:D    -4  2   i : integer;
20680:S
20681:C        2 begin
20682:C        2   ck_system_init;
20683:C        2   ck_display_init;
20684:S
20685:C        2   if num_points <= 0 then error(err_neg_points);
20686:S
20687:C        2   int_move ( xvec[1], yvec[1] );
20688:C        2   for i := 2 to num_points do
20689:C        3     int_line ( xvec[i], yvec[i]);
20690:C        2 end;
20691:S
20692:D        1 procedure polyline ( num_points : integer;
20693:D        2                      anyvar xvec, yvec : greal_list );
20694:D        2 var
20695:D    -4  2   i : integer;
20696:S
20697:C        2 begin
20698:C        2   ck_system_init;
20699:C        2   ck_display_init;
20700:S
20701:C        2   if num_points <= 0 then error(err_neg_points);
20702:S
20703:C        2   move ( xvec[1], yvec[1] );
20704:C        2   for i := 2 to num_points do
20705:C        3     line ( xvec[i], yvec[i]);
20706:C        2 end;
20707:D        1 $stackcheck on$
20708:S
20709:D  -256  1 procedure gtext( s : gstring255 );
20710:S
20711:D  -256  2 { PURPOSE : To draw a text string                                    }
20712:S
20713:C        2 begin
20714:C        2   if disp_init then
20715:C        3     begin
20716:C        3       if calc_text_xform then
20717:C        4         with gcb^ do
20718:C        5           begin
20719:C        5             set_char_size (dgl_char_width, dgl_char_height);
20720:C        5             set_text_rot  (char_rot_w, char_rot_h);
20721:C        5             calc_text_xform := false;
20722:C        5           end;
20723:C        3       with gle_gcb^ do
20724:C        4         begin
20725:C        4           info_ptr1 := addr(s[1]);
20726:C        4           info1 := strlen(s);
20727:C        4         end;
20728:C        3       gle_text ( gle_gcb );
20729:C        3     end
20730:C        3   else
20731:C        3     if system_init then error (err_dis_int)
20732:C        4     else error (err_sys_int);
```



```
20733:C        2 end; { gtext }
20734:S
20735:D        1 procedure marker ( marker_number : integer );
20736:S
20737:C        2 begin
20738:C        2   ck_system_init;
20739:C        2   ck_display_init;
20740:C        2   if (marker_number < 1) or (marker_number > 19) then marker_number := 1;
20741:S
20742:C        2   with gle_gcb^ do
20743:C        3     begin
20744:C        3       info1 := marker_number;
20745:C        3       gle_set_marker ( gle_gcb );
20746:C        3       gle_marker ( gle_gcb );
20747:C        3     end;
20748:C        2 end;
20749:S
20750:D        1 procedure input_esc (       opcode : integer;
20751:D        2                             isize  : integer;
20752:D        2                             rsize  : integer;
20753:D        2                      anyvar ilist  : gint_list;
20754:D        2                      anyvar rlist  : greal_list;
20755:D        2                      var    ierr   : integer  );
20756:D        2
20757:D        2 { Purpose : To perform an input escape function                    }
20758:S
20759:C        2 begin
20760:C        2   ck_system_init;
20761:C        2   ck_display_init;
20762:S
20763:C        2   ierr := opcode_ck ( opcode,isize,rsize);
20764:C        2   call (gcb^.proc_input_esc,opcode,isize,rsize,ilist,rlist,ierr);
20765:S
20766:C        2 end; { input_esc }
20767:S
20768:D        1 procedure output_esc (       opcode : integer;
20769:D        2                              isize  : integer;
20770:D        2                              rsize  : integer;
20771:D        2                       anyvar ilist  : gint_list;
20772:D        2                       anyvar rlist  : greal_list;
20773:D        2                       var    ierr   : integer  );
20774:S
20775:D        2 { Purpose : To perform an output escape funtion                    }
20776:S
20777:C        2 begin
20778:C        2   ck_system_init;
20779:C        2   ck_display_init;
20780:S
20781:C        2   ierr := opcode_ck ( opcode,isize,rsize);
20782:C        2   call (gcb^.proc_output_esc,opcode,isize,rsize,ilist,rlist,ierr);
20783:C        2 end. { output_esc }
20784:S
20785:D   -16  1 procedure set_text_rot (dx, dy : real);
20786:S
20787:D   -16  2 { PURPOSE : To set the new text rotation vectors                    }
20788:S
20789:D   -16  2 { calc normalize vector and save }
20790:D   -16  2 { set flag so text xform is recalculated }
20791:S
20792:D   -16  2 var
```

```
20793:D   -24  2   r : real;
20794:S
20795:C        2 begin
20796:C        2   ck_system_init;
20797:C        2   ck_display_init;
20798:S
20799:C        2   if (dx = 0) and (dy = 0) then error (err_bad_parms);
20800:S
20801:C        2   with gcb^ do
20802:C        3     begin
20803:C        3       r := sqrt ( dx*dx + dy*dy);
20804:C        3       char_rot_w := dx / r;
20805:C        3       char_rot_h := dy / r;
20806:C        3       with gle_gcb^ do
20807:C        4         begin
20808:C        4           info1 := trunc(char_rot_w * 32768);
20809:C        4           info2 := trunc(char_rot_h * 32768);
20810:C        4         end;
20811:C        3       gle_text_dir ( gle_gcb );
20812:C        3     end
20813:C        3 end; { set_text_rot }
20814:S
20815:S
20816:D   -16  1 procedure set_char_size (width,height:real);
20817:S
20818:D   -16  2 { PURPOSE : To set the new character size                                    }
20819:S
20820:D   -16  2 { save width and height, set flag so text xform is recalculated         }
20821:S
20822:C        2 begin
20823:C        2   ck_system_init;
20824:C        2   ck_display_init;
20825:S
20826:C        2   (if (width = 0.0) or (height = 0.0) then error (err_bad_parms);)
20827:S
20828:C        2   with gcb^ do
20829:C        3     begin
20830:C        3       dgl_char_width := width;
20831:C        3       dgl_char_height := height;
20832:C        3       with gle_gcb^ do
20833:C        4         begin
20834:C        4           info1 := abs(trunc((width * xwtod_scale * 7 / 9) * 8));
20835:C        4           info2 := abs(trunc((height * ywtod_scale * 9 / 15) * 8));
20836:C        4           gle_char_size ( gle_gcb );
20837:S
20838:C        4           info1 := abs(trunc((width * xwtod_scale * 2 / 9) * 8));
20839:C        4           info2 := abs(trunc((height * ywtod_scale * 6 / 15) * 8));
20840:C        4           gle_text_spacing ( gle_gcb );
20841:C        4         end;
20842:C        3     end
20843:C        3 end; { set_char_size }
20844:S
20845:S
20846:D   -16  1 procedure set_echo_pos (wx,wy : real);
20847:S
20848:D   -16  2 { Purpose : To set the locator echo position                              }
20849:S
20850:D   -24  2   function between ( x1, p, x2 : real ) : boolean; { added for 2.1 bug fix }
20851:S
20852:C        3   begin
```



```
20853:C        3     between := (((x1 <= p) and ( p <= x2)) or
20854:C        3                 ((x2 <= p) and ( p <= x1)));
20855:C        3   end;
20856:S
20857:C        2 begin
20858:C        2   ck_system_init;
20859:C        2   ck_display_init;
20860:C        2   ck_locator_init;
20861:S
20862:C        2   with gcb^ do
20863:C        3     with window_lim do
20864:C        4       begin
20865:C        4         { ck bounds }
20866:C        4         if (between (xmin,wx,xmax) and            { 2.1 bug fix }
20867:C        5             between (ymin,wy,ymax)) then
20868:C        5           begin
20869:C        5             { set world coord echo pos }
20870:C        5             w_loc_echo_x := wx;
20871:C        5             w_loc_echo_y := wy;
20872:S
20873:C        5             { convert to display units }
20874:C        5             convert_wtod (w_loc_echo_x,w_loc_echo_y,d_loc_echo_x,d_loc_echo_y);
20875:C        5           end
20876:C        5         { ignor call if outside window }
20877:C        5       end
20878:C        4 end; { set_echo_pos }
20879:S
20880:D        1 procedure set_locator_lim (    lxmin,lxmax,lymin,lymax : real;
20881:D   -32  2                             var ierr : integer              );
20882:S
20883:S
20884:D   -32  2 { Purpose : To set the locator echo position                              }
20885:S
20886:D   -32  2 var
20887:D   -40  2   txmin : real;
20888:D   -48  2   txmax : real;
20889:D   -56  2   tymin : real;
20890:D   -64  2   tymax : real;
20891:S
20892:C        2 begin
20893:C        2   ck_system_init;
20894:C        2   ck_locator_init;
20895:S
20896:S          { input limits can only be changed if the input device is not the same
20897:C        2   physical device as the display                                      }
20898:C        2   with gcb^ do
20899:C        3     if not disp_eq_loc then
20900:C        4       begin
20901:C        4         { ck parms }
20902:C        4         if (lxmin < lxmax) and (lymin < lymax) then
20903:C        5           with gle_gcbi^ do
20904:C        6             with max_loc_lim do
20905:C        7               begin
20906:C        7               { convert limits form mil to locator cord }
20907:C        7                 txmin := ((lxmin * input_res_x) + xmin);
20908:C        7                 txmax := ((lxmax * input_res_x) + xmin);
20909:C        7                 tymin := ((lymin * input_res_y) + ymin);
20910:C        7                 tymax := ((lymax * input_res_y) + ymin);
20911:S
20912:S                           { make sure new logical imits are within the physical
```

```
20913:C        7                    limits                                                  }
20914:C        7                  if (txmin >= xmin) and
20915:C        8                    (txmax <= xmax + eight_diget_epsilon) and
20916:C        8                    (tymin >= ymin) and
20917:C        8                    (tymax <= ymax + eight_diget_epsilon) then
20918:C        8                    begin
20919:C        8                      { set new limits }
20920:C        8                      locator_limits (txmin,txmax,tymin,tymax);
20921:C        8                      ierr := 0;
20922:C        8                    end
20923:C        8                  else { bad limits }
20924:C        8                    ierr := 2;
20925:C        7                end
20926:C        7          else { bad parms }
20927:C        5            ierr := 1;
20928:C        4        end
20929:C        4      else { locator and display are same device }
20930:C        4        ierr := 3;
20931:C        2 end; { set_locator_lim }
20932:S
20933:S
20934:D        1 procedure sample_locator(    echo   : integer;
20935:D        2                          var rx,ry  : real    );
20936:S
20937:D        2 { Purpose : To sample the locator device                                    }
20938:S
20939:C        2 begin
20940:C        2   if loc_init then
20941:C        3     begin
20942:C        3       if disp_init then make_pic_current;
20943:C        3       call (gcb^.proc_sample_locator,echo,rx,ry);
20944:C        3     end
20945:C        3   else
20946:C        3     if system_init then error(err_loc_int)
20947:C        4     else error(err_sys_int);
20948:S
20949:C        2 end; { sample_locator }
20950:S
20951:D        1 procedure await_locator(    echo   : integer;
20952:D        2                         var button : integer;
20953:D        2                         var rx,ry  : real    );
20954:S
20955:S
20956:D        2 { Purpose : To activate the locator, and wait for operator termination      }
20957:S
20958:S
20959:D        2 var
20960:D        2   saved_pattern,
20961:D        2   saved_linewidth,
20962:D        2   saved_linestyle,
20963:D        2   saved_length,
20964:D        2   saved_mode,
20965:D   -24  2   saved_drawing_mode : integer;
20966:S
20967:C        2 begin
20968:C        2   ck_system_init;
20969:C        2   ck_locator_init;
20970:C        2   if disp_init then
20971:C        3     with gcb^,gle_gcb^ do
20972:C        4       begin
```



```
20973:C        4         make_pic_current;
20974:C        4         saved_pattern      := current_linestyle_pattern;
20975:C        4         saved_linewidth    := current_linewidth;
20976:C        4         saved_linestyle    := current_linestyle;
20977:C        4         saved_length       := current_pattern_length;
20978:C        4         saved_mode         := current_linestyle_mode;
20979:C        4         saved_drawing_mode := current_drawing_mode;
20980:C        4
20981:C        4         info1 := 0;
20982:C        4         info2 := 0;
20983:C        4         info3 := 0;
20984:C        4         info4 := -1;
20985:C        4         gle_linestyle ( gle_gcb );
20986:S
20987:C        4         info1 := 1;
20988:C        4         gle_linewidth ( gle_gcb );
20989:S
20990:C        4         call (proc_color,cursor_color);
20991:C        4
20992:C        4         info1 := complement_mode;
20993:C        4         gle_define_drawing_mode ( gle_gcb );
20994:C        4       end;
20995:S
20996:C        2   call (gcb^.proc_await_locator,echo,button,rx,ry);
20997:S
20998:C        2   if disp_init then
20999:C        3     begin
21000:C        3       with gcb^,gle_gcb^ do
21001:C        4         begin
21002:C        4           info1 := saved_linestyle;
21003:C        4           info2 := saved_length;
21004:C        4           info3 := saved_mode;
21005:C        4           info4 := saved_pattern;
21006:C        4           gle_linestyle ( gle_gcb );
21007:S
21008:C        4           info1 := saved_linewidth;
21009:C        4           gle_linewidth ( gle_gcb );
21010:S
21011:C        4           info1 := saved_drawing_mode;
21012:C        4           gle_define_drawing_mode ( gle_gcb );
21013:S
21014:C        4           call (proc_color,dgl_current_color);
21015:C        4         end;
21016:S
21017:C        3       if echo > 1 then
21018:C        4         if not int_cp then move ( world_real_cpx, world_real_cpy )
21019:C        5         else               int_move ( world_int_cpx, world_int_cpy );
21020:C        3     end;
21021:S
21022:C        2 end; { await_locator }
21023:S
21024:D        1 procedure display_term;
21025:S
21026:D        2 { Purpose:  To terminate the graphics display                               }
21027:S
21028:C        2 begin
21029:C        2   ck_system_init;
21030:C        2   ck_display_init;
21031:S
21032:C        2   with gcb^ do
```

```
21033:C        3     begin
21034:C        3       disp_init := false;
21035:C        3       disp_file_name := '';
21036:C        3       disp_dev_adr := 0;
21037:S
21038:C        3       { reset display limits to default }
21039:S
21040:C        3       with init_display_lim do
21041:C        4         display_limits(xmin,xmax,ymin,ymax);
21042:S
21043:C        3       if disp_eq_loc then with def_loc_lim do   { locator limits no longer }
21044:C        5         locator_limits (xmin,xmax,ymin,ymax);   { are effected by the display }
21045:S
21046:C        3       disp_eq_loc := not loc_init;   { if both disabled then they are equal }
21047:S
21048:C        3       try
21049:C        4         call (gcb^.proc_color,0);
21050:C        4         gle_flush_buffer ( gle_gcb );
21051:S                   gle_get_p1p2 ( gle_gcb );      { Force read from device.  This syncs
21052:C        4                                            the OS with buffered devices    }
21053:C        4         gle_term(gle_gcb);
21054:C        4       recover
21055:C        4         { ignore timeout errors }
21056:C        4         if (escapecode <> -26) or (ioe_result <> 17) then escape(escapecode);
21057:S
21058:C        3       { dispose of color table and polygon table }
21059:S                 {if color_table_size > 0 then hpm_dispose(color_table_ptr,
21060:S                                           (color_table_size+1) * 24);
21061:S                 if number_polygon_styles > 0 then hpm_dispose(poly_table_ptr,
21062:C        3                                     number_polygon_styles * 18 );}
21063:C        3     end;
21064:S
21065:C        2 end; { display_term }
21066:S
21067:D        1 procedure locator_term;
21068:S
21069:D        2 { Purpose:  To terminate the locator device                              }
21070:S
21071:C        2 begin
21072:C        2   ck_system_init;
21073:C        2   ck_locator_init;
21074:S
21075:C        2   loc_init := false;
21076:C        2   with gcb^ do
21077:C        3     begin
21078:C        3       disp_eq_loc := not disp_init;   { if both disabled then they are equal }
21079:C        3       loc_dev_adr := 0;
21080:C        3     end;
21081:S
21082:C        2   { reset to default locator limits }
21083:C        2   with init_locator_lim do
21084:C        3     locator_limits (xmin,xmax,ymin,ymax);
21085:S
21086:C        2   gle_input_term(gle_gcbi);
21087:S
21088:C        2 end; { locator_term }
21089:S
21090:D        1 procedure graphics_term;
21091:S
21092:D        2 { Purpose:  To terminate the graphics system                             }
```



```
21093:S
21094:C        2 begin
21095:C        2   ck_system_init;
21096:S
21097:C        2   { make sure all devices are terminated }
21098:C        2   if disp_init then display_term;
21099:C        2   if loc_init then locator_term;
21100:S
21101:C        2   { return dynamic vars to system }
21102:S             {dispose(gcb);
21103:S             dispose(gle_gcb);
21104:S             dispose(gle_gcbi);
21105:C        2    dispose(gle_knob_echo_gcb);}
21106:S
21107:C        2   { set system initialized flag to disabled }
21108:C        2   system_init := false;
21109:C        2 end; { graphics_term }
21110:S
21111:D        1 procedure setup_display ( var ierr : integer);
21112:S
21113:D        2 var
21114:D    -4  2   i : integer;
21115:S
21116:D    -4  2   { Purpose:  To set up display state after it has been initialized     }
21117:S
21118:C        2 begin
21119:C        2   with gcb^ do
21120:C        3     begin
21121:C        3       with gle_gcb^ do
21122:C        4         begin
21123:C        4           max_disp_lim.xmin := display_min_x;
21124:C        4           max_disp_lim.xmax := display_max_x;
21125:C        4           max_disp_lim.ymin := display_min_y;
21126:C        4           max_disp_lim.ymax := display_max_y;
21127:S
21128:C        4           gle_get_p1p2 ( gle_gcb );
21129:S
21130:C        4           def_disp_lim.xmin := info1;
21131:C        4           def_disp_lim.xmax := info2;
21132:C        4           def_disp_lim.ymin := info3;
21133:C        4           def_disp_lim.ymax := info4;
21134:C        4         end;
21135:S
21136:C        3       disp_init := true;
21137:C        3       disp_eq_loc :=  ((disp_dev_adr = loc_dev_adr) or
21138:C        3                       ((disp_dev_adr = internal_display) and
21139:C        3                        (loc_dev_adr = internal_locator)));
21140:S
21141:C        3       { set up display limits }
21142:S
21143:C        3       with def_disp_lim do
21144:C        4         display_limits(xmin,xmax,ymin,ymax);
21145:S
21146:C        3       { set up default text size and rotation attributes }
21147:S
21148:C        3       dgl_char_width := init_char_width_factor *
21149:C        3         abs (window_lim.xmax - window_lim.xmin);
21150:C        3       dgl_char_height := init_char_height_factor *
21151:C        3         abs (window_lim.ymax - window_lim.ymin);
21152:C        3       set_char_size ( dgl_char_width, dgl_char_height );
```

```
21153:S
21154:C         3       char_rot_w := init_char_rot_w;
21155:C         3       char_rot_h := init_char_rot_h;
21156:S
21157:C         3       set_text_rot ( char_rot_w, char_rot_h );
21158:S
21159:C         3       { set up all attributes here        }
21160:S
21161:C         3       dgl_current_polygon_edge := true;
21162:C         3       dgl_current_polygon_crosshatch := false;
21163:C         3       dgl_current_polygon_linestyle := init_linestyle;
21164:C         3       dgl_current_polygon_style := 1;
21165:C         3       dgl_current_polygon_color := init_color;
21166:C         3       dgl_polygon_color_current := false;  { color not set in gle }
21167:C         3       dgl_current_polygon_density := 0;
21168:C         3       dgl_current_polygon_angle := 0;
21169:C         3       set_timing ( dgl_current_timming_mode );
21170:C         3       dgl_current_color := -1; { force calc of color }
21171:C         3       set_color(init_color);
21172:C         3       set_line_style(init_linestyle);
21173:C         3       set_line_width(init_linewidth);
21174:S
21175:C         3       { init_cpy is in device units }
21176:C         3       cpx := init_cpx;
21177:C         3       cpy := init_cpy;
21178:S
21179:C         3       with gle_gcb^ do
21180:C         4         begin
21181:C         4           marker_size_x := trunc(display_res_x * 2.5 + 0.5); { markers are 2.5 mm in size }
21182:C         4           marker_size_y := marker_size_x;
21183:C         4           info1 := marker_size_x;
21184:C         4           info2 := marker_size_y;
21185:C         4           gle_marker_size ( gle_gcb );
21186:S
21187:C         4           info1 := 1;
21188:C         4           gle_graphics_on_off ( gle_gcb );  { make sure graphics is on }
21189:C         4         end;
21190:C         3     end;
21191:C         2 end; { setup_display }
21192:S
21193:D         1 procedure display_finit (   fname : gstring255;
21194:D         2                             device_name : gstring255;
21195:D         2                             control : integer;
21196:D  -512   2                         var ierr : integer );
21197:S
21198:D  -512   2 { Purpose:  To initialize the display device                              }
21199:S
21200:D  -512   2 var
21201:D  -516   2   cnt : integer;
21202:S
21203:C         2 begin
21204:C         2   ck_system_init;
21205:S
21206:C         2   { make sure no display is currently enabled }
21207:C         2   if disp_init then display_term;
21208:S
21209:C         2   if strlen(strrtrim(strltrim(device_name))) <> 0 then
21210:C         3     with gle_gcb^ do
21211:C         4       begin
21212:C         4         device_info := addr(fname[1]);
```



```
21213:C         4         device_info_char_count := strlen(fname);
21214:C         4         spooling := 1;
21215:C         4         display_name := '      ';
21216:C         4         display_name_char_count := min(strlen(device_name),6);
21217:C         4         for cnt := 1 to display_name_char_count do
21218:C         5           display_name [cnt] := device_name[cnt];
21219:C         4         info1 := control;
21220:C         4         info2 := 0; { config DGL stuff }
21221:C         4         configure_gle (gle_gcb);
21222:C         4         ierr := error_return;
21223:C         4       end
21224:C         4   else
21225:C         3     ierr := 2;
21226:S
21227:C         2   if ierr = 0 then
21228:C         3     try
21229:C         4       with gcb^ do
21230:C         5         begin
21231:C         5           disp_dev_adr := -1;     { indicate file name }
21232:C         5           disp_file_name := fname;
21233:C         5           setup_display ( ierr );
21234:C         5         end;
21235:C         4     recover
21236:C         4       begin
21237:C         4         if escapecode = -20 then escape(escapecode); { ignor all errors except stop key }
21238:C         4         ierr := 2;
21239:C         4       end
21240:C         4   else
21241:C         3     ierr := 2;
21242:S
21243:C         2 end; { display_finit }
21244:S
21245:D         1 procedure display_init (    dev_adr : integer;
21246:D         2                             control : integer;
21247:D         2                         var ierr : integer );
21248:S
21249:D         2 { Purpose:  To initialize the display device                              }
21250:S
21251:D         2 var
21252:D   -12   2   s : string[10];
21253:D   -16   2   cnt : integer;
21254:S
21255:C         2 begin
21256:C         2   ck_system_init;
21257:S
21258:C         2   { make sure no display is currently enabled }
21259:C         2   if disp_init then display_term;
21260:S
21261:C         2   with gle_gcb^ do
21262:C         3     begin
21263:C         3       s := '';
21264:C         3       strwrite(s,1,cnt,dev_adr:0);
21265:C         3       device_info_char_count := strlen(s);
21266:C         3       device_info := addr(s[1]);
21267:C         3       spooling := 0;
21268:C         3       info1 := control;
21269:C         3       info2 := 0; { config DGL stuff }
21270:C         3       configure_gle (gle_gcb);
21271:C         3       ierr := error_return;
21272:C         3     end;
```

```
21273:S
21274:C        2   if ierr = 0 then
21275:C        3     try
21276:C        4       with gcb^ do
21277:C        5         begin
21278:C        5           disp_dev_adr := dev_adr;
21279:C        5           disp_file_name := '';
21280:C        5         end;
21281:C        4       setup_display ( ierr );
21282:C        4
21283:C        4       if gle_gcb^.complement_support = 1
21284:C        5         then gle_gcb^.cursor := dgl_cursor;
21285:C        4     recover
21286:C        4       begin
21287:C        4         if escapecode = -20 then escape(escapecode); { ignor all errors exect stop key }
21288:C        4         ierr := 2;
21289:C        4       end
21290:C        4   else
21291:C        3     ierr := 2;
21292:S
21293:C        2 end; { display_init }
21294:S
21295:D        1 procedure locator_init (    dev_adr : integer;
21296:D        2                         var ierr    : integer );
21297:S
21298:D        2 { Purpose:  To initialize the locator device                                }
21299:S
21300:D        2 var
21301:D   -12  2   s : string[10];
21302:D   -16  2   i : integer;
21303:S
21304:C        2 begin
21305:C        2   ck_system_init;
21306:S
21307:C        2   { make sure no locator is enabled }
21308:C        2   if loc_init then locator_term;
21309:S
21310:C        2   { try to init a locator }
21311:S
21312:C        2   if disp_init then make_pic_current;
21313:S
21314:C        2   with gcb^,gle_gcbi^ do
21315:C        3     begin
21316:C        3       s := '';
21317:C        3       strwrite(s,1,i,dev_adr:0);
21318:C        3       device_info_char_count := strlen(s);
21319:C        3       device_info := addr(s[1]);
21320:C        3       info1 := 0;                          { init sample loc value }
21321:C        3       info2 := 0;
21322:C        3       configure_input_gle ( gle_gcbi );
21323:C        3       ierr := error_return;
21324:S
21325:C        3       if ierr = 0 then
21326:C        4         begin
21327:C        4           loc_init := true;
21328:C        4           loc_dev_adr := dev_adr;
21329:S
21330:C        4           with gle_gcbi^,gcb^.max_loc_lim do
21331:C        5             begin
21332:C        5               xmin := input_min_x;
```



```
21333:C        5               xmax := input_max_x;
21334:C        5               ymin := input_min_y;
21335:C        5               ymax := input_max_y;
21336:C        5             end;
21337:S
21338:C        4           gle_get_input_p1p2 ( gle_gcbi );
21339:S
21340:C        4           with gle_gcbi^,gcb^,gcb^.def_loc_lim do
21341:C        5             begin
21342:C        5               xmin := info1;
21343:C        5               xmax := info2;
21344:C        5               ymin := info3;
21345:C        5               ymax := info4;
21346:C        5             end;
21347:S
21348:C        4           disp_eq_loc := disp_init and
21349:C        4                          ((disp_dev_adr = loc_dev_adr) or
21350:C        4                          ((disp_dev_adr = internal_display) and
21351:C        4                          (loc_dev_adr = internal_locator)));
21352:S
21353:S                   { If locator is not the same physical device as the graphics display,
21354:S                     then the locator limits are set to the default locator limits.
21355:S                     If the locator is the same device, then the locator limits
21356:C        4            are set to the current display limits.                          }
21357:S
21358:C        4           with gcb^ do
21359:C        5             begin
21360:C        5               if disp_eq_loc then
21361:C        6                 with cur_disp_lim do locator_limits (xmin,xmax,ymin,ymax)
21362:C        7               else
21363:C        6                 with def_loc_lim do locator_limits (xmin,xmax,ymin,ymax);
21364:C        5             end;
21365:C        4         end
21366:C        4       else
21367:C        4         ierr := 2;
21368:C        3   end;
21369:C        2 end; { locator_init }
21370:S
21371:S
21372:D        1 procedure graphics_init;
21373:S
21374:D        2 { Purpose:  To initialize the graphics system                               }
21375:S
21376:C        2 begin
21377:C        2   { make sure the system is not already init }
21378:C        2   if system_init then graphics_term;
21379:S
21380:C        2   { set state flags }
21381:C        2   system_init := true;
21382:C        2   disp_init   := false;
21383:C        2   loc_init    := false;
21384:S
21385:S
21386:C        2   { get storage space -- changed from dynamic to global 2/84 BDS }
21387:C        2   gcb := addr(gcb_space);             { DGL high level storage }
21388:S
21389:C        2   gle_gcb := addr(gle_gcb_space);     { display output device }
21390:C        2   gle_init_gcb ( gle_gcb );
21391:C        2
21392:C        2   gle_gcbi := addr(gle_gcbi_space);   { locator input device  }
```

```
21393:C         2   gle_init_input_gcb ( gle_gcbi );
21394:S
21395:C         2   gle_knob_echo_gcb := addr(gle_knob_echo_gcb_space);
21396:C         2                                    {knob echo output device (internal crt)}
21397:C         2   gle_init_gcb ( gle_knob_echo_gcb );
21398:S
21399:S             { kbdlangjumper is imported from sysglobales.  Kata becomes true if
21400:C         2     the kata keyboard is instaled                                    }
21401:C         2   if kbdlang = katakana_kbd then gle_gcb^.kata := 1
21402:C         3   else gle_gcb^.kata := 0;
21403:S
21404:C         2   { set up defaults }
21405:C         2   with gcb^ do
21406:C         3     begin
21407:C         3       { When first inited that display and locator are the same device }
21408:C         3       disp_eq_loc := true;
21409:C         3       disp_dev_adr := init_dev_adr;
21410:C         3       disp_file_name := '';
21411:C         3       loc_dev_adr  := init_dev_adr;
21412:C         3       window_lim   := init_window;
21413:C         3       aspect_ratio := init_aspect;
21414:C         3       cur_vir_lim  := init_vir_lim;
21415:C         3       viewport_lim := init_viewport;
21416:S
21417:C         3       { setup the default display/ locator limits to some large number }
21418:S
21419:C         3       with init_display_lim do
21420:C         4         display_limits ( xmin, xmax, ymin, ymax );
21421:C         3       with init_locator_lim do
21422:C         4         locator_limits ( xmin, xmax, ymin, ymax );
21423:S
21424:C         3       { explicitly set the cp to init_value }
21425:S
21426:C         3       cpx := init_cpx;
21427:C         3       cpy := init_cpy;
21428:C         3       int_cp := true;
21429:C         3       world_int_cpx := init_cpx;
21430:C         3       world_int_cpy := init_cpy;
21431:S
21432:C         3       { set up default text size and rotation attributes }
21433:S
21434:C         3       dgl_char_width := init_char_width;
21435:C         3       dgl_char_height := init_char_height;
21436:S
21437:C         3       char_rot_w := init_char_rot_w;
21438:C         3       char_rot_h := init_char_rot_h;
21439:S
21440:C         3       { set flag, indicating that the text xform needs to be recalulated      }
21441:S
21442:C         3       calc_text_xform := true;
21443:S
21444:C         3       dgl_current_color := init_color;
21445:C         3       dgl_current_linestyle := init_linestyle;
21446:C         3       dgl_current_linewidth := init_linewidth;
21447:C         3       dgl_current_timming_mode := init_timming_mode;
21448:C         3       cursor_color := init_color;
21449:C         3       disp_just := centered;
21450:C         3       display_echo_mult := 1;
21451:C         3       graphics_error := 0;
21452:S
```



```
21453:C         3       number_polygon_styles := default_poly_table_size;
21454:C         3       color_table_size := default_color_table_size;
21455:C         3       dgl_current_polygon_color := 1;
21456:C         3       dgl_current_polygon_linestyle := 1;
21457:C         3       dgl_current_polygon_density := 0;
21458:C         3       dgl_current_polygon_angle := 90;
21459:C         3       dgl_current_polygon_edge := true;
21460:C         3       dgl_current_polygon_crosshatch := false;
21461:C         3       dgl_current_polygon_style := 1;
21462:C         3       dgl_current_color_model := 1;
21463:C         3     end;
21464:C         2 end; { graphics_init }
21465:S
21466:C         1 end. {module DGL_LIB}
21467:S
21468:D         1
21469:D         1 $LIST ON$
21470:S

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

LIBRARIAN

Description

The LIBRARIAN is a utility program used to create libraries, to link modules together or to disassemble code files.

Usage

The LIBRARIAN is invoked from the main command level by pressing "L".

Requirements

SYSGLOBALS, FS, ASM, MISC, LOADER, LDR, SYSDEVS, and CI.

Notes

The LIBRARIAN is sometimes called the "Linker"; however, it is not the same as the "linking loader" found in INITLOAD.

```
    1:S                                                 (*
    2:S
    3:S           (c) Copyright Hewlett-Packard Company, 1984.
    4:S           All rights are reserved.  Copying or other
    5:S           reproduction of this program except for archival
    6:S           purposes is prohibited without the prior
    7:S           written consent of Hewlett-Packard Company.
    8:S
    9:S
   10:S                       RESTRICTED RIGHTS LEGEND
   11:S
   12:S           Use, duplication, or disclosure by the Government
   13:S           is subject to restrictions as set forth in
   14:S           paragraph (b) (3) (B) of the Rights in Technical
   15:S           Data and Computer Software clause in
   16:S           DAR 7-104.9(a).
   17:S
   18:S           HEWLETT-PACKARD COMPANY
   19:D         0 Fort Collins, Colorado                        *)
   20:S
   21:S
   22:D         0 $MODCAL,debug off,iocheck off,range off,ovflcheck off$ $ref 50$
   23:S
   24:D         0 program linker(input, output, keyboard);
   25:S
   26:S
   27:D         1 import sysglobals,fs,loader,ldr,asm,sysdevs,ci,misc;
   28:S
   29:S
   30:D         1 const   pagelines = 63;
   31:D         1         pageblocks = 2;
   32:D         1         entrysize = 26;
   33:S
   34:D         1 type    address = integer;
   35:D         1         point = ^integer;
   36:S
   37:D         1 var     keyboard:       text;
   38:D      -2 1         todaysdate:     daterec;
   39:D      -4 1         linkerdate:     daterec;
   40:D     -18 1         tempstring:     string[12];
   41:D    -100 1         gvaluestring:   string80;
   42:D    -100 1         copyright,
   43:D    -264 1         listfilename:   string80;
   44:D    -928 1         listing:        text;
   45:D    -928 1         pagenum,
   46:D    -932 1         linenum:        shortint;
   47:D    -932 1         linking,booting,
   48:D    -932 1         outopen,
   49:D    -932 1         verifying,defsout,
   50:D    -932 1         printopen,
   51:D    -939 1         printeron:      boolean;
   52:D    -940 1         commandchar: char;
   53:D    -944 1         startgvr:       addrec;
   54:D    -948 1         startgvrmod:    moddescptr;
   55:D    -952 1         modsave:        moddescptr;
   56:D    -956 1         infost:         addrec;
   57:S
   58:D    -960 1         ires:   integer;        (saved ioresult)
   59:D    -964 1         errors: integer;
   60:D    -968 1         esccode: integer;
```



```
   61:D    -976 1         lowheap0,highheap0: addrec;
   62:S
   63:D    1058 1         infilename:     string80;
   64:D    1060 1         vmodnum:        shortint;
   65:S
   66:D    1060 1         (output file information: )
   67:D    1064 1         outdirectory:   addrec;         (new library directory pointer)
   68:D    1728 1         outfile:        phyle;          (file being written)
   69:D    1728 1         firstoutblock,
   70:D    1736 1         outblock:       integer;        (next block within library to write)
   71:D    1740 1         nextblock:      integer;        (next block within module to write)
   72:D    1822 1         outfilename:    string80;
   73:D    1826 1         outmodnum:      integer;        (number of modules created so far)
   74:D    1826 1         outdirectsize,
   75:D    1834 1         maxmodules:     integer;
   76:S
   77:D    1834 1         (linker information:  )
   78:D    1838 1         totalpatchspace: integer;       (bytes of patch space)
   79:D    1842 1         patchbytes:     integer;
   80:D    1842 1         backwardpatches,
   81:D    1850 1         forwardpatches: moddescptr;
   82:D    1854 1         newmodname:     addrec;         (new name for linked module)
   83:D    1858 1         infostart:      addrec;         (pointer to bottom of linker memory)
   84:D    1862 1         newexttable:    address;        (location of new EXT table)
   85:D    1866 1         newextsize:     integer;        (size in bytes of EXT table)
   86:D    1870 1         newdirectory:   addrec;         (pointer to new module directory)
   87:D    1874 1         loadgvr:        gvrptr;
   88:S
   89:D         1 procedure dobeep;
   90:C         2 begin beep;if streaming then escape(-1); end;
   91:S
   92:D         1 procedure errorline;
   93:C         2 begin ires := ioresult; fgotoxy(output, 0, 22); write(bellchar, cteol); end;
   94:S
   95:D         1 procedure ioerror;
   96:C         2 begin write(' ioresult = ',ires:1); escape(123); end;
   97:S
   98:D      -82 1 procedure getcommandchar(s:string80; var c:char);
   99:C         2 begin
  100:C         2   fgotoxy(output,0,23); write(s,cteol);
  101:C         2   read(keyboard,c);
  102:C         2   fgotoxy(output,0,22); writeln(cteol);write(cteol);
  103:C         2   if (c>='a') and (c<='z') then c:= chr(ord(c)-32);
  104:C         2 end;
  105:S
  106:D         1 procedure writedate(var f: text; date: daterec);
  107:D         2 type months = packed array[0..35] of char;
  108:D         2 const monthname = months['JanFebMarAprMayJunJulAugSepOctNovDec'];
  109:D      -4 2 var i,j: shortint;
  110:C         2 begin
  111:C         2 with date do
  112:C         3   begin
  113:C         3   if (month in [1..12]) and (day>0)
  114:C         4   and (year<100) then
  115:C         4     begin { Valid date }
  116:C         4     write(f, day:2, '-');
  117:C         4     j := (month - 1) * 3;
  118:C         4     for i := j to j+2 do write(f, monthname[i]);
  119:C         4     write(f, '-',year:2);
  120:C         4     end
```

```
   121:C        4   else write(f, '<no date>'); { Invalid date }
   122:C        3   end;
   123:C        2 end; {datestring}
   124:S
   125:D        1 procedure gbytes(var p: integer; size: integer);
   126:C        2 begin
   127:C        2  p := lowheap.a;        lowheap.a := lowheap.a + size;
   128:C        2  if lowheap.a > highheap.a then escape(112);
   129:C        2 end;
   130:S
   131:D        1 procedure blockwrite(anyvar f: fib; anyvar obj: window; blocks,block: integer);
   132:C        2 begin
   133:C        2   call (f.am, addr(f), writebytes, obj, blocks*fblksize, block*fblksize);
   134:C        2   if ioresult <> ord(inoerror) then escape(113);
   135:C        2 end;
   136:S
   137:D        1 procedure readblocks(var f: fib; anyvar obj: window; size, block: integer);
   138:C        2 begin
   139:C        2   call (f.am, addr(f), readbytes, obj, size, block*fblksize);
   140:C        2   if ioresult <> ord(inoerror) then escape(-10);
   141:C        2 end;
   142:S
   143:D        1 procedure gvrstring(var gvp:gvrptr; var val:integer; pcrel,nores: boolean);
   144:S
   145:S          (*advances g past the GVR, adds any absolute part to VAL,
   146:D        2   and constructs a string representing the GVR in gvaluestring          *)
   147:S
   148:D        2 type
   149:D        2   rpp = ^referenceptr;
   150:D        2 var
   151:D    -2  2   Rcount:       shortint;
   152:D    -3  2   done:         boolean;
   153:D    -8  2   g:            gvrptr;
   154:D   -12  2   i:            integer;
   155:S
   156:D        2 procedure sign(sub: boolean);
   157:C        3 begin
   158:C        3  if sub then sappend(gvaluestring,'-')
   159:C        4  else if strlen(gvaluestring)>0 then sappend(gvaluestring,'+');
   160:C        3 end;
   161:S
   162:C        2 begin {gvrstring}
   163:C        2  gvaluestring := '';       Rcount := 0;
   164:C        2  repeat
   165:C        3  if pcrel then g := loadgvr
   166:C        4          else g := gvp;
   167:C        3  if g <> NIL then with g^ do
   168:C        5   begin
   169:C        5   if longoffset then
   170:C        6     g:=gvrptr(integer(g)+sizeof(generalvalue,true))
   171:C        6   else
   172:C        6     g:=gvrptr(integer(g)+sizeof(generalvalue,false));
   173:C        5   if valueextend then
   174:C        6     begin
   175:C        6     if not pcrel then val:= val + veptr(g)^.value;
   176:C        6     g:=gvrptr(integer(g)+sizeof(valueextension,sint));
   177:C        6     end;
   178:C        5   case primarytype of
   179:C        6    absolute: {no more value};
   180:C        6    relocatable: Rcount := Rcount + 1;
```



```
   181:C        6    global: begin sign(false); sappend(gvaluestring,'Gbase'); end;
   182:C        6    general:
   183:C        6     begin
   184:C        6     done := false;
   185:C        6     repeat with rpp(g)^ do
   186:C        8       begin
   187:C        8        if adr=0 then
   188:C        9          if op=addit then Rcount := Rcount + 1
   189:C       10          else Rcount := Rcount - 1
   190:C       10        else if adr=1 then
   191:C       10          begin sign(op=subit); sappend(gvaluestring,'Gbase'); end
   192:C       10        else
   193:C       10          begin sign(op=subit);
   194:C       10          if newmods^.unresbits.bmp^[adr] or nores
   195:C       11              then sappend(gvaluestring,symbolptr(newexttable+4*adr)^)
   196:C       11              else sappend(gvaluestring,symbolptr(point(newexttable+4*adr)^)^);
   197:C       10          end;
   198:C        8       done := last;
   199:C        8       g := gvrptr(integer(g)+sizeof(referenceptr));
   200:C        8       end;
   201:C        7     until done;
   202:C        6     end; {general}
   203:C        6    end; {primarytype cases}
   204:C        5   if not pcrel then gvp := g;
   205:C        5   end; {with g^}
   206:C        3  pcrel := not pcrel;
   207:C        3  until pcrel;
   208:C        2  while Rcount <> 0 do
   209:C        3   begin sign(Rcount<0); sappend(gvaluestring,'Rbase');
   210:C        3         if Rcount < 0 then Rcount := Rcount + 1
   211:C        4         else Rcount := Rcount - 1;
   212:C        3   end;
   213:C        2  if (val <> 0) or (strlen(gvaluestring)=0) then
   214:C        3   begin
   215:C        3   if val >= 0 then sign(false);
   216:C        3   strwrite(gvaluestring,strlen(gvaluestring)+1,i,val:1);
   217:C        3   end;
   218:C        2 end; {gvrstring}
   219:S
   220:S
   221:D        1 procedure printheader(var f: text);
   222:D    -4  2 var time: timerec;
   223:C        2 begin
   224:C        2  write(f,'Librarian  [Rev.  3.0 ');
   225:C        2  if ioresult <> 0 then
   226:C        3    begin
   227:C        3    printopen := false;
   228:C        3    printeron := false;
   229:C        3    escape(118);
   230:C        3    end;
   231:C        2  writedate(f, linkerdate);
   232:C        2  write(f,']',' ':7);
   233:C        2  writedate(f, todayсdate);
   234:C        2  systime(time);
   235:C        2  with time do write(f, hour:4,':',minute:2,':',centisecond div 100:2);
   236:C        2  if pagenum > 0 then write(f,'page ':10,pagenum:1);
   237:C        2  writeln(f);
   238:C        2  writeln(f);
   239:C        2 end;
   240:S
```

```
 241:D        1 procedure pageeject;
 242:D    -4  2 var i: integer;
 243:C        2 begin
 244:C        2  if linenum > 0 then page(listing);
 245:C        2  linenum := 0;
 246:C        2 end;
 247:S
 248:D        1 procedure list;
 249:C        2 begin
 250:C        2   if linenum >= pagelines then pageeject;
 251:C        2   if linenum = 0 then
 252:C        3     begin
 253:C        3     pagenum := pagenum + 1;
 254:C        3     printheader(listing);
 255:C        3     linenum := linenum + 2;
 256:C        3     end;
 257:C        2   linenum := linenum + 1;
 258:C        2 end;
 259:S
 260:D        1 procedure listln;
 261:C        2 begin writeln(listing); linenum := linenum + 1;
 262:C        2 end;
 263:S
 264:D        1 procedure quit;
 265:D    -1  2 var ch: char;
 266:C        2 begin
 267:C        2   if (outopen and (outmodnum>0)) or
 268:C        3      (booting and (outblock>0)) then
 269:C        3   begin
 270:C        3   errorline;
 271:C        3   if booting then writeln('WARNING:  You didn''t finish booting')
 272:C        4   else writeln('WARNING:  You didn''t ''Keep'' the output file.');
 273:C        3   if streaming then escape(123)
 274:C        4   else
 275:C        4     begin
 276:C        4     write('Are you sure you want to quit?  (type Y if yes)  ');
 277:C        4     read(keyboard, ch);
 278:C        4     if (ch<>'y') and (ch<>'Y') then commandchar := ' ';
 279:C        4     end;
 280:C        3   end;
 281:C        2 end;
 282:S
 283:D        1 function readint(var value: integer): boolean;
 284:D   -82  2 var s: string80;
 285:D   -86  2     i: integer;
 286:C        2 begin
 287:C        2   readln(s);
 288:C        2   strread(s,1,i,value);
 289:C        2   if ioresult<>ord(inoerror) then
 290:C        3     if i <= strlen(s) then escape(124);
 291:C        2   readint := ioresult=ord(inoerror);
 292:C        2 end;
 293:S
 294:D        1 procedure unassemble;
 295:S
 296:D        2 type hex = 0..15;
 297:D        2      htoctyp = array[0..15] of char;
 298:D        2      decodestatetype = (consts,code,abscode,startcase,casetable,endofproc,quittype,notype);
 299:S
 300:D        2 const htoc = htoctyp['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'];
```



```
 301:S
 302:D        2 var
 303:D    -4  2   nilgvr:       gvrptr;
 304:D    -5  2   dumped:       boolean;
 305:D    -6  2   fortranflag:  boolean;
 306:D    -8  2   rangetype: (norange, prange, linerange);
 307:D   -20  2   lowrange, highrange, lastline: integer;
 308:D   -24  2   decodestate,oldstate: decodestatetype;
 309:D   -36  2   PC,tablePC,casecodestart: integer;
 310:D   -40  2   codecount: integer; {bytes left in current inbuf}
 311:D   -44  2   codeindex: integer; {next byte of code in inbuf}
 312:D   -46  2   instrsize: 0..10;  {byte count of current instruction}
 313:D   -58  2   refgvr: addrec;       reflim,refloc: address;
 314:D   -66  2   inbuf,refptr: addrec;
 315:S
 316:D        2 procedure dumpmod (mp:moddescptr);
 317:D    -4  3 var junkint:    integer;
 318:D   -36  3     producername: string[30];
 319:D   -40  3     textstep: addrec;
 320:S
 321:C        3 begin {dumpmod}
 322:C        3  dumped := true;
 323:C        3  with mp^,directory.drp^ do
 324:C        4   begin
 325:C        4   pageeject;
 326:C        4   textstep.a:=directory.a+sizeof(moduledirectory);
 327:C        4   list; write(listing,'MODULE    ');
 328:C        4   if strlen(textstep.syp^) = 0 then write(listing,'(no name)')
 329:C        5                                    else write(listing,textstep.syp^);
 330:C        4   write(listing,'    Created ');
 331:C        4   writedate(listing, date); writeln(listing);
 332:C        4   list; write(listing,'NOTICE:  ');
 333:C        4   if strlen(notice)=0 then writeln(listing,'(none)')
 334:C        5                           else writeln(listing,notice);
 335:C        4   fortranflag := (producer = 'F');
 336:C        4   case producer of
 337:C        5     'M': producername := 'Modcal Compiler';
 338:C        5     'P': producername := 'Pascal Compiler';
 339:C        5     'L': producername := 'Librarian';
 340:C        5     'F': producername := 'FORTRAN Compiler';
 341:C        5     'B': producername := 'BASIC Compiler';
 342:C        5     'A': producername := 'Assembler';
 343:C        5     'C': producername := ''C'' Compiler';
 344:C        5     'D': producername := 'Ada Compiler';
 345:C        5     otherwise producername := '' '';
 346:C        5               producername[2] := producer;
 347:C        5     end;
 348:C        4   list; write(listing,'  Produced by ', producername, ' of ');
 349:C        4   writedate(listing,revision); writeln(listing);
 350:C        4   if systemid = 0 then systemid := 1;
 351:C        4   list; writeln(listing,'  Revision number ',systemid:1);
 352:C        4   list; writeln(listing,'  Directory size ',directorysize:6,' bytes');
 353:C        4   list; writeln(listing,'  Module size    ',modulesize:6,' bytes');
 354:C        4   junkint:=strlen(textstep.syp^);
 355:C        4   textstep.a := textstep.a+junkint+2-ord(odd(junkint));
 356:C        4   if executable then
 357:C        5     begin
 358:C        5     startgvr := textstep;
 359:C        5     junkint := 0;
 360:C        5     gvrstring(textstep.gvp,junkint,false,false);
```

```
361:C        5     list; writeln(listing,'  Execution address    ',gvaluestring);
362:C        5     end
363:C        5   else
364:C        5     begin
365:C        5     startgvr.gvp:=NIL;
366:C        5     list; writeln(listing,'  Module NOT executable');
367:C        5     end;
368:C        4   list; writeln(listing,'  Code base      ',relocatablebase,
369:C        4                             '   Size ',relocatablesize,' bytes');
370:C        4   list; writeln(listing,'  Global base    ',globalbase,
371:C        4                             '   Size ',glcbalsize,' bytes');
372:C        4   if extsize <= 8 then extsize := 0;
373:C        4   list; writeln(listing,'  EXT    block ',extblock:3,'     Size ',extsize,
374:C        4                  ' bytes');
375:C        4   list; writeln(listing,'  DEF    block ',defblock:3,'     Size ',defsize,
376:C        4                  ' bytes');
377:C        4   list;
378:C        4   if sourcesize <> 0 then
379:C        5       writeln(listing,'  EXPORT block ',sourceblock:3,'     Size ',
380:C        5                  sourcesize,' bytes')
381:C        5   else
382:C        5     writeln(listing,'  No EXPORT text');
383:C        4   list; writeln(listing,'  There are ',textrecords,' TEXT records');
384:C        4   listln; listln;
385:C        4   end; {with mp^,directory^}
386:C        3 end; {dumpmod}
387:S
388:D        2 procedure prepunassem;
389:D    -4  3 var i: integer;
390:C        3 begin
391:C        3   modsave := newmods;
392:C        3   newmods := NIL;
393:C        3   infost  := lowheap;
394:C        3   loadinfo(vmodnum, true, true);
395:C        3   with newmods^,directory.drp^ do
396:C        4    begin
397:C        4    newexttable := extaddr.a;
398:C        4    gbytes(unresbits.a, ((extsize div 4 + 15) div 16)*2);
399:C        4    for i := 2 to extsize div 4 - 1 do unresbits.bmp^[i] := false;
400:C        4    for i := 2 to listsize-1 do unresbits.bmp^[listaddr^[i] div 4] := true;
401:C        4    end;
402:C        3   if not dumped then dumpmod(newmods);
403:C        3 end;
404:S
405:D        2 procedure nextref;
406:C        3 begin
407:C        3 if refgvr.a < reflim then
408:C        4   if refgvr.gvp^.longoffset then
409:C        5     refloc:=refloc+refgvr.gvp^.long
410:C        5   else
411:C        5     refloc:=refloc+refgvr.gvp^.short;
412:C        3 end;
413:S
414:D        2 procedure listinstruction;
415:D        3 type        regtype  = (D,A);
416:D        3             regrange = 0..7;
417:D        3             siz = (bytesiz,wordsiz,longsiz,invalid);
418:D        3             opsizetype = array[bytesiz..longsiz] of string[3];
419:S
420:D        3 const       opsize = opsizetype['.b ','.w ','.l '];
```



```
421:S
422:D        3 var
423:D   -10  3   hexout: packed array[0..4,0..3] of hex;
424:D   -11  3   firstline: boolean;  { 1st line of current instruction? }
425:D   -14  3   bytesleft: 0..10;    { to be listed ir current instr }
426:D  -270  3   instrbuf: string[255];{ alpha form of instruction }
427:S
428:D  -270  3   instr: packed record case integer of
429:D  -270  3             1: (opcode: 0..15;
430:D  -270  3                 case integer of
431:D  -270  3                   1: (cond: 0..15;
432:D  -270  3                       displ: -128..127);
433:D  -270  3                   2: (reg1: regrange;
434:D  -270  3                       opmode: 0..7;
435:D  -270  3                       eamode: 0..7;
436:D  -270  3                       eareg: regrange);
437:D  -270  3                   3: (dummy: 0..7;
438:D  -270  3                       bit8: boolean;
439:D  -270  3                       size: siz)
440:D  -270  3                );
441:D  -270  3             3: (w: shortint);
442:D  -270  3             4: (lb, rb: byte);
443:D  -272  3             end; {instr}
444:S
445:D  -272  3   ext:   packed record
446:D  -272  3           case integer of
447:D  -272  3             1: (uwordext: 0..65535);
448:D  -272  3             2: (wordext: shortint);
449:D  -272  3             3: (longext: integer);
450:D  -272  3             4: (regclass: regtype;
451:D  -272  3                 reg: regrange;
452:D  -272  3                 long: boolean;
453:D  -272  3                 dummy: 0..7;
454:D  -272  3                 byteext: -128..127);
455:D  -272  3             5: (mask: packed array [0..15] of boolean);
456:D  -276  3             end;
457:S
458:D        3   procedure emitint(val: integer);
459:D    -4  4   var i: integer;
460:C        4   begin
461:C        4     strwrite(instrbuf, strlen(instrbuf)+1, i, val:1);
462:C        4   end;
463:S
464:D        3   procedure comma;
465:C        4   begin  sappend(instrbuf, ','); end;
466:S
467:D        3   procedure space;
468:C        4   begin  sappend(instrbuf, ' '); end;
469:S
470:D        3   procedure printinstrword;
471:D    -4  4   var k: integer;
472:C        4   begin write(listing,' ');
473:C        4     for k := 0 to 3 do
474:C        5       write(listing,htoc[hexout[(instrsize-bytesleft) div 2,k]]);
475:C        4     bytesleft := bytesleft-2;
476:C        4   end;
477:S
478:D        3   procedure getinstrbytes(size: shortint);
479:C        4   begin
480:C        4   if codecount < size then escape(121);
```

```
   481:C        4   moveleft(inbuf.stp^[codeindex],ext,size);
   482:C        4   moveleft(ext,hexout[instrsize div 2],size);
   483:C        4   instrsize := instrsize+size;
   484:C        4   codeindex := codeindex+size;
   485:C        4   codecount := codecount-size;
   486:C        4   end;
   487:S
   488:D        3   procedure getinstruction;
   489:C        4   begin
   490:C        4   instrsize := 0;
   491:C        4   getinstrbytes(2);     instr.w := ext.wordext;
   492:C        4   end;
   493:S
   494:D        3   procedure defineword;
   495:C        4   begin
   496:C        4   instrbuf := 'dc.w ';
   497:C        4   with instr do
   498:C        5     begin
   499:C        5     emitint(w);
   500:C        5     while strlen(instrbuf) < 11 do space;
   501:C        5     sappend(instrbuf,'   or dc.b ');
   502:C        5     emitint(lb); comma; emitint(rb);
   503:C        5     while strlen(instrbuf) < 30 do space;
   504:C        5     sappend(instrbuf,'   or dc.b ''  ''');
   505:C        5     if (lb >= 32) and (lb < 127) then instrbuf[43] := chr(lb);
   506:C        5     if (rb >= 32) and (rb < 127) then instrbuf[44] := chr(rb);
   507:C        5     end;
   508:C        4   end;
   509:S
   510:D        3   procedure extend(size: integer; pcrel: boolean);
   511:D   -12  4     var offset, location, PCtemp: integer;
   512:C        4     begin
   513:C        4       location := PC+instrsize;
   514:C        4       if size = 1 then {byte extension}
   515:C        5         begin
   516:C        5         PCtemp := location + 1;
   517:C        5         size := 2;
   518:C        5         end
   519:C        5       else PCtemp := location;
   520:S
   521:C        4       while (refgvr.a<reflim) and (refloc<PCtemp) do
   522:C        5         begin
   523:C        5         offset := 0;
   524:C        5         gvrstring(refgvr.gvp,offset,false,false);
   525:C        5         nextref;
   526:C        5         end;
   527:S
   528:C        4       getinstrbytes(size);
   529:C        4       if odd(PCtemp)    then offset := ext.byteext
   530:C        5       else if size = 2 then offset := ext.wordext
   531:C        6                        else offset := ext.longext;
   532:C        4       if pcrel then offset := offset + location;
   533:C        4       if refloc=PCtemp then
   534:C        5         begin
   535:C        5         gvrstring(refgvr.gvp,offset,pcrel,false) ;
   536:C        5         nextref;
   537:C        5         end
   538:C        5       else
   539:C        5         gvrstring(nilgvr,offset,pcrel,false);
   540:C        4       sappend(instrbuf,gvaluestring);
```



```
   541:C        4     end;
   542:S
   543:D        3   procedure decode;
   544:D        4   label 1,2;
   545:D        4   type
   546:D        4     opndtype = (source,dest);
   547:D        4     regsymtype = array[regtype] of string[1];
   548:D        4     extsiztype = array[bytesiz..longsiz] of 1..4;
   549:D        4     arithoptype = array[8..13] of string[3];
   550:S
   551:D        4     condcodetype = array[0..15] of string[2];
   552:S
   553:D        4   const
   554:D        4     SP = 7;
   555:S
   556:D        4     regsym = regsymtype['d','a'];
   557:D        4     extsize = extsiztype[1,2,4];
   558:S
   559:D        4     condcode = condcodetype['t','f','hi','ls','cc','cs','ne','eq',
   560:D        4                             'vc','vs','pl','mi','ge','lt','gt','le'];
   561:D        4     arithop = arithoptype['or','sub',?,'cmp','and','add'];
   562:S
   563:S
   564:D        4     procedure osize;
   565:D    -2  5     var size: siz;
   566:C        5     begin size := instr.size;
   567:C        5     if size = invalid then goto 2;
   568:C        5     sappend(instrbuf, opsize[size])
   569:C        5     end;
   570:S
   571:D        4     procedure emitdir(regclass: regtype; reg: regrange);
   572:C        5     begin if (regclass = A) and (reg = SP) then sappend(instrbuf,'sp')
   573:C        6           else begin
   574:C        6                sappend(instrbuf,regsym[regclass]);
   575:C        6                setstrlen(instrbuf, strlen(instrbuf) + 1);
   576:C        6                instrbuf[strlen(instrbuf)] := htoc[reg];
   577:C        6                end;
   578:C        5     end;
   579:S
   580:D        4     procedure emitardef(reg: regrange);
   581:C        5     begin sappend(instrbuf,'(');
   582:C        5           emitdir(A,reg);
   583:C        5           sappend(instrbuf,')');
   584:C        5     end;
   585:S
   586:D        4     procedure emitardisp(reg: regrange);
   587:C        5     begin extend(2,false);
   588:C        5           emitardef(reg);
   589:C        5     end;
   590:S
   591:D        4     procedure emitpostincr(reg: regrange);
   592:C        5     begin emitardef(reg);
   593:C        5           sappend(instrbuf, '+');
   594:C        5     end;
   595:S
   596:D        4     procedure emitpredecr(reg: regrange);
   597:C        5     begin sappend(instrbuf, '-');
   598:C        5           emitardef(reg);
   599:C        5     end;
   600:S
```

```
601:D  4   procedure emitindx(pcrel: boolean);
602:C  5   begin
603:C  5   extend(1,pcrel); sappend(instrbuf, '(');
604:C  5   with instr, ext do
605:C  6     begin
606:C  6     if not pcrel then
607:C  7       begin emitdir(A, eareg); comma; end;
608:C  6     emitdir(regclass,reg);
609:C  6     if long then sappend(instrbuf,'.l)')
610:C  7           else sappend(instrbuf,'.w)');
611:C  6     end;
612:C  5   end;
613:S
614:D  4   procedure emitimm(val: integer);
615:C  5   begin
616:C  5     sappend(instrbuf,'#'); emitint(val);
617:C  5   end;
618:S
619:D  4   procedure immediate(fsize: siz);
620:C  5   begin
621:C  5     if fsize = invalid then goto 2;
622:C  5     sappend(instrbuf,'#');
623:C  5     extend(extsize[fsize],false);
624:C  5   end; {immediate}
625:S
626:D  4   procedure emitea(fsize: siz);
627:C  5   begin
628:C  5     with instr do
629:C  6       case eamode of
630:C  7     0: emitdir(D,eareg);
631:C  7     1: emitdir(A,eareg);
632:C  7     2: emitardef(eareg);
633:C  7     3: emitpostincr(eareg);
634:C  7     4: emitpredecr(eareg);
635:C  7     5: emitardisp(eareg);
636:C  7     6: emitindx(false);
637:C  7     7: case eareg of
638:C  8       0: extend(2,false);
639:C  8       1: extend(4,false);
640:C  8       2: extend(2,true);
641:C  8       3: emitindx(true);
642:C  8       4: immediate(fsize);
643:C  8       5..7: goto 2;
644:C  8        end; {case eareg}
645:C  7      end; {case eamode}
646:C  5   end; {emitea}
647:S
648:D  4   procedure opcode0;
649:D  5     { bit, MOVEP, immediate, MOVES }
650:S
651:D  5   type bitoptype = array[siz] of string[5];
652:D  5        immoptype = array[0..6] of string[4];
653:D  5   const bitop = bitoptype['btst ','bchg ','bclr ','bset '];
654:D  5         immop = immoptype['ori','andi','subi','addi','','eori','cmpi'];
655:S
656:C  5   begin { opcode0 }
657:C  5     with instr do
658:C  6       if bit8 then
659:C  7         if eamode = 1 then
660:C  8           begin
```



```
661:C  8             if odd(opmode) then instrbuf := 'movep.l '
662:C  9                            else instrbuf := 'movep.w ';
663:C  8             if opmode <= 5 then
664:C  9               begin emitardisp(eareg);
665:C  9                 comma; emitdir(D,reg1);
666:C  9               end
667:C  9             else begin
668:C  9                  emitdir(D,reg1);
669:C  9                  comma; emitardisp(eareg);
670:C  9                  end;
671:C  8           end
672:C  8         else begin {dynamic bit}
673:C  8                instrbuf := bitop[size];
674:C  8                emitdir(D,reg1);
675:C  8                comma; emitea(bytesiz);
676:C  8              end
677:C  8       else if reg1=4 then
678:C  8         begin instrbuf := bitop[size];
679:C  8           immediate(bytesiz); comma;
680:C  8           emitea(bytesiz {invalid});
681:C  8         end
682:C  8       else    { NOT bit8 }
683:C  8         begin instrbuf := immop[reg1];
684:C  8         if reg1=7 then
685:C  9         begin { moves }
686:C  9           instrbuf:='moves'; osize;
687:C  9           getinstrbytes(2);
688:C  9           if ext.long then
689:C 10             begin emitdir(ext.regclass,ext.reg); comma; emitea(size);
690:C 10             end
691:C 10           else
692:C 10             begin emitea(size); comma; emitdir(ext.regclass,ext.reg);
693:C 10             end;
694:C  9         end
695:C  9         else
696:C  9           if (eamode=7) and (eareg=4) then
697:C 10             begin
698:C 10             space; immediate(wordsiz); comma;
699:C 10             if size = bytesiz then sappend(instrbuf, 'ccr')
700:C 11                               else sappend(instrbuf, 'sr');
701:C 10             end
702:C 10           else begin
703:C 10             osize; immediate(size); comma; emitea(size);
704:C 10             end;
705:C  8         end;
706:C  5   end; {opcode0}
707:S
708:D  4   procedure move;
709:D  5   { opcodes 1..3: move byte,long,word }
710:D -2 5 var lsize: siz;
711:C  5   begin
712:C  5     with instr do
713:C  6       begin
714:C  6         case opcode of
715:C  7       1: lsize := bytesiz;
716:C  7       2: lsize := longsiz;
717:C  7       3: lsize := wordsiz;
718:C  7        end;
719:C  6        instrbuf := 'move';
720:C  6        if opmode=1 then sappend(instrbuf,'a');
```

```
721:C        6            sappend(instrbuf,opsize[lsize]);
722:C        6            emitea(lsize); comma;
723:C        6            {kluge to make emitea emit destination}
724:C        6            eamode := opmode; eareg := reg1;
725:C        6            if (eamode=7) and (eareg>1) then goto 2;
726:C        6            emitea(lsize  {invalid});
727:C        6          end;
728:C        5      end; {move}
729:S
730:D        4      procedure opcode4;
731:D        5      type miscoptype = array[0..7] of string[5];
732:D        5           unoptype = array[0..5] of string[4];
733:D        5      const predecr = 4; { eamode for predecrement }
734:D        5            miscop =
735:D        5             miscoptype['reset','nop','stop','rte','rtd','rts','trapv','rtr'];
736:D        5            unop = unoptype['negx', 'clr', 'neg', 'not', '', 'tst'];
737:D   -82  5      var    regstring: string80;
738:S
739:D        5        procedure emitreglist
740:D        6          (optype: opndtype; predecr: boolean; var regstring: string80);
741:D        6          { emit register list to 'regstring' according to mask }
742:D        6        type
743:D        6          regmasksymtype = array[0..15] of string[2];
744:D        6        const
745:D        6          regmasksym = regmasksymtype
746:D        6            ['d0','d1','d2','d3','d4','d5','d6','d7',
747:D        6             'a0','a1','a2','a3','a4','a5','a6','a7'];
748:D        6        var
749:D        6          state: (start,          {waiting for a '1'}
750:D        6                  open,           {have seen a lone '1'}
751:D    -2  6                  cont);          {at least two consecutive '1's}
752:D   -14  6          j,k,bitcount: integer;
753:S
754:D        6          procedure transition(b: boolean);
755:D    -2  7          var states: shortint;
756:C        7          begin
757:C        7            if b then
758:C        8              if optype = source then states := 6 else states := 5;
759:C        7            case state of
760:C        8          start: if b then
761:C        9                   begin state := open;
762:C        9                   sappend(regstring,regmasksym[bitcount]);
763:C        9                   end;
764:C        8          open : if b then
765:C        9                   begin state := cont;
766:C        9                   sappend(regstring,'-');
767:C        9                   end
768:C        9                 else begin state := start;
769:C        9                      sappend(regstring,'/');
770:C        9                      end;
771:C        8          cont : if not b then
772:C        9                   begin state := start;
773:C        9                   sappend(regstring,regmasksym[bitcount-1]);
774:C        9                   sappend(regstring,'/');
775:C        9                   end;
776:C        8            end; {case}
777:C        7          end; {transition}
778:S
779:C        6        begin {emitreglist}
780:C        6          getinstrbytes(2);
```



```
781:C        6          if ext.wordext = 0 then regstring := '(none) '
782:C        7          else
783:C        7            begin
784:C        7            state := start;
785:C        7            bitcount := 0; regstring := '';
786:C        7            if not predecr then
787:C        8              for j := 1 downto 0 do
788:C        9                begin
789:C        9                  for k := 7 downto 0 do
790:C       10                    begin transition(ext.mask[k+j*8]);
791:C       10                      bitcount := bitcount+1;
792:C       10                    end;
793:C        9                transition(false);
794:C        9                end
795:C        9            else
796:C        8              for j := 0 to 1 do
797:C        9                begin
798:C        9                  for k := 0 to 7 do
799:C       10                    begin transition(ext.mask[k+j*8]);
800:C       10                      bitcount := bitcount+1;
801:C       10                    end;
802:C        9                transition(false);
803:C        9                end;
804:C        7            end;
805:C        6          if optype = source then
806:C        7            regstring[strlen(regstring)] := ','
807:C        7          else setstrlen(regstring, strlen(regstring)-1);
808:C        6        end; {emitreglist}
809:S
810:D        5      procedure emitunop;
811:C        6      begin with instr do
812:C        7       begin
813:C        7       instrbuf := unop[reg1]; osize;
814:C        7       emitea(size {invalid});
815:C        7       end;
816:C        6      end;
817:S
818:D        5      procedure emitsreg;
819:C        6      begin
820:C        6        with ext do
821:C        7        begin
822:C        7          if dummy<>0 then goto 2;
823:C        7          if not long then
824:C        8          begin
825:C        8            if byteext=0 then sappend(instrbuf,'sfc')
826:C        9            else if byteext=1 then sappend(instrbuf,'dfc')
827:C       10                 else goto 2;
828:C        8          end
829:C        8          else
830:C        8          begin
831:C        8            if byteext=0 then sappend(instrbuf,'usp')
832:C        9            else if byteext=1 then sappend(instrbuf,'vbr')
833:C       10                 else goto 2;
834:C        8          end;
835:C        7        end;
836:C        6      end;
837:S
838:D        5      procedure jmpstates;
839:C        6      begin with instr do
840:C        7        case eamode of
```

```
841:C  8          2,5,6:;
842:C  8          7: if eareg>3 then goto 2;
843:C  8          otherwise goto 2;
844:C  8          end;
845:C  6      end;
846:S
847:C  5      begin {opcode4}
848:C  5        with instr do
849:C  6          if bit8 then
850:C  7            if ord(size) = 2 then
851:C  8              begin instrbuf := 'chk.';
852:C  8                emitea(wordsiz); comma;
853:C  8                emitdir(D,reg1);
854:C  8              end
855:C  8            else begin instrbuf := 'lea ';
856:C  8                   emitea(invalid); comma;
857:C  8                   emitdir(A,reg1);
858:C  8                 end
859:C  8          else { NOT bit8 }
860:C  7            case reg1 of
861:C  8              0: if size=invalid then
862:C  9                   begin instrbuf := 'move sr,'; emitea(wordsiz);
863:C  9                   end
864:C  9                 else emitunop;
865:C  8              1: if size=invalid then
866:C  9                   begin instrbuf := 'move ccr,'; emitea(wordsiz);
867:C  9                   end
868:C  9                 else emitunop;
869:C  8                 2: if size = invalid then
870:C  9                   begin instrbuf := 'move ';
871:C  9                   emitea(wordsiz);
872:C  9                   sappend(instrbuf, ',ccr');
873:C  9                   end
874:C  9                 else emitunop;
875:C  8                 3: if size = invalid then
876:C  9                   begin instrbuf := 'move ';
877:C  9                   emitea(wordsiz);
878:C  9                   sappend(instrbuf, ',sr');
879:C  9                   end
880:C  9                 else emitunop;
881:C  8                 4: case ord(size) of
882:C  9                      0: begin instrbuf := 'nbcd ';
883:C  9                           emitea(bytesiz (irvalid));
884:C  9                         end;
885:C  9                      1: if eamode = 0 then
886:C 10                           begin instrbuf := 'swap '; emitdir(D,eareg);
887:C 10                           end
888:C 10                         else
889:C 10                           begin instrbuf := 'pea '; emitea(invalid);
890:C 10                           end;
891:C  9                   2,3: if eamode = 0 then
892:C 10                           begin instrbuf := 'ext';
893:C 10                           sappend(instrbuf,opsize[pred(size)]);
894:C 10                           emitdir(D,eareg);
895:C 10                           end
896:C 10                         else
897:C 10                           begin
898:C 10                           instrbuf := 'movem';
899:C 10                           sappend(instrbuf,opsize[pred(size)]);
900:C 10                           emitreglist(source,eamode=predecr,regstring);
```



```
901:C 10                           sappend(instrbuf,regstring);
902:C 10                           emitea(invalid);
903:C 10                           end;
904:C  9                      end; {case size}
905:C  8                 5: if size = invalid then
906:C  9                      begin instrbuf := 'tas ';
907:C  9                      emitea(bytesiz (invalid));
908:C  9                      end
909:C  9                    else emitunop;
910:C  8                 6: if size<longsiz then goto 2
911:C  9                    else
912:C  9                      begin instrbuf := 'movem';
913:C  9                      sappend(instrbuf,opsize[pred(size)]);
914:C  9                      emitreglist(dest,false,regstring);
915:C  9                      emitea(invalid); comma;
916:C  9                      sappend(instrbuf,regstring);
917:C  9                      end;
918:C  8                 7: if ord(size) = 2 then
919:C  9                      begin instrbuf := 'jsr ';
920:C  9                      jmpstates;
921:C  9                      emitea(invalid);
922:C  9                      end
923:C  9                    else if ord(size) = 3 then
924:C 10                      begin instrbuf := 'jmp ';
925:C 10                      jmpstates;
926:C 10                      emitea(invalid);
927:C 10                      end
928:C 10                    else
929:C 10                      case eamode of
930:C 11                        0,1:
931:C 11                           begin instrbuf := 'trap ';
932:C 11                           emitimm(eareg+8*eamode);
933:C 11                           if eareg + 8*eamode = 9 then
934:C 12                             begin
935:C 12                             comma; immediate(wordsiz);
936:C 12                             comma; extend(4,true);
937:C 12                             end
938:C 12                           else
939:C 12                           if eareg + 8*eamode = 1 then
940:C 13                             begin comma; immediate(wordsiz);
941:C 13                             end
942:C 13                           else
943:C 13                           if eareg + 8*eamode = 0 then
944:C 14                             begin
945:C 14                             comma; getinstrbytes(2);
946:C 14                             lastline := ext.uwordext;
947:C 14                             emitimm(lastline);
948:C 14                             while strlen(instrbuf) < 20 do space;
949:C 14                             sappend(instrbuf, 'COMPILED LINE NUMBER ');
950:C 14                             emitint(lastline);
951:C 14                             end;
952:C 11                           end;
953:C 11                        2: begin instrbuf := 'link ';
954:C 11                           emitdir(A,eareg); comma;
955:C 11                           immediate(wordsiz);
956:C 11                           end;
957:C 11                        3: begin instrbuf := 'unlk '; emitdir(A,eareg);
958:C 11                           end;
959:C 11                        4,5:
960:C 11                           begin instrbuf := 'move ';
```

```
   961:C       11                        if eamode = 5 then sappend(instrbuf,'usp,');
   962:C       11                        emitdir(A,eareg);
   963:C       11                        if eamode = 4 then sappend(instrbuf,',usp');
   964:C       11                      end;
   965:C       11                    6: begin
   966:C       11                       instrbuf := miscop[eareg];
   967:C       11                       if (eareg=2) or (eareg=4) then {stop}{rtd}
   968:C       12                          begin space; immediate(wordsiz);
   969:C       12                          end;
   970:C       11                      end;
   971:C       11                    7: begin     { movec }
   972:C       11                       if ord(size)<>1 then goto 2;
   973:C       11                       instrbuf := 'movec ';
   974:C       11                       getinstrbytes(2);
   975:C       11                       if eareg=2 then
   976:C       12                       begin
   977:C       12                       emitsreg; comma; emitdir(ext.regclass,ext.reg);
   978:C       12                       end
   979:C       12                       else
   980:C       12                         if eareg=3 then
   981:C       13                         begin
   982:C       13                         emitdir(ext.regclass,ext.reg); comma; emitsreg;
   983:C       13                         end
   984:C       13                         else goto 2;
   985:C       11                      end;
   986:C       11                  end; {case eamode}
   987:C        8              end; {case reg1}
   988:C        5     end; {opcode4}
   989:S
   990:D        4     procedure quick;
   991:C        5     begin
   992:C        5     with instr do if reg1 = 0 then emitimm(8)
   993:C        7                               else emitimm(reg1);
   994:C        5     comma;
   995:C        5     end;
   996:S
   997:D        4     procedure shift;
   998:D        5     type
   999:D        5       shiftoptype = array[0..7] of string[4];
  1000:D        5     const
  1001:D        5       shiftop =
  1002:D        5         shiftoptype['asr','lsr','roxr','ror','asl','lsl','roxl','rol'];
  1003:C        5     begin
  1004:C        5       with instr do
  1005:C        6         if size = invalid then
  1006:C        7           begin instrbuf := shiftop[4*ord(bit8)+reg1];
  1007:C        7           space; emitea(bytesiz {invalid});
  1008:C        7           end
  1009:C        7         else
  1010:C        7           begin
  1011:C        7           instrbuf := shiftop[4*ord(bit8)+eamode mod 4];
  1012:C        7           osize;
  1013:C        7           if eamode div 4 = 1 then
  1014:C        8             begin
  1015:C        8             emitdir(D,reg1);
  1016:C        8             comma;
  1017:C        8             end
  1018:C        8           else quick;
  1019:C        7           emitdir(D,eareg);
  1020:C        7           end;
```



```
  1021:C        5     end; {shift}
  1022:S
  1023:C        4   begin {decode}
  1024:C        4     with instr do
  1025:C        5       case opcode of
  1026:C        6         0: opcode0;
  1027:C        6         1,2,3: move;
  1028:C        6         4: opcode4;
  1029:C        6         5: if size = invalid then
  1030:C        7              begin
  1031:C        7              if eamode = 1 then
  1032:C        8                begin
  1033:C        8                instrbuf := 'db';
  1034:C        8                if cond = 1 then sappend(instrbuf,'ra')
  1035:C        9                else sappend(instrbuf,condcode[cond]);
  1036:C        8                space; emitdir(D, eareg);
  1037:C        8                comma; extend(2,true);
  1038:C        8                end
  1039:C        8              else
  1040:C        8                begin
  1041:C        8                instrbuf := 's';
  1042:C        8                sappend(instrbuf,condcode[cond]);
  1043:C        8                space; emitea(bytesiz {invalid});
  1044:C        8                end;
  1045:C        7              end
  1046:C        7            else
  1047:C        7              begin
  1048:C        7              if bit8 then instrbuf := 'subq'
  1049:C        8              else instrbuf := 'addq';
  1050:C        7              osize; quick;
  1051:C        7              emitea(size {invalid});
  1052:C        7              end;
  1053:C        6         6: begin instrbuf := 'b';
  1054:C        6              if cond = 0 then sappend(instrbuf,'ra')
  1055:C        7              else if cond = 1 then sappend(instrbuf,'sr')
  1056:C        8                                 else sappend(instrbuf,condcode[cond]);
  1057:S
  1058:C        6              if displ = 0 then
  1059:C        7                begin space; extend(2,true); end
  1060:C        7              else
  1061:C        7                begin sappend(instrbuf,'.s ');
  1062:C        7                ext.longext := pc + 2 + displ;
  1063:C        7                gvrstring(nilgvr,ext.longext,true,false);
  1064:C        7                sappend(instrbuf,gvaluestring);
  1065:C        7                end;
  1066:C        6            end;
  1067:C        6         7: begin instrbuf := 'moveq ';
  1068:C        6              emitimm(displ); comma;
  1069:C        6              emitdir(D,reg1);
  1070:C        6            end;
  1071:S
  1072:S
  1073:S
  1074:C        6 8,9,11,12,13: begin
  1075:C        6              instrbuf := arithop[opcode];
  1076:C        6              if size=invalid then
  1077:C        7               begin
  1078:C        7               if odd(opcode) then
  1079:C        8                begin
  1080:C        8                sappend(instrbuf,'a');
```

```
1081:C        8              if bit8 then
1082:C        9                   begin
1083:C        9                   sappend(instrbuf, opsize[longsiz]);
1084:C        9                   emitea(longsiz);
1085:C        9                   end
1086:C        9              else begin
1087:C        9                   sappend(instrbuf,opsize[wordsiz]);
1088:C        9                   emitea(wordsiz);
1089:C        9                   end;
1090:C        8              comma; emitdir(A,reg1);
1091:C        8              end
1092:C        8             else
1093:C        8              begin
1094:C        8              if opcode = 8 then instrbuf := 'div'
1095:C        9                            else instrbuf := 'mul';
1096:C        8              if bit8 then sappend(instrbuf,'s ')
1097:C        9                      else sappend(instrbuf,'u ');
1098:C        8              emitea(wordsiz); comma; emitdir(D,reg1);
1099:C        8              end
1100:C        8             end
1101:C        7            else if (not bit8) or (eamode > 1) or (opcode = 11) then
1102:C        8              begin
1103:C        8              if opcode = 11 then
1104:C        9               if bit8 then
1105:C       10                if eamode = 1 then
1106:C       11                  begin
1107:C       11                  sappend(instrbuf,'m'); osize;
1108:C       11                  emitpostincr(eareg); comma; emitpostincr(reg1);
1109:C       11                  goto 1;
1110:C       11                  end
1111:C       11                else instrbuf := 'eor';
1112:C        8              osize;
1113:C        8              if bit8 then begin emitdir(D,reg1); comma; emitea(size);
1114:C        9                                 end
1115:C        9                      else begin emitea(size); comma; emitdir(D,reg1);
1116:C        9                                 end;
1117:C        8              end
1118:C        8            else
1119:C        8              begin
1120:C        8              if odd(opcode) then begin sappend(instrbuf,'x'); osize; end
1121:C        9              else if opcode = 8 then instrbuf := 'sbcd '
1122:C       10              else if size = bytesiz then instrbuf := 'abcd '
1123:C       11              else begin
1124:C       11                   instrbuf := 'exg ';
1125:C       11                   if eamode = 0 then
1126:C       12                     begin emitdir(D,reg1); comma; emitdir(D,eareg) end
1127:C       12                   else if opmode = 5 then
1128:C       13                     begin emitdir(A,reg1); comma; emitdir(A,eareg) end
1129:C       13                   else
1130:C       13                     begin emitdir(D,reg1); comma; emitdir(A,eareg) end;
1131:C       11                   goto 1;
1132:C       11                   end;
1133:C        8              if eamode = 0 then
1134:C        9                   begin emitdir(D,eareg); comma; emitdir(D,reg1);
1135:C        9                   end
1136:C        9              else begin emitpredecr(eareg); comma; emitpredecr(reg1);
1137:C        9                   end;
1138:C        8              end;
1139:C        6             end;
1140:C        6         14:shift;
```



```
1141:C        6          otherwise goto 2;
1142:C        6         end; {case}
1143:C        4        goto 1;
1144:C        4       2: begin defineword;
1145:C        4            if decodestate <> abscode then decodestate := consts;
1146:C        4          end;
1147:C        4     1: end; {decode}
1148:S
1149:D        3   procedure definecaseword;
1150:D    -4  4   var savepc: integer;
1151:C        4   begin
1152:C        4   instrsize := 0;
1153:C        4   instrbuf := 'case jump   ';
1154:C        4   savepc := pc; pc := tablepc;
1155:C        4   extend(2, true);
1156:C        4   pc := savepc;
1157:C        4   end;
1158:S
1159:D        3   procedure decodestuff;
1160:D        4   label 1;
1161:D    -4  4   var temp: integer;
1162:S
1163:D        4     procedure printprocboundary;
1164:D        5     label 1;
1165:D   -16  5     var  defaddr,deflimit,len,gvrbase: integer;
1166:D   -20  5          veloc:  addrec;
1167:C        5     begin
1168:C        5       defaddr:=newmods^.defaddr.a;
1169:C        5       deflimit:=defaddr+newmods^.defsize;
1170:C        5       while defaddr < deflimit do
1171:C        6         begin
1172:C        6           len:=strlen(symbolptr(defaddr)^);
1173:C        6           len:=len+2-ord(odd(len));
1174:C        6           gvrbase:=defaddr+len;
1175:C        6           with gvrptr(gvrbase)^ do
1176:C        7             if primarytype = loadgvr^.primarytype then
1177:C        8               begin
1178:C        8               veloc.a:=gvrbase+sizeof(generalvalue,false);
1179:C        8               if veloc.vep^.value = PC then goto 1;
1180:C        8               end;
1181:C        6           defaddr:=defaddr+len+ord(symtableptr(defaddr)^[len+1]);
1182:C        6         end;
1183:C        5         listln;
1184:C        5 1:    list; write(listing,
1185:C        5 '- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ',
1186:C        5 '- - - - - ');
1187:C        5       if defaddr < deflimit then
1188:C        6         write(listing,symbolptr(defaddr)^);
1189:C        5       writeln(listing);
1190:C        5     end; {printprocboundary}
1191:S
1192:C        4   begin {decodestuff}
1193:C        4  1: case decodestate of
1194:C        5       consts:
1195:C        5           begin getinstruction;
1196:C        5             if (instr.w = 20054) {LINK A6}
1197:C        6              or (instr.w = 20033) {TRAP #1} then
1198:C        6               begin decodestate := code; decode;
1199:C        6                 if (rangetype = norange) or
1200:C        7                    ((rangetype = pcrange) and (PC >= lowrange) and (PC < highrange)) or
```

```
1201:C        7                   ((rangetype = linerange) and (lastline >= lowrange) and (lastline < highrange))
1202:C        7                   then printprocboundary;
1203:C        6                 end
1204:C        6             else if (instr.w = 20217) (JMP long abs)
1205:C        7              or (instr.w = 24576) (BRA 16 bit) then
1206:C        7                decode
1207:C        7             else defineword;
1208:C        5           end;
1209:C        5       abscode,
1210:C        5       code:     begin
1211:C        5                 getinstruction;
1212:C        5                 decode;
1213:C        5                 if decodestate <> abscode then
1214:C        6                   if instr.w = 20062 (UNLK A6) then decodestate := endofproc
1215:C        7                   else if instr.w = 20219 (JMP pc indexed) then
1216:C        8                     begin oldstate := code; decodestate := startcase end;
1217:C        5                 end;
1218:C        5       startcase:
1219:C        5         begin
1220:C        5           tablePC := PC;
1221:C        5           definecaseword;
1222:C        5           casecodestart:=ext.wordext+PC;
1223:C        5           decodestate := casetable;
1224:C        5         end;
1225:C        5       casetable:
1226:C        5         begin
1227:C        5           if PC = casecodestart
1228:C        6           then begin decodestate := oldstate; goto 1 end
1229:C        6           else
1230:C        6             begin definecaseword;
1231:C        6               if not fortranflag then
1232:C        7                 begin
1233:C        7                   temp:=ext.wordext+tablePC;
1234:C        7                   if temp<casecodestart then casecodestart := temp;
1235:C        7                 end;
1236:C        6             end;
1237:C        5         end;
1238:C        5       endofproc:
1239:C        5         begin getinstruction; decode;
1240:C        5           if (instr.w = 20085 (RTS) )
1241:C        6              or (instr.w div 8 = 2522 (JMP (An)) ) then
1242:C        6             decodestate := consts;
1243:C        5         end;
1244:C        5       end; (case)
1245:C        4   end; (decodestuff)
1246:S
1247:C        3 begin (listinstruction)
1248:C        3   decodestuff;
1249:C        3   bytesleft := instrsize; firstline := true;
1250:C        3   if (rangetype = norange) or
1251:C        4      ((rangetype = pcrange) and (PC >= lowrange) and (PC < highrange)) or
1252:C        4      ((rangetype = linerange) and (lastline >= lowrange) and (lastline < highrange)) then
1253:C        4    repeat
1254:C        5     list;
1255:C        5     if firstline then write(listing,PC:8,' ')
1256:S
1257:C        6     else write(listing,'':9  {17}  );
1258:C        5     printinstrword;
1259:C        5     if bytesleft>0 then printinstrword
1260:C        6     else
```



```
1261:C        6       if firstline then write(listing,'':5);
1262:C        5     if firstline then
1263:C        6       begin writeln(listing,'':9,instrbuf); firstline := false end
1264:C        6     else writeln(listing);
1265:C        5    until bytesleft = 0;
1266:C        3   PC:=PC + instrsize;
1267:C        3 end; (listinstruction)
1268:S
1269:D        2 procedure getcodeblocks;
1270:D    -4  3 var junkint: integer;
1271:D    -8  3     pclimit: integer;
1272:D   -12  3     textstep: addrec;
1273:D   -14  3     textrecctr: shortint;
1274:C        3 begin
1275:C        3  with newmods^,directory.drp^ do
1276:C        4    begin
1277:C        4    textstep.a:=directory.a+sizeof(moduledirectory);
1278:C        4    junkint:=strlen(textstep.syp^);
1279:C        4    textstep.a := textstep.a+junkint+2-ord(odd(junkint));
1280:C        4    if executable then textstep.a := textstep.a + textstep.gvp^.short;
1281:C        4    textrecctr:=textrecords;
1282:C        4    while textrecctr > 0 do with textstep.tdp^ do
1283:C        6      begin
1284:C        6      textrecctr:=textrecctr-1;
1285:C        6      list; writeln(listing,'TEXT RECORD #',
1286:C        6      textrecords-textrecctr, ' of ''', fdirectory^[vmodnum].dtid, ''':');
1287:C        6      list; writeln(listing,'  TEXT start block ',textstart:4,
1288:C        6                            Size ',textsize,' bytes');
1289:C        6      list; writeln(listing,'  REF  start block ',refstart:4,
1290:C        6                            Size ',refsize,' bytes');
1291:C        6      textstep.a :=textstep.a+sizeof(textdescriptor);
1292:C        6      PC := 0;
1293:C        6      loadgvr := textstep.gvp;
1294:C        6      gvrstring(textstep.gvp,PC,false,false);
1295:C        6      gbytes(inbuf.a,textsize);
1296:C        6      readblocks(filefib.fbp^, inbuf.p^,textsize,fileblock+textstart);
1297:C        6      codeindex:=0;  codecount:=textsize;
1298:C        6      gbytes(refptr.a,refsize);
1299:C        6      readblocks(filefib.fbp^,refptr.p^,refsize,fileblock+refstart);
1300:S
1301:C        6      refgvr:=refptr;
1302:C        6      reflim:=refptr.a+refsize;
1303:C        6      refloc:=PC;       nextref;
1304:C        6      pclimit := PC + textsize;
1305:S
1306:C        6      list; writeln(listing,'  LOAD address     ',gvaluestring);
1307:C        6      listln;
1308:C        6      while PC < pclimit do listinstruction;
1309:C        6      listln; listln;
1310:C        6      lowheap := inbuf;
1311:C        6      end;
1312:C        4    end; (with newmods^,directory^)
1313:C        3 end; (getcodeblocks)
1314:S
1315:D        2 procedure listdefs;
1316:D        3 var
1317:D    -8  3   len,val:      integer;
1318:D   -16  3   lim,p1:       addrec;
1319:C        3 begin
1320:C        3 prepunassem;
```

```
1321:C        3 with newmods^ do
1322:C        4   begin
1323:C        4   list; writeln(listing,'  DEF table of ''',
1324:C        4         fdirectory^[vmodnum].dtid, ''':');
1325:C        4   listln;
1326:C        4   p1 := defaddr;
1327:C        4   lim.a := p1.a + defsize;
1328:C        4   while p1.a < lim.a do
1329:C        5     begin
1330:C        5     len:=strlen(p1.syp^);
1331:C        5     list; write(listing,'':5,p1.syp^,'':(30-len));
1332:C        5     p1.a := p1.a + len+2-ord(odd(len));
1333:C        5     val := 0;
1334:C        5     gvrstring(p1.gvp,val,false,false);
1335:C        5     writeln(listing,gvaluestring);
1336:C        5     end;
1337:C        4   end;
1338:C        3 listln;
1339:C        3 newmods := modsave;
1340:C        3 lowheap := infost;
1341:C        3 end;
1342:S
1343:D        2 procedure listexts;
1344:D        3 var
1345:D    -4  3   i:            integer;
1346:D    -8  3   p1:           addrec;
1347:C        3 begin
1348:C        3 prepunassem;
1349:C        3 with newmods^ do
1350:C        4   begin
1351:C        4   list; writeln(listing,'  EXT table of ''',
1352:C        4         fdirectory^[vmodnum].dtid, ''':');
1353:C        4   listln;
1354:C        4   for i:=2 to listsize-1 do if listaddr^[i] <> 0 then
1355:C        6     begin
1356:C        6       p1.a := extaddr.a + listaddr^[i];
1357:C        6       list; writeln(listing,'':5,p1.syp^);
1358:C        6     end;
1359:C        4   end;
1360:C        3 listln;
1361:C        3 newmods := modsave;
1362:C        3 lowheap := infost;
1363:C        3 end;
1364:S
1365:D        2 procedure listtext;
1366:D        3 const pagesize = pageblocks * blocksize;
1367:D        3 var
1368:D    -8  3   textbuf,ptr:  addrec;
1369:D   -20  3   i,j,pages:    integer;
1370:D   -24  3   readsize:     integer;
1371:D   -25  3   linestart:    boolean;
1372:S
1373:D        3   procedure dochar(c: char);
1374:C        4   begin
1375:C        4   if linestart then list;
1376:C        4   linestart := (c = eol);
1377:C        4   if linestart then writeln(listing) else write(listing, c);
1378:C        4 end;
1379:S
1380:C        3 begin
```



```
1381:C        3 prepunassem;
1382:C        3 gbytes(textbuf.a, pagesize);
1383:C        3 with newmods^, directory.drp^ do
1384:C        4   begin
1385:C        4   list; writeln(listing,'  DEFINE SOURCE of ''',
1386:C        4         fdirectory^[vmodnum].dtid, ''':');
1387:C        4   listln;
1388:C        4   pages := (sourcesize + (pagesize-1)) div pagesize;
1389:C        4   for i := 0 to pages-1 do
1390:C        5    begin
1391:C        5    readsize := sourcesize - i*pagesize;                    { scs 1/17/83 }
1392:C        5    if readsize > pagesize then readsize := pagesize;       { scs 1/17/83 }
1393:C        5    readblocks(filefib.fbp^,textbuf.p^,readsize,            { scs 1/17/83 }
1394:C        5                                  fileblock+sourceblock+i*pageblocks);
1395:C        5    ptr := textbuf;      linestart := true;
1396:C        5    repeat
1397:C        6      case ptr.cp^ of
1398:C        7        chr(etx),
1399:C        7        nullchar:  ptr.a := textbuf.a + pagesize;
1400:C        7       otherwise dochar(ptr.cp^);
1401:C        7       end;
1402:C        6      ptr.a := ptr.a + 1;
1403:C        6    until ptr.a >= textbuf.a + pagesize;
1404:C        5    if not linestart then dochar(eol);
1405:C        5    end;
1406:C        4   end;
1407:C        3 listln;
1408:C        3 newmods := modsave;
1409:C        3 lowheap := infost;
1410:C        3 end;
1411:S
1412:D        2 procedure disassemble;
1413:C        3 begin
1414:C        3 prepunassem;
1415:C        3 nilgvr := NIL;
1416:C        3 getcodeblocks;
1417:C        3 newmods := modsave;
1418:C        3 lowheap := infost;
1419:C        3 end;
1420:S
1421:D        2 procedure getbounds;
1422:C        3 begin
1423:C        3   lastline := -1;
1424:C        3   fgotoxy(output, 0,13);
1425:C        3   write('lower bound? ');
1426:C        3   if readint(lowrange) then
1427:C        4        begin
1428:C        4        write('upper bound? ');
1429:C        4        if not readint(highrange) then
1430:C        5           highrange := maxint;
1431:C        4        end
1432:C        4   else begin
1433:C        4        lowrange := minint;
1434:C        4        highrange := maxint;
1435:C        4        end;
1436:C        3 end;
1437:S
1438:C        2 begin {unassemble}
1439:C        2   fortranflag := false;
1440:C        2   decodestate := notype;        dumped := false;
```

```
1441:C        2   repeat
1442:C        3     fgotoxy(output, 0,2);
1443:C        3     writeln('Q  Quit',cteos);
1444:C        3     writeln('S  Stop unassembling');
1445:C        3     writeln('T  print import Text');
1446:C        3     writeln('E  print Ext table');
1447:C        3     writeln('D  print Def table');
1448:C        3     writeln('A  unassemble all (Assembler conventions)');
1449:C        3     writeln('C  unassemble all (Compiler  conventions)');
1450:C        3     writeln('P  PC   range     (Assembler conventions)');
1451:C        3     writeln('L  Line range     (Compiler  conventions)', cteos);
1452:C        3     getcommandchar('unassemble option?',commandchar);
1453:C        3     if commandchar <> ' ' then
1454:C        4      case commandchar of
1455:C        5       'S':  decodestate := quittype;
1456:C        5       'Q':  begin
1457:C        5             decodestate := quittype;
1458:C        5             quit;
1459:C        5             end;
1460:C        5       'A':  begin rangetype := norange; decodestate := abscode; disassemble; end;
1461:C        5       'C':  begin rangetype := norange; decodestate := consts; disassemble; end;
1462:C        5       'P':  begin rangetype := pcrange; decodestate := abscode; getbounds; disassemble; end;
1463:C        5       'L':  begin rangetype := linerange; decodestate := consts; getbounds; disassemble; end;
1464:C        5       'T':  listtext;
1465:C        5       'D':  listdefs;
1466:C        5       'E':  listexts;
1467:C        5       otherwise dobeep;
1468:C        5       end;
1469:C        3   until decodestate = quittype;
1470:C        2 end; {unassemble}
1471:S
1472:S
1473:D        1 procedure makenewgvr(var oldptr: addrec;
1474:D        2                      modptr: moddescptr);
1475:S
1476:D    -2  2 var  refsize:   shortint;
1477:D    -2  2      lastptr,
1478:D    -2  2      firstptr,
1479:D    -2  2      vptr,
1480:D   -18  2      gptr:      addrec;
1481:S
1482:D        2   procedure runlist(var oldptr: addrec; modptr: moddescptr; sub: boolean);
1483:D    -1  3   var done:     boolean;
1484:D    -6  3       defptr:   addrec;
1485:S
1486:D        3     procedure addref(add: shortint; sub: boolean);
1487:D   -12  4     var iptr,jptr,tp:   addrec;
1488:D   -12  4         notdone,
1489:D   -14  4         notcancels:     boolean;
1490:C        4     begin
1491:C        4     if add = 0 then
1492:C        5       if sub then vptr.vep^.value  := vptr.vep^.value - modptr^.relocdelta
1493:C        6              else vptr.vep^.value  := vptr.vep^.value + modptr^.relocdelta
1494:C        6     else if add = 1 then
1495:C        6       if sub then vptr.vep^.value  := vptr.vep^.value - modptr^.globaldelta
1496:C        7              else vptr.vep^.value  := vptr.vep^.value + modptr^.globaldelta;
1497:S
1498:C        4     iptr := lastptr; notdone := true; notcancels := true;
1499:C        4     while (iptr.a > firstptr.a) and notdone do
1500:C        5       begin
```



```
1501:C        5       iptr.a := iptr.a - sizeof(referenceptr);
1502:C        5       with iptr.rpp^ do
1503:C        6         if adr <= add then
1504:C        7           begin
1505:C        7           if adr = add then notcancels := (op = subit) = sub;
1506:C        7           iptr.a := iptr.a + sizeof(referenceptr);
1507:C        7           notdone := false;
1508:C        7           end;
1509:C        5       end;
1510:S
1511:C        4     if notcancels then
1512:C        5       begin
1513:C        5       gbytes(jptr.a, sizeof(referenceptr));
1514:C        5       lastptr := lowheap;
1515:C        5       while jptr.a > iptr.a do
1516:C        6         begin
1517:C        6         tp.a := jptr.a - sizeof(referenceptr);
1518:C        6         jptr.rpp^ := tp.rpp^;
1519:C        6         jptr := tp;
1520:C        6         end;
1521:C        5       with iptr.rpp^ do
1522:C        6         begin adr := add; last := false;
1523:C        6         if sub then op := subit else op := addit;
1524:C        6         end;
1525:C        5       end
1526:C        5     else
1527:C        5       begin
1528:C        5       tp.a := iptr.a - sizeof(referenceptr);
1529:C        5       while iptr.a < lastptr.a do
1530:C        6         begin
1531:C        6         tp.rpp^ := iptr.rpp^;
1532:C        6         tp := iptr;
1533:C        6         iptr.a := iptr.a + sizeof(referenceptr);
1534:C        6         end;
1535:C        5       lastptr.a := lastptr.a - sizeof(referenceptr);
1536:C        5       lowheap := lastptr;
1537:C        5       end;
1538:C        4     end;
1539:S
1540:C        3   begin {runlist}
1541:C        3   with oldptr.gvp^ do
1542:C        4     begin
1543:C        4     if longoffset then oldptr.a := oldptr.a+sizeof(generalvalue, true)
1544:C        5                   else oldptr.a := oldptr.a+sizeof(generalvalue, false);
1545:C        4     if valueextend then
1546:C        5       begin
1547:C        5       if sub then vptr.vep^.value := vptr.vep^.value - oldptr.vep^.value
1548:C        6              else vptr.vep^.value := vptr.vep^.value + oldptr.vep^.value;
1549:C        5       oldptr.a := oldptr.a + sizeof(valueextension, sint);
1550:C        5       end;
1551:C        4     if primarytype <> absolute then
1552:C        5       begin
1553:C        5       if modptr = NIL then
1554:C        6         begin
1555:C        6         modptr := newmods;      done := false;
1556:C        6         repeat
1557:C        7         with modptr^ do
1558:C        8           if patchmod then modptr := link
1559:C        9           else if oldptr.a < defaddr.a then modptr := link
1560:C       10           else if oldptr.a > defaddr.a + defsize then modptr := link
```

```
1561:C       11          else done := true;
1562:C        7        until done;
1563:C        6        end;
1564:C        5      case primarytype of
1565:C        6       relocatable:      addref(0, sub);
1566:C        6       global:           addref(1, sub);
1567:C        6       general:
1568:C        6        begin
1569:C        6        done := false;
1570:C        6        repeat with oldptr.rpp^ do
1571:C        8          begin
1572:C        8          defptr := modptr^.extaddr.ptp^[adr];
1573:C        8          if modptr^.unresbits.bmp^[adr] then
1574:C        9            addref(defptr.rp.adr, sub <> (op = subit))
1575:C        9          else
1576:C        9            begin
1577:C        9            defptr.a := defptr.a + strlen(defptr.syp^) + 2
1578:C        9                      - ord(odd(strlen(defptr.syp^)));
1579:C        9            runlist(defptr, NIL, sub <> (op = subit));
1580:C        9            end;
1581:C        8          oldptr.a := oldptr.a + sizeof(referenceptr);
1582:C        8          done := last;
1583:C        8          end;
1584:C        7        until done;
1585:C        6        end; {general}
1586:C        6      end; {case}
1587:C        5    end; {primarytype <> absolute}
1588:C        4  end; {with}
1589:C        3  end; {runlist}
1590:S
1591:C        2 begin {makenewgvr}
1592:C        2   gbytes(gptr.a, sizeof(generalvalue));
1593:C        2   gptr.gvp^ := oldptr.gvp^;
1594:C        2   with gptr.gvp^ do
1595:C        3     begin
1596:C        3     if not longoffset then
1597:C        4       lowheap.a := lowheap.a -
1598:C        4       (sizeof(generalvalue) - sizeof(generalvalue, false));
1599:C        3     gbytes(vptr.a, sizeof(valueextension, sint));
1600:C        3     vptr.vep^.value := 0;
1601:C        3     valueextend := true;
1602:C        3     end;
1603:S
1604:C        2   firstptr := lowheap;  lastptr := firstptr;
1605:S
1606:C        2   runlist(oldptr, modptr, false);
1607:C        2   with gptr.gvp^ do
1608:C        3     begin
1609:C        3     refsize := lastptr.a - firstptr.a;
1610:C        3     if refsize = 0 then primarytype := absolute
1611:C        4     else
1612:C        4       begin
1613:C        4       if refsize = sizeof(referenceptr) then with firstptr.rpp^ do
1614:C        6         if adr <= 1 then if op = addit then
1615:C        8           begin
1616:C        8           if adr = 0 then primarytype := relocatable
1617:C        9                      else primarytype := global;
1618:C        8           lastptr := firstptr;
1619:C        8           lowheap := lastptr;
1620:C        8           refsize := 0;
```



```
1621:C        8           end;
1622:C        4       if refsize > 0 then
1623:C        5         begin
1624:C        5         firstptr.a := lastptr.a - sizeof(referenceptr);
1625:C        5         firstptr.rpp^.last := true;
1626:C        5         end;
1627:C        4       end;
1628:C        3     short := lastptr.a - gptr.a;        {even if it is long variety}
1629:C        3     end;
1630:C        2 end;
1631:S
1632:S
1633:D        1 procedure compressgvr(gvptr: addrec);
1634:D    -4  2 var vptr:   addrec;
1635:C        2 begin
1636:C        2 with gvptr.gvp^ do if valueextend then
1637:C        4   begin
1638:C        4   if longoffset then vptr.a := gvptr.a + sizeof(generalvalue, true)
1639:C        5                     else vptr.a := gvptr.a + sizeof(generalvalue, false);
1640:C        4   with vptr.vep^ do
1641:C        5    if value = 0 then
1642:C        6     begin
1643:C        6     lowheap.a := lowheap.a - sizeof(valueextension, sint);
1644:C        6     fastmove(point(vptr.a + sizeof(valueextension, sint)), vptr.p,
1645:C        6              lowheap.a - vptr.a);
1646:C        6     valueextend := false; short := short - sizeof(valueextension, sint);
1647:C        6     end;
1648:C        4   end;
1649:C        2 end;
1650:S
1651:D        1 procedure rsolve;
1652:D   -12  2 var modptr, lastptr, nextptr: moddescptr;
1653:D   -20  2     mrbase,mgbase:    integer;
1654:D   -24  2     sp:               addrec;
1655:D   -26  2     len:              shortint;
1656:D   -28  2     i:                shortint;
1657:S
1658:C        2 begin
1659:C        2   modptr := newmods; lastptr := NIL; {reverse the pointers}
1660:C        2   while modptr <> NIL do with modptr^ do
1661:C        4     begin nextptr := link;      link := lastptr;
1662:C        4           lastptr := modptr;    modptr := nextptr;
1663:C        4     end;
1664:C        2   newmods := lastptr;
1665:S
1666:C        2   startgvr.p := NIL;    startgvrmod := NIL;
1667:C        2   modptr := newmods;    totalpatchspace := 0;
1668:C        2   forwardpatches:=NIL;  backwardpatches:=NIL;
1669:C        2   mrbase := startreloc; mgbase := startglobal;
1670:C        2   while modptr <> NIL do with modptr^ do
1671:C        4     begin
1672:C        4     if patchmod then
1673:C        5       begin
1674:C        5       patchbase := mrbase;
1675:C        5       mrbase := mrbase + patchsize;
1676:C        5       totalpatchspace := totalpatchspace + patchsize;
1677:C        5       if forwardpatches = NIL then forwardpatches := modptr
1678:C        6       else lastptr^.patchlink := modptr;
1679:C        5       lastptr := modptr;
1680:C        5       end
```

```
1681:C        5     else with directory.drp^ do
1682:C        6       begin
1683:C        6       relocbase := mrbase;    relocdelta :=  mrbase - relocatablebase;
1684:C        6       mrbase := mrbase + relocatablesize + ord(odd(relocatablesize));
1685:C        6       globase := mgbase;      globaldelta := mgbase - globalbase;
1686:C        6       mgbase := mgbase - globalsize      - ord(odd(globalsize));
1687:S
1688:C        6       gbytes(unresbits.a, ((extsize div 4 + 15) div 16)*2);
1689:C        6       for i := 2 to extsize div 4 - 1 do unresbits.bmp^[i] := false;
1690:C        6       unresbits.bmp^[0] := true;        unresbits.bmp^[1] := true;
1691:C        6       extaddr.ptp^[0].rp.w := 0;
1692:C        6       extaddr.ptp^[1].rp.w := 4;
1693:S
1694:C        6       sp := directory;
1695:C        6       sp.a := sp.a+sizeof(moduledirectory);
1696:S
1697:C        6       if newmodname.syp = NIL then newmodname := sp;
1698:C        6       if startgvr.p = NIL then
1699:C        7         if executable then
1700:C        8           begin
1701:C        8           startgvrmod := modptr;
1702:C        8           startgvr.a := sp.a+strlen(sp.syp^)+2-ord(odd(strlen(sp.syp^)));
1703:C        8           end;
1704:C        6       end;
1705:C        4     modptr := link;
1706:C        4     end;
1707:C        2   totalreloc :=  mrbase - startreloc;
1708:C        2   totalglobal := startglobal - mgbase;
1709:S
1710:C        2 end; {rsolve}
1711:S
1712:S
1713:D        1 procedure mergeexts;
1714:S
1715:D    -4  2 var ilist:      addrec;
1716:D    -8  2     slist:      sortlistptr;
1717:D   -12  2     sptr:       addrec;
1718:D   -14  2     listlen:    shortint;
1719:D   -16  2     sortlen:    shortint;
1720:D   -18  2     minindex:   shortint;
1721:D   -22  2     modptr:     moddescptr;
1722:D   -22  2     len,
1723:D   -30  2     i:          integer;
1724:D   -31  2     done:       boolean;
1725:D   -31  2     strptr,
1726:D   -40  2     newstrptr:  addrec;
1727:S
1728:C        2 begin
1729:C        2   slist := lowheap.slp;  listlen := 0;
1730:C        2   modptr := newmods;
1731:C        2   while modptr <> NIL do with modptr^ do
1732:C        4     begin
1733:C        4     if not patchmod then if not resolved then
1734:C        6       begin
1735:C        6       listlen := listlen + 1;
1736:C        6       gbytes(sptr.a, sizeof(sortdesc));
1737:C        6       with sptr.sdp^ do
1738:C        7         begin
1739:C        7         modp := modptr;
1740:C        7         N := 0;
```



```
1741:C        7         end;
1742:C        6       end;
1743:C        4     modptr := link;
1744:C        4     end;
1745:S
1746:C        2   gbytes(ilist.a, listlen * sizeof(shortint));
1747:C        2   for i := 1 to listlen do ilist.ilp^[i] := i-1;
1748:C        2   sortlen := listlen;
1749:C        2   gbytes(newexttable, 8);
1750:C        2   newextsize := 8;
1751:S
1752:C        2   while listlen > 0 do
1753:C        3     begin
1754:C        3     while sortlen > 0 do with slist^[ilist.ilp^[sortlen]], modp^ do
1755:C        5       begin
1756:C        5       done := false;
1757:C        5       repeat
1758:C        6        if N >= listsize then
1759:C        7          begin
1760:C        7          listlen := listlen - 1;
1761:C        7          for i := sortlen to listlen do ilist.ilp^[i] := ilist.ilp^[i+1];
1762:C        7          done := true;
1763:C        7          end
1764:C        7        else if listaddr^[N] = 0 then N := N + 1
1765:C        8        else
1766:C        8          begin
1767:C        8          ext := symbolptr(extaddr.a + listaddr^[N]);
1768:C        8          i := sortlen;  minindex := ilist.ilp^[i];
1769:C        8          repeat
1770:C        9            if i >= listlen then done := true
1771:C       10            else if ext^ <= slist^[ilist.ilp^[i+1]].ext^ then done := true
1772:C       11            else begin ilist.ilp^[i] := ilist.ilp^[i+1]; i := i + 1; end;
1773:C        9          until done;
1774:C        8          ilist.ilp^[i] := minindex;
1775:C        8          end;
1776:C        6       until done;
1777:C        5       sortlen := sortlen - 1;
1778:C        5       end;
1779:C        3     if listlen > 0 then
1780:C        4       begin
1781:C        4       strptr.syp := slist^[ilist.ilp^[1]].ext;
1782:C        4       len := strlen(strptr.syp^) + 4 - strlen(strptr.syp^) mod 4;
1783:C        4       gbytes(newstrptr.a, len);
1784:C        4       fastmove(strptr.p, newstrptr.p, len);
1785:C        4       i := 1; done := false;
1786:C        4       repeat with slist^[ilist.ilp^[i]], modp^ do
1787:C        6         if ext^ = newstrptr.syp^ then
1788:C        7           begin
1789:C        7           wordrecptr(ext)^.w := newextsize;
1790:C        7           unresbits.bmp^[listaddr^[N] div 4] := true;
1791:C        7           N := N + 1;
1792:C        7           i := i + 1; done := i > listlen;
1793:C        7           end
1794:C        7         else done := true;
1795:C        5       until done;
1796:C        4       sortlen := i-1;
1797:C        4       newextsize := newextsize + len;
1798:C        4       end;
1799:C        3     end;
1800:S
```

```
1801:C        2   if newextsize <= 8 then newextsize := 0;
1802:S
1803:C        2 end;
1804:S
1805:D        1 function gvrequal(a,b: addrec; offset: integer): boolean;
1806:D        2 var       boff,
1807:D    -8  2           aoff:  integer;
1808:D   -12  2           b0:    gvrptr;
1809:C        2 begin
1810:C        2 gvrequal := false;
1811:C        2 b0 := b.gvp;
1812:C        2 with a.gvp^ do
1813:C        3   if primarytype = b0^.primarytype then
1814:C        4     begin
1815:C        4     if longoffset     then a.a := a.a + 4
1816:C        5                       else a.a := a.a + 2;
1817:C        4     if b0^.longoffset then b.a := b.a + 4
1818:C        5                       else b.a := b.a + 2;
1819:C        4     if valueextend then
1820:C        5       begin
1821:C        5       aoff := a.vep^.value;
1822:C        5       a.a := a.a + sizeof(valueextension, sint);
1823:C        5       end
1824:C        5     else aoff := 0;
1825:C        4     if b0^.valueextend then
1826:C        5       begin
1827:C        5       boff := b.vep^.value;
1828:C        5       b.a := b.a + sizeof(valueextension, sint);
1829:C        5       end
1830:C        5     else boff := 0;
1831:C        4     if aoff + offset = boff then
1832:C        5       if primarytype = general then
1833:C        6         begin
1834:C        6         while (a.rpp^.w
1835:C        7                 = b.rpp^.w)
1836:C        7               and (a.rpp^.last = false) do
1837:C        7           begin
1838:C        7           a.a := a.a + sizeof(referenceptr);
1839:C        7           b.a :=  b.a + sizeof(referenceptr);
1840:C        7           end;
1841:C        6         gvrequal := a.rpp^.w = b.rpp^.w;
1842:C        6         end
1843:C        6       else gvrequal := true;
1844:C        4     end;
1845:C        2 end;
1846:S
1847:D        1 procedure makedir;
1848:S
1849:D    -4  2 var  modptr:            moddescptr;
1850:D    -4  2      newtextrec,
1851:D   -12  2      lasttextrec:       addrec;
1852:D   -14  2      len:               shortint;
1853:D   -14  2      extblocks,
1854:D   -14  2      newtextrecs,
1855:D   -14  2      movebytes,
1856:D   -30  2      textrecs :         integer;
1857:D   -30  2      index,
1858:D   -38  2      newptr:            address;
1859:D   -38  2      tempdirptr,
1860:D   -38  2      oldindex,
```



```
1861:D   -38  2      oldptr,
1862:D   -54  2      ptr:               addrec;
1863:S
1864:S
1865:D        2   procedure mergetext;
1866:D    -1  3   var  merged:  boolean;
1867:D    -1  3        lastptr,
1868:D   -10  3        newptr:  addrec;
1869:C        3   begin
1870:C        3    if newtextrec.tdp^.textsize = 0 then lowheap := newtextrec
1871:C        4    else
1872:C        4     begin
1873:C        4     if lasttextrec.tdp <> NIL then
1874:C        5       begin
1875:C        5       lastptr.a := lasttextrec.a + sizeof(textdescriptor);
1876:C        5       newptr.a  := newtextrec.a  + sizeof(textdescriptor);
1877:C        5       merged := gvrequal(lastptr, newptr, lasttextrec.tdp^.textsize );
1878:C        5       end
1879:C        5     else merged := false;
1880:S
1881:C        4     if merged then
1882:C        5       begin
1883:C        5       lasttextrec.tdp^.textsize := lasttextrec.tdp^.textsize + newtextrec.tdp^.textsize;
1884:C        5       lowheap := newtextrec;
1885:C        5       end
1886:C        5     else
1887:C        5       begin newtextrecs := newtextrecs + 1;
1888:C        5       lasttextrec := newtextrec;
1889:C        5       end;
1890:C        4     end;
1891:C        3   end;
1892:S
1893:C        2 begin {makedir}
1894:C        2   gbytes(tempdirptr.a, sizeof(moduledirectory));
1895:C        2   if newmodname.syp=NIL then
1896:C        3     begin gbytes(newmodname.a, 2);
1897:C        3     newmodname.syp^ := '';
1898:C        3     end
1899:C        3   else
1900:C        3     begin
1901:C        3     len := strlen(newmodname.syp^) + 2 - ord(odd(strlen(newmodname.syp^)));
1902:C        3     gbytes(index, len);
1903:C        3     fastmove(newmodname.p, point(index), len);
1904:C        3     end;
1905:S
1906:C        2   if startgvr.p<>NIL then
1907:C        3     begin
1908:C        3     oldptr := startgvr;
1909:C        3     ptr := lowheap;     makenewgvr(oldptr, startgvrmod);
1910:C        3     compressgvr(ptr);
1911:C        3     end;
1912:S
1913:C        2   lasttextrec.tdp := NIL;
1914:C        2   newtextrecs := 0;
1915:C        2   modptr := newmods;
1916:C        2   while modptr <> NIL do with modptr^ do
1917:C        4     begin
1918:C        4     if patchmod then
1919:C        5       begin
1920:C        5       gbytes(newtextrec.a, sizeof(textdescriptor));
```

```
1921:C        5       newtextrec.tdp^.textsize := patchsize;
1922:C        5       gbytes(oldptr.a, sizeof(generalvalue, false));
1923:C        5       with oldptr.gvp^ do
1924:C        6         begin
1925:C        6         primarytype := relocatable;     datasize := sint;
1926:C        6         patchable := false;             longoffset := false;
1927:C        6         if patchbase = 0 then begin valueextend := false; short := 2; end
1928:C        7         else begin
1929:C        7              gbytes(newptr, sizeof(valueextension, sint));
1930:C        7              veptr(newptr)^.value := patchbase;
1931:C        7              valueextend := true;  short := 6;
1932:C        7              end
1933:C        7         end;
1934:C        5       mergetext;
1935:C        5       end
1936:C        5     else with directory.drp^ do
1937:C        6       begin
1938:C        6       oldindex.a := directory.a + sizeof(moduledirectory);
1939:C        6       oldindex.a := oldindex.a + strlen(oldindex.syp^) + 2 -
1940:C        6                     ord(odd(strlen(oldindex.syp^)));
1941:C        6       if executable then oldindex.a := oldindex.a + oldindex.gvp^.short;
1942:C        6       textrecs := textrecords;
1943:C        6       while textrecs > 0 do with oldindex.tdp^ do
1944:C        8         begin
1945:C        8         gbytes(newtextrec.a, sizeof(textdescriptor));
1946:C        8         if odd(textsize) then textsize := textsize + 1;
1947:C        8         newtextrec.tdp^.textsize := textsize;
1948:C        8         oldindex.a := oldindex.a + sizeof(textdescriptor);
1949:C        8         ptr := lowheap; makenewgvr(oldindex, modptr);
1950:C        8         compressgvr(ptr);
1951:C        8         mergetext;
1952:C        8         textrecs := textrecs - 1;
1953:C        8         end;
1954:C        6       end;
1955:C        4     modptr := link;
1956:C        4     end;
1957:S
1958:C        2   with tempdirptr.drp^ do
1959:C        3     begin
1960:C        3     date := todaysdate;
1961:C        3     revision := linkerdate;
1962:C        3     producer := 'L';
1963:C        3     systemid := 3;
1964:C        3     notice := copyright;
1965:C        3     directorysize := lowheap.a - tempdirptr.a;
1966:C        3     {modulesize := }
1967:C        3     executable := (startgvr.p <> NIL);
1968:C        3     relocatablesize := totalreloc;
1969:C        3     relocatablebase := startreloc;
1970:C        3     globalsize := totalglobal;
1971:C        3     globalbase := startglobal;
1972:C        3     {extblock :=}
1973:C        3     {extsize  :=}
1974:C        3     {defblock := }
1975:C        3     {defsize  := }
1976:C        3     sourceblock := 0;                   {implement later}
1977:C        3     sourcesize  := 0;
1978:C        3     textrecords := newtextrecs;
1979:S
1980:C        3     nextblock := (directorysize +(blocksize-1)) div blocksize;
```



```
1981:S
1982:C        3     extblock :=  nextblock;
1983:C        3     extsize  := newextsize;
1984:C        3     extblocks := (newextsize + (blocksize-1)) div blocksize;
1985:C        3     blockwrite(outfile,point(newexttable)^,extblocks,outblock+nextblock);
1986:C        3     nextblock := nextblock + extblocks;
1987:S
1988:S              {lowheap.a := newdirectory.a + directorysize;
1989:S              fastmove(tempdirptr.p, newdirectory.p, directorysize);
1990:C        3     } newdirectory := tempdirptr;
1991:C        3     end;
1992:S
1993:C        2 end;
1994:S
1995:D        1 procedure mergedefs;
1996:S
1997:D    -4  2 var slist:      sortlistptr;
1998:D    -8  2     ilist:      addrec;
1999:D   -10  2     listlen:    shortint;
2000:D   -12  2     sortlen:    shortint;
2001:D   -14  2     minindex:   shortint;
2002:D   -18  2     modptr:     moddescptr;
2003:D   -18  2     len,
2004:D   -26  2     i:          integer;
2005:D   -27  2     done:       boolean;
2006:D   -27  2     strptr,
2007:D   -27  2     newstrptr,
2008:D   -40  2     sptr:       addrec;
2009:S
2010:D   -44  2     newdeftable: address;
2011:D   -48  2     defblocks:  integer;
2012:D   -49  2     c:          char;
2013:S
2014:C        2 begin
2015:C        2   slist := lowheap.slp;  listlen := 0;
2016:C        2   modptr := newmods;
2017:C        2   while modptr <> NIL do with modptr^ do
2018:C        4     begin
2019:C        4     if not patchmod then if defsize > 0 then
2020:C        6       begin
2021:C        6       listlen := listlen + 1;
2022:C        6       gbytes(sptr.a, sizeof(sortdesc));
2023:C        6       with sptr.sdp^ do
2024:C        7         begin
2025:C        7         modp := modptr;
2026:C        7         def := defaddr;
2027:C        7         end;
2028:C        6       end;
2029:C        4     modptr := link;
2030:C        4     end;
2031:S
2032:C        2   gbytes(ilist.a, listlen * sizeof(shortint));
2033:C        2   for i := 1 to listlen do ilist.ilp^[i] := i-1;
2034:C        2   sortlen := listlen;
2035:C        2   newdeftable := lowheap.a;
2036:S
2037:C        2   while listlen > 0 do
2038:C        3     begin
2039:C        3     while sortlen > 0 do with slist^[ilist.ilp^[sortlen]], modp^ do
2040:C        5       begin
```

```
2041:C        5       done := false;
2042:C        5       repeat
2043:C        6        if def.a >= defaddr.a + defsize then
2044:C        7          begin
2045:C        7          listlen := listlen - 1;
2046:C        7          for i := sortlen to listlen do ilist.ilp^[i] := ilist.ilp^[i+1];
2047:C        7          done := true;
2048:C        7          end
2049:C        7        else
2050:C        7          begin
2051:C        7          len := strlen(def.syp^) + 2 - ord(odd(strlen(def.syp^)));
2052:C        7          with gvrptr(def.a+len)^ do
2053:C        8            if patchable then def.a := def.a + len + short
2054:C        9            else
2055:C        9             begin
2056:C        9             i := sortlen;  minindex := ilist.ilp^[i];
2057:C        9             repeat
2058:C       10              if i >= listlen then done := true
2059:C       11              else if def.syp^ <= slist^[ilist.ilp^[i+1]].def.syp^ then done := true
2060:C       12              else begin ilist.ilp^[i] := ilist.ilp^[i+1]; i := i + 1; end;
2061:C       10             until done;
2062:C        9             ilist.ilp^[i] := minindex;
2063:C        9             end;
2064:C        7          end
2065:C        7       until done;
2066:C        5       sortlen := sortlen - 1;
2067:C        5       end;
2068:C        3     if listlen > 0 then
2069:C        4       begin
2070:C        4       with slist^[ilist.ilp^[1]] do
2071:C        5         begin
2072:C        5         strptr := def;
2073:C        5         len := strlen(strptr.syp^) + 2 - ord(odd(strlen(strptr.syp^)));
2074:C        5         gbytes(newstrptr.a, len);
2075:C        5         fastmove(strptr.p, newstrptr.p, len);
2076:C        5         def.a := strptr.a + len;
2077:C        5         makenewgvr(def, modp);
2078:C        5         end;
2079:C        4       i := 2; done := false;
2080:C        4       repeat with slist^[ilist.ilp^[i]], modp^ do
2081:C        6         if i > listlen then done := true
2082:C        7         else if def.syp^ = newstrptr.syp^ then
2083:C        8          begin
2084:C        8            if printeron then
2085:C        9              begin
2086:C        9                list; writeln(listing,'duplicate symbol definition for:  ',
2087:C        9                                                          def.syp^);  {**!!!!*}
2088:C        9              end
2089:C        9            else
2090:C        9              begin
2091:C        9                errorline;
2092:C        9                writeln('duplicate symbol:  ',def.syp^);
2093:C        9                if streaming then escape(119);
2094:C        9                write('Press ''C'' to continue, any other key to abort ',cteol);
2095:C        9                read(keyboard,c);
2096:C        9                if (c <> 'C') and (c <> 'c') then escape(119);
2097:C        9                fgotoxy(output, 0, 22);
2098:C        9                writeln(cteol);
2099:C        9                write('LINKING ...', cteol);
2100:C        9              end;
```



```
2101:C        8            def.a := def.a + strlen(def.syp^) + 2 - ord(odd(strlen(def.syp^)));
2102:C        8            def.a := def.a + def.gvp^.short;
2103:C        8            i := i + 1;
2104:C        8          end
2105:C        8        else done := true;
2106:C        5       until done;
2107:C        4       sortlen := i-1;
2108:C        4       end;
2109:C        3     end;
2110:S
2111:C        2   with newdirectory.drp^ do
2112:C        3     begin
2113:C        3     defblock := nextblock;
2114:C        3     if defsout then defsize := lowheap.a - newdeftable
2115:C        4     else defsize := 0;
2116:C        3     defblocks := (defsize + (blocksize-1)) div blocksize;
2117:C        3     if defblocks > 0 then
2118:C        4      blockwrite(outfile,point(newdeftable)^,defblocks,outblock+nextblock);
2119:C        3     nextblock := nextblock + defblocks;
2120:C        3     end;
2121:C        2   lowheap.slp := slist;
2122:S
2123:C        2 end;
2124:S
2125:D        1 procedure copytext;
2126:S
2127:D     -4  2 var patchptr:            patchdescptr;
2128:D    -12  2     loadaddr,loadaddr0: address;
2129:D    -16  2     modptr:             moddescptr;     {current module being loaded}
2130:D    -20  2     gvrp:               gvrptr;
2131:D    -20  2     patching,
2132:D    -22  2     merging:            boolean;        {whether text records are combined}
2133:S
2134:D    -22  2     textbuffer,         {base of text record buffer}
2135:D    -22  2     textbuftop,         {end  of text record buffer}
2136:D    -22  2     textindex,          {pointer to next space available in text buffer}
2137:D    -22  2     object,             {object in text record being modified by ref record}
2138:S
2139:D    -22  2     refbuffer,          {base of ref table buffer}
2140:D    -22  2     refbuftop,          {end  of ref table buffer}
2141:D    -22  2     outrefindex,        {pointer to next space available in ref buffer}
2142:D    -22  2     inrefindex,         {pointer to next record in ref buffer to process}
2143:S
2144:D    -22  2     newptr,            {base of new gvr on heap}
2145:D    -22  2     valptr,             {value extension in new gvr on heap}
2146:S
2147:D    -22  2     oldindex,           {pointer to old text descriptors}
2148:D    -22  2     newindex:           {pointer to new text descriptors}
2149:D    -70  2                 addrec;
2150:S
2151:D    -70  2     vevalue,
2152:D    -70  2     newbytes,           {size of new gvr on heap}
2153:D    -70  2     offsetbytes,        {distance from last object referenced by new refs}
2154:D    -70  2     oldtextrec,         {text records left to process from old module}
2155:S
2156:D    -70  2     textbufblocks,      {maximum blocks allocated for text buffer}
2157:D    -70  2     textinblock,        {file relative block index into old text}
2158:D    -70  2     textinsize,         {number of bytes left to read from old text}
2159:D    -70  2     textoutblock,       {file relative block index into new text}
2160:D    -70  2     textoutsize,        {number of bytes processed into new text}
```

```
2161:S
2162:D   -70  2     refbufblocks,       {maximum blocks allocated for ref buffer}
2163:D   -70  2     refinblock,         {file relative block index into old ref table}
2164:D   -70  2     refinsize,          {number of bytes left to read from old ref}
2165:D   -70  2     refoutblock,        {file relative block index into new ref table}
2166:D   -70  2     refoutsize:         {number of bytes processed into new ref table}
2167:D  -126  2                 integer;
2168:S
2169:D        2   procedure starttext;
2170:C        3   begin
2171:C        3   if not merging then with newindex.tdp^ do
2172:C        5     begin
2173:C        5     textstart := nextblock;     textoutblock := nextblock + outblock;
2174:C        5     nextblock := nextblock + (textsize + (blocksize - 1)) div blocksize;
2175:C        5     refoutblock := nextblock + outblock;
2176:S
2177:C        5     textoutsize := 0;           textindex   := textbuffer;
2178:C        5     refoutsize := 0;            outrefindex := refbuffer;
2179:C        5     offsetbytes := 0;           object := textbuffer;
2180:S
2181:C        5     valptr.a := newindex.a + sizeof(textdescriptor);
2182:C        5     patching := (valptr.gvp^.primarytype = relocatable) and (totalpatchspace > 0);
2183:C        5     if patching then
2184:C        6      if valptr.gvp^.valueextend then
2185:C        7       begin
2186:C        7       valptr.a := valptr.a + sizeof(generalvalue, false);
2187:C        7       loadaddr0 := valptr.vep^.value;
2188:C        7       end
2189:C        7      else loadaddr0 := 0;
2190:C        5      loadaddr := object.a - loadaddr0;
2191:C        5     end;
2192:C        3   end;
2193:S
2194:D        2   procedure endtext;
2195:D    -4  3   var lastblocks: integer;
2196:D    -8  3       id: addrec;
2197:D   -12  3       org: integer;
2198:C        3   begin
2199:C        3   with newindex.tdp^ do
2200:C        4     begin
2201:C        4     merging := (textoutsize < textsize);
2202:C        4     if not merging then
2203:C        5       begin
2204:C        5       if textindex.a > textbuffer.a then
2205:C        6         begin
2206:C        6         lastblocks := (textindex.a-textbuffer.a+(blocksize-1)) div blocksize;
2207:C        6         blockwrite(outfile, textbuffer.p^, lastblocks, textoutblock);
2208:C        6         end;
2209:C        5       if outrefindex.a > refbuffer.a then
2210:C        6         begin
2211:C        6         lastblocks := (outrefindex.a - refbuffer.a + (blocksize-1)) div blocksize;
2212:C        6         blockwrite(outfile, refbuffer.p^, lastblocks, refoutblock);
2213:C        6         end;
2214:C        5       refstart := nextblock;    refsize := refoutsize;
2215:C        5       nextblock := nextblock + (refoutsize + (blocksize - 1)) div blocksize;
2216:C        5       newindex.a := newindex.a + sizeof(textdescriptor);
2217:C        5       org := 0; gvrstring(newindex.gvp,org,false,true);
2218:C        5       if printeron then
2219:C        6        begin
2220:C        6        list; writeln(listing,
```



```
2221:C        6         '(load record:  size = ',textsize:1,', load address = ',gvaluestring, ')');
2222:C        6        end;
2223:C        5       end;
2224:C        4     end;
2225:C        3   end;
2226:S
2227:D        2   procedure dumptext(writebytes: integer);
2228:D    -4  3   var writeblocks:      integer;
2229:C        3   begin
2230:C        3     writeblocks := writebytes div blocksize;
2231:C        3     writebytes := writeblocks * blocksize;
2232:C        3     blockwrite(outfile, textbuffer.p^, writeblocks, textoutblock);
2233:C        3     textoutblock := textoutblock + writeblocks;
2234:C        3     textindex.a := textindex.a - writebytes;
2235:C        3     object.a := object.a - writebytes;
2236:C        3     loadaddr := loadaddr - writebytes;
2237:C        3     fastmove(point(textbuffer.a + writebytes), textbuffer.p,
2238:C        3              textindex.a - textbuffer.a);
2239:C        3   end;
2240:S
2241:D        2   procedure checktextbuf(obsize: integer);
2242:D    -8  3   var readbytes, writebytes:      integer;
2243:C        3   begin
2244:C        3   while textindex.a < object.a + obsize do
2245:C        4     begin
2246:C        4     readbytes :=  textbuftop.a - textindex.a;
2247:C        4     if textinsize <= readbytes then readbytes := textinsize
2248:C        5     else
2249:C        5       begin
2250:C        5       if object.a < textindex.a then writebytes := object.a    - textbuffer.a
2251:C        6                                 else writebytes := textindex.a - textbuffer.a;
2252:C        5       if writebytes < readbytes then
2253:C        6         readbytes := readbytes - readbytes mod blocksize
2254:C        6       else begin dumptext(writebytes); readbytes := 0; end;
2255:C        5       end;
2256:C        4     if readbytes > 0 then
2257:C        5       begin
2258:C        5       readblocks(modptr^.filefib.fbp^, textindex.p^, readbytes, textinblock);
2259:C        5       textinblock := textinblock + readbytes div blocksize;
2260:C        5       textinsize := textinsize - readbytes;
2261:C        5       textindex.a := textindex.a + readbytes;
2262:C        5       end;
2263:C        4     end;
2264:C        3   end;
2265:S
2266:D        2   procedure dumprefs;
2267:D    -8  3   var writebytes, writeblocks:    integer;
2268:C        3   begin
2269:C        3     writeblocks := (outrefindex.a - refbuffer.a) div blocksize;
2270:C        3     writebytes := writeblocks * blocksize;
2271:C        3     blockwrite(outfile, refbuffer.p^, writeblocks, refoutblock);
2272:C        3     refoutblock := refoutblock + writeblocks;
2273:C        3     outrefindex.a := outrefindex.a - writebytes;
2274:C        3     fastmove(point(refbuffer.a + writebytes), refbuffer.p,
2275:C        3              outrefindex.a - refbuffer.a);
2276:C        3   end;
2277:S
2278:S
2279:D        2   procedure  checkinref;
2280:D        3   const maxrefsize = 254;
```

```
2281:D    -8  3   var  refinbytes, readbytes:   integer;
2282:C        3   begin
2283:C        3     refinbytes := refbuftop.a - inrefindex.a;
2284:C        3     if refinbytes < maxrefsize then
2285:C        4       begin
2286:C        4       if refinsize > 0 then
2287:C        5         repeat
2288:C        6         if outrefindex.a > refbuffer.a + blocksize then
2289:C        7               readbytes := inrefindex.a - (outrefindex.a + blocksize)
2290:C        7         else readbytes := inrefindex.a - (refbuffer.a + (2 * blocksize));
2291:C        6         if refinsize <= readbytes then readbytes := refinsize
2292:C        7         else if outrefindex.a - refbuffer.a < readbytes then
2293:C        8                readbytes := readbytes - readbytes mod blocksize
2294:C        8              else begin dumprefs; readbytes := 0; end;
2295:C        6         if readbytes > 0 then
2296:C        7           begin
2297:C        7           fastmove(inrefindex.p, point(inrefindex.a - readbytes), refinbytes);
2298:C        7           inrefindex.a := inrefindex.a - readbytes;
2299:C        7           readblocks(modptr^.filefib.fbp^, point(inrefindex.a + refinbytes)^,
2300:C        7                                            readbytes, refinblock);
2301:C        7           refinblock := refinblock + readbytes div blocksize;
2302:C        7           refinsize := refinsize - readbytes;
2303:C        7           end;
2304:C        6         until readbytes > 0;
2305:C        4       end;
2306:C        3   end;
2307:S
2308:D        2   procedure putref;
2309:D    -2  3   var newbytes: shortint;
2310:D    -6  3       valptr:   addrec;
2311:C        3   begin
2312:C        3   compressgvr(newptr);
2313:C        3   with newptr.gvp^ do
2314:C        4     if longoffset then long := offsetbytes
2315:C        5     else if offsetbytes < 256 then short := offsetbytes
2316:C        6     else
2317:C        6       begin
2318:C        6       valptr.a := newptr.a + sizeof(generalvalue,false);
2319:C        6       moveright(valptr.p^, point(valptr.a + 2)^,
2320:C        6                 lowheap.a - valptr.a);
2321:C        6       lowheap.a := lowheap.a + 2;
2322:C        6       longoffset := true;
2323:C        6       long := offsetbytes;
2324:C        6       end;
2325:C        3   offsetbytes := 0;
2326:C        3   newbytes := lowheap.a - newptr.a;
2327:C        3   if outrefindex.a + newbytes > inrefindex.a then dumprefs;
2328:C        3   fastmove(newptr.p, outrefindex.p, newbytes);
2329:C        3   outrefindex.a := outrefindex.a + newbytes;
2330:C        3   refoutsize := refoutsize + newbytes;
2331:C        3   lowheap := newptr;
2332:C        3   end; (putref)
2333:S
2334:D        2   procedure patcherror(dsize: datatype);
2335:S
2336:D        3     procedure printmessage(var f: text);
2337:D    -4  4     var index: addrec;
2338:C        4     begin
2339:C        4     index.a := modptr^.directory.a+sizeof(moduledirectory);
2340:C        4     write(f, 'Can''t patch byte ',
```



```
2341:C        4       object.a - loadaddr - loadaddr0:1,
2342:C        4       ' in text record ',oldtextrec:1,
2343:C        4       ' of module ',index.syp^);
2344:C        4     end;
2345:S
2346:C        3   begin
2347:C        3   errors := errors + 1;
2348:C        3   errorline; printmessage(output);
2349:C        3   if printeron then
2350:C        4     begin list;
2351:C        4     write(listing, '*** ERROR *** ');
2352:C        4     printmessage(listing); writeln(listing);
2353:C        4     end
2354:C        4   else escape(128);
2355:C        3   end;
2356:S
2357:D        2   procedure makepatch;
2358:D    -8  3   var r, rptr:     addrec;
2359:D   -12  3       objectaddr:  address;
2360:D   -16  3       patchaddr:   address;
2361:D   -16  3       foundpatchmodptr,
2362:D   -24  3       patchmodptr: moddescptr;
2363:D   -24  3       foundlastpptr,
2364:D   -24  3       lastpatchptr,
2365:D   -36  3       patchptr:    patchdescptr;
2366:D   -36  3       patchdelta,
2367:D   -48  3       delta2,foundpatchdelta:  integer;
2368:D   -50  3       patchstate: (nopatch,longpatch,shortpatch, oldpatch);
2369:D   -51  3       backwardlist: boolean;
2370:C        3   begin
2371:C        3   objectaddr := object.a - loadaddr;
2372:C        3   with valptr.vep^ do
2373:C        4    value := value + object.sw^ + objectaddr;
2374:C        3   with newptr.gvp^ do
2375:C        4    begin
2376:C        4    patchable := false;
2377:C        4    if primarytype = absolute then primarytype := relocatable
2378:C        5    else
2379:C        5     begin
2380:C        5     if primarytype <> general then
2381:C        6       begin
2382:C        6       gbytes(rptr.a, sizeof(referenceptr));
2383:C        6       with rptr.rpp^ do
2384:C        7        begin adr := 0; op := addit; last := false; end;
2385:C        6       gbytes(rptr.a, sizeof(referenceptr));
2386:C        6       with rptr.rpp^ do
2387:C        7        begin adr := ord(primarytype)-1;
2388:C        7        op := addit; last := true; end;
2389:C        6       primarytype := general;  short := short + 4;
2390:C        6       end
2391:C        6     else
2392:C        6       begin
2393:C        6       rptr.a := valptr.a + sizeof(valueextension,sint);
2394:C        6       with rptr.rpp^ do
2395:C        7        if (adr=0) and (op=subit) then
2396:C        8          if last then
2397:C        9            begin
2398:C        9            primarytype := absolute; lowheap := rptr;
2399:C        9            short := short - 2;
2400:C        9            end
```

```
2401:C         9           else
2402:C         9             begin
2403:C         9             moveleft(point(rptr.a+2)^,rptr.p^,
2404:C         9               short-(valptr.a-newptr.a)-6);
2405:C         9             lowheap.a := lowheap.a - sizeof(referenceptr);
2406:C         9             short := short - 2;
2407:C         9             end
2408:C         9         else
2409:C         8           begin
2410:C         8           gbytes(r.a, sizeof(referenceptr));
2411:C         8           moveright(rptr.p^,point(rptr.a+2)^,
2412:C         8             short-(valptr.a-newptr.a)-4);
2413:C         8           adr := 0; op := addit; last := false;
2414:C         8           short := short + 2;
2415:C         8           end;
2416:C         6       end;
2417:C         5     end;
2418:C         4   end;
2419:C         3   patchstate := nopatch;
2420:C         3   for backwardlist := false to true do
2421:C         4    begin
2422:C         4    if backwardlist then patchmodptr := backwardpatches
2423:C         5                    else patchmodptr := forwardpatches;
2424:C         4    while (patchmodptr <> NIL) and (patchstate < oldpatch) do
2425:C         5     with patchmodptr^ do
2426:C         6      begin
2427:C         6      patchaddr := patchbase;
2428:C         6      patchptr := patchlist;
2429:C         6      while (patchptr<>NIL) and (patchstate < oldpatch) do
2430:C         7       with patchptr^ do
2431:C         8        begin
2432:C         8        patchdelta := patchaddr-objectaddr;
2433:C         8        if (-32768<=patchdelta) and (patchdelta<32768)
2434:C         9         then if gvrequal(newptr,patchref,0)
2435:C        10          then begin
2436:C        10               object.sw^ := patchdelta;
2437:C        10               lowheap := newptr;
2438:C        10               patchstate := oldpatch;
2439:C        10               end;
2440:C         8        if patchref.gvp^.datasize = sword then
2441:C         9            patchaddr := patchaddr + 4
2442:C         9        else patchaddr := patchaddr + 6;
2443:C         8        lastpatchptr := patchptr;
2444:C         8        patchptr := patchlist;
2445:C         8        end;
2446:C         6      patchdelta := patchaddr-objectaddr;
2447:C         6      if (patchstate < shortpatch) then
2448:C         7       if (-32768<=patchdelta) and (patchdelta<32768) then
2449:C         8        begin
2450:C         8        if patchsize - (patchaddr - patchbase) >= 4 then
2451:C         9         if newptr.gvp^.primarytype = relocatable then
2452:C        10          begin
2453:C        10          delta2 := valptr.vep^.value - (patchaddr+2);
2454:C        10          if (-32768 <= delta2) and (delta2 < 32768) then
2455:C        11           if object.a + patchdelta >= textbuffer.a then
2456:C        12            begin
2457:C        12            patchstate := shortpatch;
2458:C        12            foundpatchdelta := patchdelta;
2459:C        12            foundlastpptr := lastpatchptr;
2460:C        12            foundpatchmodptr := patchmodptr;
```



```
2461:C        12            end;
2462:C        10          end;
2463:C         8        if (patchstate < longpatch) and not backwardlist then
2464:C         9         if patchsize - (patchaddr - patchbase) >= 6 then
2465:C        10          begin
2466:C        10          patchstate := longpatch;
2467:C        10          foundpatchdelta := patchdelta;
2468:C        10          foundlastpptr := lastpatchptr;
2469:C        10          foundpatchmodptr := patchmodptr;
2470:C        10          end;
2471:C         8        end;
2472:C         6      patchmodptr := patchlink;
2473:C         6      end;
2474:C         4    end;
2475:C         3   if patchstate = nopatch then patcherror(newptr.gvp^.datasize)
2476:C         4   else if patchstate < oldpatch then
2477:C         5    with foundpatchmodptr^ do
2478:C         6     begin
2479:C         6     gbytes(r.a, sizeof(patchdescriptor));
2480:C         6     if patchlist = NIL then patchlist := r.pdp
2481:C         7     else foundlastpptr^.patchlist := r.pdp;
2482:C         6     with r.pdp^ do
2483:C         7      begin
2484:C         7      patchlist := NIL;
2485:C         7      patchref := newptr;
2486:C         7      if patchstate = longpatch then newptr.gvp^.datasize := sint
2487:C         8      else begin
2488:C         8           newptr.gvp^.datasize := sword;
2489:C         8           if foundpatchdelta < 0 then
2490:C         9            begin
2491:C         9            r.a := object.a + foundpatchdelta;
2492:C         9            r.uw^.w := 24576 {BRA pc relative};
2493:C         9            r.a := r.a + 2;
2494:C         9            r.sw^ := delta2;
2495:C         9            if printeron then
2496:C        10             begin
2497:C        10             list; write(listing, '(backward patch)  BRA ');
2498:C        10             gvrp := patchref.gvp; vevalue := 0;
2499:C        10             gvrstring(gvrp, vevalue, false, true);
2500:C        10             writeln(listing, gvaluestring,
2501:C        10               '':20-strlen(gvaluestring),
2502:C        10               r.a-2-loadaddr:10);
2503:C        10             end;
2504:C         9            end;
2505:C         8           end;
2506:C         7      end;
2507:C         6     object.sw^ := foundpatchdelta;
2508:C         6     end;
2509:C         3   end;
2510:S
2511:C         2 begin {procedure copytext}
2512:C         2   {estimate data structures at 3/2(totalpatchspace) + 1/4(workspace) }
2513:C         2   textbufblocks := ((highheap.a - lowheap.a) * 3   -
2514:C         2                      totalpatchspace          * 6  ) div (blocksize * 4);
2515:S
2516:C         2   refbufblocks := textbufblocks div 4;
2517:C         2   if refbufblocks < 4 then refbufblocks := 4;
2518:S
2519:C         2   textbufblocks := textbufblocks - refbufblocks;
2520:C         2   if textbufblocks < 3 then textbufblocks := 3;
```

```
2521:S
2522:C        2   gbytes(textbuffer.a, textbufblocks * blocksize);
2523:C        2   textbuftop := lowheap;
2524:C        2   gbytes(refbuffer.a,  refbufblocks  * blocksize);
2525:C        2   refbuftop  := lowheap;
2526:S
2527:C        2   newindex.a := newdirectory.a + sizeof(moduledirectory);
2528:C        2   newindex.a := newindex.a + strlen(newindex.syp^) + 2 -
2529:C        2                 ord(odd(strlen(newindex.syp^)));
2530:C        2   if newdirectory.drp^.executable then
2531:C        3     newindex.a := newindex.a + newindex.gvp^.short;
2532:S
2533:C        2   merging := false;
2534:C        2   modptr := newmods;
2535:C        2   while modptr <> NIL do with modptr^ do
2536:C        4     begin
2537:C        4     if patchmod then
2538:C        5       begin
2539:C        5       if printeron then
2540:C        6         begin
2541:C        6         list; writeln(listing, '(patch space)', patchsize:29,patchbase:10);
2542:C        6         end;
2543:C        5       starttext;
2544:C        5       patchptr := patchlist;
2545:C        5       while patchptr <> NIL do with patchptr^, patchref.gvp^ do
2546:C        7         begin
2547:C        7         if textbuftop.a - textindex.a < 6 then
2548:C        8           dumptext(textindex.a - textbuffer.a);
2549:C        7         if printeron then
2550:C        8           begin list;
2551:C        8           gvrp := patchref.gvp; vevalue := 0;
2552:C        8           gvrstring(gvrp, vevalue, false, true);
2553:C        8           end;
2554:C        7         if valueextend then
2555:C        8           begin
2556:C        8           if longoffset then valptr.a := patchref.a +sizeof(generalvalue,true)
2557:C        9           else valptr.a := patchref.a +sizeof(generalvalue,false);
2558:C        8           vevalue := valptr.vep^.value;
2559:C        8           end
2560:C        8         else vevalue := 0;
2561:C        7         if datasize = sword {PC relative branch} then
2562:C        8           begin
2563:C        8           if printeron then
2564:C        9             writeln(listing, '  BRA ',gvaluestring,
2565:C        9                ':26-strlen(gvaluestring), object.a-loadaddr:20);
2566:C        8           object.uw^.w := 24576 {BRA pc relative};
2567:C        8           object.a := object.a + 2;
2568:C        8           object.sw^ := vevalue - (object.a - loadaddr);
2569:C        8           object.a := object.a + 2;
2570:C        8           offsetbytes := offsetbytes + 4;
2571:C        8           end
2572:C        8         else {long absolute branch}
2573:C        8           begin
2574:C        8           if printeron then
2575:C        9             writeln(listing, '  JMP ', gvaluestring,
2576:C        9                ':26-strlen(gvaluestring), object.a-loadaddr:20);
2577:C        8           object.uw^.w := 20217 {JMP long absolute};
2578:C        8           object.a := object.a + 2;
2579:C        8           object.si^ := vevalue;
2580:C        8           object.a := object.a + 4;
```



```
2581:C        8           gbytes(newptr.a, short);
2582:C        8           fastmove(patchref.p, newptr.p, short);
2583:C        8           if valueextend then
2584:C        9             begin
2585:C        9             valptr.a := newptr.a + (valptr.a - patchref.a);
2586:C        9             valptr.vep^.value := 0;
2587:C        9             end;
2588:C        8           offsetbytes := offsetbytes + 2;
2589:C        8           putref;
2590:C        8           offsetbytes := 4;
2591:C        8           end;
2592:C        7         textindex := object;
2593:C        7         patchptr := patchlist;
2594:C        7         end;
2595:C        5       object.a := textindex.a + patchsize - (object.a - loadaddr - patchbase);
2596:C        5       while textindex.a < object.a do
2597:C        6         begin
2598:C        6         if textindex.a >= textbuftop.a - 2 then
2599:C        7           dumptext(textindex.a - textbuffer.a);
2600:C        6         textindex.sw^ := -1;
2601:C        6         textindex.a := textindex.a + 2;
2602:C        6         offsetbytes := offsetbytes + 2;
2603:C        6         end;
2604:C        5       textoutsize := textoutsize + patchsize;
2605:C        5       endtext;
2606:C        5       forwardpatches := patchlink;
2607:C        5       patchlink := backwardpatches;
2608:C        5       backwardpatches := modptr;
2609:C        5       end
2610:C        5     else with directory.drp^ do
2611:C        6       begin
2612:C        6       oldindex.a := directory.a + sizeof(moduledirectory);
2613:C        6       if printeron then
2614:C        7         begin
2615:C        7         list; writeln(listing, oldindex.syp^, '':32-strlen(oldindex.syp^),
2616:C        7                  relocatablesize:10,relocbase:10,
2617:C        7                  globalsize:10,     globase:10);
2618:C        7         end;
2619:C        6       oldindex.a := oldindex.a + strlen(oldindex.syp^) + 2 -
2620:C        6                     ord(odd(strlen(oldindex.syp^)));
2621:C        6       if executable then oldindex.a := oldindex.a + oldindex.gvp^.short;
2622:C        6       for oldtextrec := 1 to textrecords do
2623:C        7         begin
2624:C        7         if oldindex.tdp^.textsize > 0 then
2625:C        8           begin
2626:C        8           starttext;
2627:C        8           loadaddr0 := object.a - loadaddr;
2628:C        8           with oldindex.tdp^ do
2629:C        9             begin
2630:C        9             refinsize := refsize;
2631:C        9             textinsize := textsize;
2632:C        9             refinblock := fileblock+refstart;
2633:C        9             textinblock := fileblock+textstart;
2634:C        9             textoutsize := textoutsize + textsize;
2635:C        9             end;
2636:S
2637:C        8           inrefindex := refbuftop;
2638:S
2639:C        8           while (refbuftop.a - inrefindex.a) + refinsize > 0 do
2640:C        9             begin
```

```
2641:C        9           checkinref;
2642:C        9           newptr := lowheap;
2643:C        9           with inrefindex.gvp^ do
2644:C       10             if longoffset then
2645:C       11               begin
2646:C       11               newbytes := long;
2647:C       11               valptr.a := newptr.a + sizeof(generalvalue, true);
2648:C       11               end
2649:C       11             else
2650:C       11               begin
2651:C       11               newbytes := short;
2652:C       11               valptr.a := newptr.a + sizeof(generalvalue, false);
2653:C       11               end;
2654:C        9           object.a := object.a + newbytes;
2655:C        9           offsetbytes := offsetbytes + newbytes;
2656:C        9           makenewgvr(inrefindex, modptr);
2657:C        9           with newptr.gvp^, valptr.vep^ do
2658:C       10             begin
2659:C       10             case datasize of   {$range off$}
2660:C       11               sbyte:
2661:C       11                 begin
2662:C       11                 checktextbuf(sizeof(sbyterec));
2663:C       11                 value         := value + object.sb^.sb;
2664:C       11                 object.sb^.sb := value;
2665:C       11                 value         := value - object.sb^.sb;
2666:C       11                 end;
2667:C       11               sword:
2668:C       11                 begin
2669:C       11                 checktextbuf(sizeof(shortint));
2670:C       11                 value         := value + object.sw^;
2671:C       11                 object.sw^    := value;
2672:C       11                 value         := value - object.sw^;
2673:C       11                 end;
2674:C       11               sint:
2675:C       11                 begin
2676:C       11                 checktextbuf(sizeof(integer));
2677:C       11                 object.si^ := object.si^ + value;
2678:C       11                 value := 0;
2679:C       11                 end;
2680:C       11               ubyte:
2681:C       11                 begin
2682:C       11                 checktextbuf(sizeof(ubyterec));
2683:C       11                 value         := value + object.ub^.ub;
2684:C       11                 object.ub^.ub := value;
2685:C       11                 value         := value - object.ub^.ub;
2686:C       11                 end;
2687:C       11               uword:
2688:C       11                 begin
2689:C       11                 checktextbuf(sizeof(wordrec));
2690:C       11                 value         := value + object.uw^.w;
2691:C       11                 object.uw^.w  := value;
2692:C       11                 value         := value - object.uw^.w;
2693:C       11                 end;        {$range on$}
2694:C       11               otherwise escape(111);
2695:C       11               end; {case datasize}
2696:C       10             if primarytype = absolute then
2697:C       11               begin
2698:C       11               if value <> 0 then
2699:C       12                 if patchable and patching then makepatch
2700:C       13                 else patcherror(datasize);
```



```
2701:C       11                 end
2702:C       11             else if patching and patchable then makepatch
2703:C       12             else putref;
2704:C       10             end; {with gvrptr(newptr)^, valptr^}
2705:C        9           end; {while there are any ref's }
2706:S
2707:C        8         newbytes := textindex.a + textinsize  - object.a;
2708:C        8         offsetbytes := offsetbytes + newbytes;
2709:C        8         object.a := object.a + newbytes;
2710:C        8         checktextbuf(0);
2711:S
2712:C        8         endtext;
2713:C        8         end;
2714:C        7       oldindex.a := oldindex.a + sizeof(textdescriptor);
2715:C        7       oldindex.a := oldindex.a + oldindex.gvp^.short;
2716:C        7       end; {for oldtextrec}
2717:C        6      end; {with directory^ do}
2718:C        4    modptr := link;
2719:C        4    end;
2720:C        2 end; {copytext}
2721:S
2722:D        1 procedure printdirectentry(modnum: shortint; var entry: direntry);
2723:C        2 begin
2724:C        2 with entry do
2725:C        3   begin
2726:C        3   upc(dtid);
2727:C        3   list; write(listing, modnum:4,' ',dtid,
2728:C        3   dlastblk-dfirstblk:21-strlen(dtid),'  ');
2729:C        3   writedate(listing, daccess);
2730:C        3   writeln(listing, dfirstblk:7);
2731:C        3   end;
2732:C        2 end;
2733:S
2734:D        1 procedure bootmod(modnum: shortint);
2735:D        2 const sectorsize = 256;
2736:D        2 var     buffer, bufptr, valptr, ptr, endrefs, mname,
2737:D   -28  2         infostart:      addrec;
2738:D   -36  2         object, recordnum:  integer;
2739:S
2740:D        2 procedure writesector(anyvar f: fib; anyvar obj: window; size,sector: integer);
2741:C        3 begin
2742:C        3   call (f.am, addr(f), writebytes, obj, size, sector * sectorsize);
2743:C        3   if ioresult <> 0 then escape(114);
2744:C        3 end;
2745:S
2746:C        2 begin
2747:C        2   infostart := lowheap;
2748:C        2   loadinfo(modnum,true, true);
2749:C        2   with newmods^,directory.drp^ do
2750:C        3     begin
2751:C        3     if extsize > 8 then escape(120);
2752:C        3     mname.a := directory.a + sizeof(moduledirectory);
2753:C        3     ptr.a := mname.a + strlen(mname.syp^) + 2 - ord(odd(strlen(mname.syp^)));
2754:C        3     if executable then with ptr.gvp^, fibp(addr(outfile))^ do
2755:C        5       begin
2756:C        5       if fstartaddress = 0 then
2757:C        6         if valueextend then
2758:C        7           begin
2759:C        7           valptr.a := ptr.a + sizeof(generalvalue,false);
2760:C        7           fstartaddress := valptr.vep^.value;
```

```
2761:C        7           end;
2762:C        5       ptr.a := ptr.a + short;
2763:C        5       end;
2764:C        3     recordnum := 0;
2765:C        3     while textrecords > 0 do with ptr.tdp^ do
2766:C        5       begin
2767:C        5       recordnum := recordnum + 1;
2768:C        5       if refsize > 0 then       {check to make sure code is "absolute"}
2769:C        6         begin
2770:C        6         gbytes(buffer.a, refsize);
2771:C        6         readblocks(filefib.fbp^,buffer.p^,refsize,fileblock+refstart);
2772:C        6         bufptr := buffer; endrefs.a := buffer.a + refsize;
2773:C        6         object := 0;
2774:C        6         while bufptr.a < endrefs.a do with bufptr.gvp^ do
2775:C        8           begin
2776:C        8           if longoffset then
2777:C        9                begin object := object + long;
2778:C        9                bufptr.a := bufptr.a + sizeof(generalvalue, true);
2779:C        9                end
2780:C        9           else begin object := object + short;
2781:C        9                bufptr.a := bufptr.a + sizeof(generalvalue, false);
2782:C        9                end;
2783:C        8           if valueextend then
2784:C        9             begin
2785:C        9             errorline;
2786:C        9             write  ('Can''t relocate byte ',object:1,
2787:C        9                     ' in record ',recordnum:1,
2788:C        9                     ' of module ',mname.syp^);
2789:C        9             escape(128);
2790:C        9             end;
2791:C        8           if primarytype = general then
2792:C        9             begin
2793:C        9             while not bufptr.rpp^.last do
2794:C       10               bufptr.a := bufptr.a + sizeof(referenceptr);
2795:C        9             bufptr.a := bufptr.a + sizeof(referenceptr);
2796:C        9             end;
2797:C        8           end;
2798:C        6         lowheap := buffer;
2799:C        6         end;
2800:C        5       gbytes(buffer.a, sizeof(integer));
2801:C        5       ptr.a := ptr.a + sizeof(textdescriptor);
2802:C        5       with ptr.gvp^ do
2803:C        6         begin
2804:C        6         if valueextend then
2805:C        7           begin
2806:C        7           valptr.a := ptr.a + sizeof(generalvalue,false);
2807:C        7           buffer.p^ := valptr.vep^.value;
2808:C        7           end
2809:C        7         else buffer.p^ := 0;
2810:C        6         ptr.a := ptr.a + short;
2811:C        6         end;
2812:S
2813:C        5       gbytes(bufptr.a, sizeof(integer));
2814:C        5       bufptr.p^ := textsize;
2815:S
2816:C        5       gbytes(bufptr.a, textsize);
2817:C        5       readblocks(filefib.fbp^,bufptr.p^,textsize,fileblock+textstart);
2818:C        5       writesector(outfile, buffer.p^, textsize+2*sizeof(integer), outblock);
2819:C        5       outblock := outblock +
2820:C        5                 (textsize + 2*sizeof(integer) + (sectorsize-1)) div sectorsize;
```



```
2821:C        5       lowheap := buffer;
2822:C        5       textrecords := textrecords - 1;
2823:C        5       end;
2824:C        3     end;
2825:C        2   lowheap := infostart;
2826:C        2   newmods := NIL;
2827:C        2 end;
2828:S
2829:D        1 procedure copymodule(modnum: shortint);
2830:D        2 const fosize = sizeof(addrec)+sizeof(fib,1);
2831:D        2       fsize  = fosize+ord(odd(fosize));
2832:D    -6  2 var startblock, numblocks, transblocks:        shortint;
2833:D   -10  2     copybuffer: addrec;
2834:D   -12  2     bufblocks:  shortint;
2835:C        2 begin
2836:C        2 if booting then bootmod(modnum)
2837:C        3 else if linking then begin
2838:C        4                      if loadfib.a >= highheap.a then
2839:C        5                        begin
2840:C        5                        fastmove(highheap.p, lowheap.p, fsize);
2841:C        5                        highheap.a := highheap.a + fsize;
2842:C        5                        lowheap.a := lowheap.a + fsize;
2843:C        5                        loadfib.a := loadfib.a - (highheap.a - lowheap.a);
2844:C        5                        end;
2845:C        4                      loadinfo(modnum, true, true)
2846:C        4                      end
2847:C        4 else
2848:C        4   begin
2849:C        4   bufblocks := (highheap.a -lowheap.a) div blocksize;
2850:C        4   gbytes(copybuffer.a, bufblocks * blocksize);
2851:S
2852:C        4   if outmodnum>=maxmodules then escape(127);
2853:C        4   outmodnum := outmodnum + 1;
2854:C        4   outdirectory.fdp^[outmodnum] := fdirectory^[modnum];
2855:C        4   with fdirectory^[modnum] do
2856:C        5     begin
2857:C        5     startblock := dfirstblk;      numblocks := dlastblk-startblock;
2858:C        5     end;
2859:C        4   with outdirectory.fdp^[outmodnum] do
2860:C        5     begin
2861:C        5     dfirstblk := outblock;        dlastblk := outblock + numblocks;
2862:C        5     end;
2863:C        4   while numblocks > 0 do
2864:C        5     begin
2865:C        5     if numblocks <= bufblocks then transblocks := numblocks
2866:C        6                               else transblocks := bufblocks;
2867:C        5     readblocks(loadfib.fbp^, copybuffer.p^, transblocks*blocksize, startblock);
2868:C        5     blockwrite(outfile, copybuffer.p^, transblocks, outblock);
2869:C        5     startblock := startblock + transblocks;
2870:C        5     outblock := outblock + transblocks;
2871:C        5     numblocks := numblocks - transblocks;
2872:C        5     end;
2873:C        4     lowheap := copybuffer;
2874:C        4     {if printeron then printdirectentry(outmodnum, outdirectory.fdp^[outmodnum]); }
2875:C        4   end;
2876:C        2 end;
2877:S
2878:D        1 procedure writedirectory;
2879:C        2 begin
2880:C        2   with newdirectory.drp^ do
```

```
2881:C        3   begin
2882:C        3     modulesize := nextblock * blocksize;
2883:C        3     blockwrite(outfile, newdirectory.drp^, extblock, outblock);
2884:C        3   end;
2885:S
2886:C        2   if outmodnum>=maxmodules then escape(127);
2887:C        2   outmodnum := outmodnum + 1;
2888:C        2   with outdirectory.fdp^[outmodnum] do
2889:C        3     begin
2890:C        3     dfirstblk := outblock;
2891:C        3     outblock := outblock + nextblock;
2892:C        3     dlastblk := outblock;
2893:C        3     dfkind := codefile;
2894:C        3     moveleft(newmodname.syp^, dtid, sizeof(filname));
2895:C        3     if strlen(dtid) > fnlength then setstrlen(dtid, fnlength);
2896:C        3     dlastbyte := 256;
2897:C        3     daccess := todaysdate;
2898:C        3     end;
2899:C        2   if printeron then printdirectentry(outmodnum, outdirectory.fdp^[outmodnum]); }
2900:C        2 end;
2901:S
2902:D        1 procedure trim(var s: string);
2903:D    -4  2 var first, last: shortint;
2904:C        2 begin
2905:C        2   last := strlen(s);
2906:C        2   while last > 0 do
2907:C        3     begin
2908:C        3     if s[last] = ' ' then
2909:C        4       begin last := last - 1; if last = 0 then s := ''; end
2910:C        4     else
2911:C        4       begin
2912:C        4       first := 1;  while s[first] = ' ' do first := first + 1;
2913:C        4       s := str(s, first, last - first + 1);  last := 0;
2914:C        4       end;
2915:C        3     end;
2916:C        2 end;
2917:S
2918:D        1 procedure toggleprinter;
2919:D   -82  2 var newlistname: string80;
2920:D  -100  2     fvid: vid;
2921:D  -222  2     ftitle: fid;
2922:D  -226  2     fsegs:  integer;
2923:D  -228  2     fkind:  filekind;
2924:C        2 begin
2925:C        2 printeron := not printeron;
2926:C        2 if printeron then
2927:C        3   begin
2928:C        3   fgotoxy(output, 13,3); write('     ',cteol);
2929:C        3   readln(newlistname);
2930:C        3   fixname(newlistname, textfile);
2931:C        3   if scantitle(newlistname, fvid, ftitle, fsegs, fkind) then ; { jws 3/2/84}
2932:C        3   if strlen(newlistname)>0 then
2933:C        4     begin
2934:C        4     listfilename := newlistname;
2935:C        4     if fsegs=0 then                                          { jws 3/2/84 }
2936:C        5           sappend(newlistname, '[*]');
2937:C        4     pageeject;
2938:C        4     if (pagenum=0) and (linenum=0)
2939:C        5        then close(listing)
2940:C        5        else close(listing, 'lock');
```



```
2941:C        4     rewrite(listing, newlistname);
2942:C        4     pagenum := 0;       linenum := 0;
2943:C        4     printopen := ioresult = 0;
2944:C        4     printeron := printopen;
2945:C        4     if not printopen then escape(118);
2946:C        4     end
2947:C        4   else printeron := printopen;
2948:C        3   end;
2949:C        2 end;
2950:S
2951:D        1 procedure copyon;
2952:C        2 begin
2953:C        2  lowheap := infostart;
2954:C        2  linking := false;
2955:C        2 end;
2956:S
2957:D        1 procedure closein;
2958:C        2 begin
2959:C        2  if fdirectory <> NIL then
2960:C        3   begin
2961:C        3   if loadfib.a >= highheap.a then
2962:C        4     begin
2963:C        4     close(loadfib.php^);
2964:C        4     loadfib.a := loadfib.a - sizeof(addrec);
2965:C        4     loadfib := loadfib.arp^;
2966:C        4     end;
2967:C        3    highheap := highheap0; fdirectory := NIL;
2968:C        3    vmodnum := 0; verifying := false;
2969:C        3   end;
2970:C        2 end;
2971:S
2972:D        1 procedure initlink;
2973:C        2 begin
2974:C        2  linking := true;       defsout := true;
2975:C        2  infostart := lowheap;  newmodname.syp := NIL;
2976:C        2  startreloc := 0;       startglobal := 0;
2977:C        2  copyright := '';
2978:C        2 end;
2979:S
2980:D        1 procedure link;
2981:C        2 begin
2982:C        2  errors := 0;
2983:C        2  fgotoxy(output, 0,23); write('LINKING ...');
2984:C        2  rsolve;
2985:C        2  if printeron then
2986:C        3    begin
2987:C        3    list; writeln(listing, 'link map', 'Rsize':34, 'Rbase':10, 'Gsize':10, 'Gbase':10);
2988:C        3    list; writeln(listing, '------':42, '------':10, '------':10, '------':10);
2989:C        3    end;
2990:C        2  newdirectory := lowheap;
2991:C        2  mergeexts;
2992:C        2  makedir;        {also write new ext table, move down directory}
2993:C        2  mergedefs;
2994:C        2  copytext;
2995:C        2  writedirectory;
2996:S
2997:C        2  if printeron then
2998:C        3    begin
2999:C        3    list; writeln(listing, '------':42,'------':20);
3000:C        3    list;
```

```
3001:C        3    if newmodname.syp = NIL then write(listing, '(no name)', '':32-9)
3002:C        4    else write(listing, newmodname.syp^, '':32-strlen(newmodname.syp^));
3003:C        3    writeln(listing, totalreloc:10, totalglobal:20);
3004:C        3    listln;
3005:C        3    end;
3006:C        2  closein;
3007:C        2  closefiles;
3008:C        2  lowheap := infostart; (release memory used by linker)
3009:C        2  linking := false;      newmods := NIL;
3010:C        2  if errors > 0 then escape(122);
3011:C        2 end; (link)
3012:S
3013:S
3014:D        1 procedure printdirectory;
3015:D    -2  2 var numfiles:   shortint;
3016:D    -4  2     modnum:     shortint;
3017:C        2 begin
3018:C        2  list; writeln(listing, 'FILE DIRECTORY OF:  ''', loadfib.fbp^.ftid, '''');
3019:C        2  listln;
3020:C        2  numfiles := fdirectory^[0].dnumfiles;
3021:C        2  for modnum := 1 to numfiles do
3022:C        3    printdirectentry(modnum, fdirectory^[modnum]);
3023:C        2  listln;
3024:C        2 end;
3025:S
3026:D        1 procedure copyfile;
3027:D    -2  2 var modnum:     shortint;
3028:C        2 begin
3029:C        2  for modnum := 1 to fdirectory^[0].dnumfiles do
3030:C        3   copymodule(modnum);
3031:C        2  closein;
3032:C        2 end;
3033:S
3034:D        1 procedure verifynext;
3035:C        2 begin
3036:C        2  if vmodnum < fdirectory^[0].dnumfiles then
3037:C        3   begin
3038:C        3   vmodnum := vmodnum + 1;
3039:C        3   upc(fdirectory^[vmodnum].dtid)
3040:C        3   end
3041:C        3  else
3042:C        3   begin
3043:C        3   vmodnum := 0;
3044:C        3   verifying := false;
3045:C        3   end;
3046:C        2 end;
3047:S
3048:D        1 procedure verifymod;
3049:C        2 begin
3050:C        2  vmodnum := 0;  verifying := true;
3051:C        2  verifynext;
3052:C        2 end;
3053:S
3054:D        1 procedure xfer;
3055:D    -2  2 var modnum:     shortint;
3056:C        2 begin
3057:C        2  if verifying then
3058:C        3   begin
3059:C        3   copymodule(vmodnum);
3060:C        3   verifynext;
```



```
3061:C        3   end
3062:C        3  else
3063:C        3   begin
3064:C        3   for modnum := 1 to fdirectory^[0].dnumfiles do
3065:C        4    with fdirectory^[modnum] do
3066:C        5     begin
3067:C        5     upc(dtid);
3068:C        5     if dtid = fdirectory^[vmodnum].dtid then
3069:C        6       copymodule(modnum);
3070:C        5     end;
3071:C        3   vmodnum := 0;
3072:C        3   end;
3073:C        2 end;
3074:S
3075:D        1 procedure openin;
3076:D        2 const fosize = sizeof(addrec)+sizeof(fib,1);
3077:D        2       fsize  = fosize+ord(odd(fosize));
3078:C        2 begin
3079:C        2  closein;
3080:C        2  fgotoxy(output, 22,13); write(cteol);
3081:C        2  if strlen(infilename)=0 then   { if no name then get it }
3082:C        3  begin readln(infilename); fixname(infilename, codefile);
3083:C        3  end;
3084:C        2  if strlen(infilename) > 0 then
3085:C        3  begin
3086:C        3    openlinkfile(infilename);
3087:C        3    if fdirectory = NIL then
3088:C        4    begin errorline;
3089:C        4          write('cannot open ''', infilename, ''', ');
3090:C        4          ioerror;
3091:C        4    end
3092:C        4    else
3093:C        4    begin highheap.a := highheap.a - fsize;
3094:C        4          lowheap.a := lowheap.a - fsize;
3095:C        4          fastmove(lowheap.p, highheap.p, fsize);
3096:C        4          loadfib.a := loadfib.a + (highheap.a - lowheap.a);
3097:C        4          if fdirectory^[0].dnumfiles = 1 then vmodnum := 1
3098:C        5                                          else verifymod;
3099:C        4     end;
3100:C        3   end;
3101:C        2 end;   { openin }
3102:S
3103:D        1 procedure closeout;
3104:C        2 begin
3105:C        2  closein;
3106:C        2  with outdirectory.fdp^[0] do
3107:C        3   begin
3108:C        3   deovblk := outblock;
3109:C        3   dnumfiles := outmodnum;
3110:C        3   outopen := false;     outmodnum := 0;
3111:C        3   lowheap := outdirectory;
3112:C        3   blockwrite(outfile, outdirectory.fdp^, outdirectsize, 0);
3113:C        3   close(outfile, 'lock');
3114:C        3   if ioresult <> 0 then escape(126);
3115:C        3   end;
3116:C        2 end;
3117:S
3118:D        1 procedure openout(boot: boolean);
3119:D    -8  2 var i,j: integer;
3120:D   -18  2     nul: string[1]; typestring: string[6];
```

```
3121:D   -30  2     thirdparm: string[10];
3122:C        2 begin
3123:C        2  thirdparm := 'shared';
3124:C        2  if linking then lowheap := infostart;
3125:C        2  linking := false;
3126:C        2  if outopen then
3127:C        3    begin
3128:C        3    close(outfile);
3129:C        3    lowheap := outdirectory;
3130:C        3    end;
3131:C        2  outopen := false;
3132:C        2  fgotoxy(output, 22,4); write(cteol);
3133:C        2  readln(outfilename);
3134:C        2  trim(outfilename);
3135:C        2  if strlen(outfilename) > 0 then
3136:C        3   begin
3137:C        3   mul := '';
3138:C        3   if boot then
3139:C        4    begin
3140:C        4    fixname(outfilename, sysfile);
3141:C        4    reset(outfile, outfilename); close(outfile, 'PURGE');
3142:C        4    typestring := '.SYSTM'; fmaketype(outfile, outfilename, thirdparm, typestring);
3143:C        4    outopen := (ioresult = 0);
3144:C        4    outblock := 0;
3145:C        4    end
3146:C        4   else
3147:C        4    begin
3148:C        4    fixname(outfilename, codefile);
3149:C        4    typestring := '.CODE'; fmaketype(outfile, outfilename, thirdparm, typestring);
3150:C        4    if ioresult = 0 then
3151:C        5     begin
3152:C        5     gbytes(outdirectory.a, outdirectsize*blocksize);
3153:C        5     with outdirectory.fdp^[0] do
3154:C        6       begin
3155:C        6       dfirstblk := 0; dlastblk := outdirectsize;
3156:C        6       dfkind := untypedfile {volume entry};
3157:C        6       moveleft(outfilename, dvid, sizeof(volname));
3158:C        6       if strlen(outfilename) > vnlength then setstrlen(dvid, vnlength);
3159:C        6       deovblk := outdirectsize; dnumfiles := 0;
3160:C        6       dloadtime := 0; dlastboot := todaysdate;
3161:C        6       end;
3162:C        5     outblock := outdirectsize;
3163:C        5     outopen := true;
3164:C        5     end;
3165:C        4    end;
3166:C        3   if outopen then booting := boot
3167:C        4   else
3168:C        4    begin
3169:C        4    booting := false;
3170:C        4    errorline;
3171:C        4    write('cannot open ''', outfilename, ''', ');
3172:C        4    ioerror;
3173:C        4    end;
3174:C        3   end;
3175:C        2 end;    { openout }
3176:S
3177:D        1 procedure setmaxmodules;
3178:D    -8  2 var total, excess: integer;
3179:C        2 begin
3180:C        2  fgotoxy(output, 30,6); write(cteol);
```



```
3181:C        2  if readint(maxmodules) then
3182:C        3    begin
3183:C        3    if maxmodules > 300000 then
3184:C        4      begin
3185:C        4      maxmodules := 38;
3186:C        4      escape(125);
3187:C        4      end;
3188:C        3    if maxmodules <= 0 then maxmodules := 0;
3189:C        3    outdirectsize := ((maxmodules+1)*entrysize+(blocksize-1)) div blocksize;
3190:C        3    maxmodules := outdirectsize*blocksize div entrysize - 1;
3191:C        3    end;
3192:C        2 end;
3193:S
3194:D        1 procedure setreloc;
3195:C        2 begin
3196:C        2  fgotoxy(output, 21,7); write(cteol);
3197:C        2  if readint(startreloc) then ;
3198:C        2 end;
3199:S
3200:D        1 procedure setglobal;
3201:C        2 begin
3202:C        2  fgotoxy(output, 21,8); write(cteol);
3203:C        2  if readint(startglobal) then ;
3204:C        2 end;
3205:S
3206:D        1 procedure setcopyright;
3207:C        2 begin
3208:C        2   fgotoxy(output, 0,12); write(cteol);
3209:C        2   fgotoxy(output, 22,11); write(cteol);
3210:C        2   readln(copyright);
3211:C        2 end;
3212:S
3213:D        1 procedure makepatchspace;
3214:D    -4  2 var pmod:       addrec;
3215:C        2 begin
3216:C        2  fgotoxy(output, 23,9); write(cteol);
3217:C        2  if readint(patchbytes) then
3218:C        3   if patchbytes > 0 then
3219:C        4    begin
3220:C        4    patchbytes := patchbytes + ord(odd(patchbytes));
3221:C        4    gbytes(pmod.a, sizeof(moduledescriptor,true));
3222:C        4    with pmod.mdp^ do
3223:C        5      begin
3224:C        5      patchmod := true;   patchsize := patchbytes;
3225:C        5      link := newmods;    newmods := pmod.mdp;
3226:C        5      patchlink := NIL;   patchlist := NIL;
3227:C        5      end;
3228:C        4    end;
3229:C        2 end;
3230:S
3231:D        1 procedure setname;
3232:D   -82  2 var s: string80;
3233:C        2 begin
3234:C        2  fgotoxy(output, 24,6); write(cteol);
3235:C        2  readln(s); trim(s);
3236:C        2  if strlen(s)=0 then newmodname.syp := NIL
3237:C        3  else
3238:C        3   begin
3239:C        3   upc(s);
3240:C        3   gbytes(newmodname.a, strlen(s)+2-ord(odd(strlen(s))));
```

```
3241:C        3   moveleft(s, newmodname.syp^, strlen(s)+1);
3242:C        3   end;
3243:C        2 end;    {setname}
3244:S
3245:D        1 procedure openmod;
3246:D   -82  2 var s: string80;
3247:D   -84  2     i: shortint;
3248:C        2 begin
3249:C        2   verifying := false;   vmodnum := 0;
3250:C        2   fgotoxy(output, 18,18); write(cteol);
3251:C        2   readln(s); trim(s);
3252:C        2   if strlen(s) > 0 then
3253:C        3    begin
3254:C        3    upc(s);          i := 1;
3255:C        3    while (i <= fdirectory^[0].dnumfiles) and (vmodnum = 0) do
3256:C        4     with fdirectory^[i] do
3257:C        5      begin
3258:C        5      upc(dtid);
3259:C        5      if s = dtid then vmodnum := i
3260:C        6      else i:= i + 1;
3261:C        5      end;
3262:C        3    if vmodnum = 0 then
3263:C        4      begin
3264:C        4      errorline; write('module ''', s,''' not found in file');
3265:C        4      escape(123);
3266:C        4      end;
3267:C        3    end;
3268:C        2 end;    {openmod}
3269:S
3270:D        1 procedure findmod(var s:filname; var n:shortint);
3271:D    -4  2 var i : integer;
3272:C        2 begin
3273:C        2   if strlen(s)=0 then   { if no name given then get it }
3274:C        3   begin readln(s); trim(s); end;
3275:C        2   n:= 0;
3276:C        2   if strlen(s) > 0 then
3277:C        3   begin
3278:C        3     upc(s);          i := 1;
3279:C        3     while (i <= fdirectory^[0].dnumfiles) and (n = 0) do
3280:C        4       with fdirectory^[i] do
3281:C        5       begin
3282:C        5         upc(dtid);
3283:C        5         if s = dtid then n := i else i:= i + 1;
3284:C        5       end;
3285:C        3     if n = 0 then n:=-1;        { signal not found }
3286:C        3   end;
3287:C        2 end;    { findmod }
3288:S
3289:D        1 procedure clear(N: shortint);
3290:C        2 begin
3291:C        2   repeat writeln(cteol); n := n - 1; until n <= 0;
3292:C        2 end;
3293:S
3294:D        1 procedure none;
3295:C        2 begin write('(none)'); clear(1); end;
3296:S
3297:D        1 procedure doedit;
3298:D        2 var
3299:D   -48  2   firstmodname,untilmodname, oldvmodname : filname;
3300:D   -60  2   firstmodnum, untilmodnum,  oldvmodnum  : integer;
```



```
3301:D  -142  2   oldfilename: string80;
3302:D  -398  2   modlist     : string255;
3303:D  -399  2   assoc : boolean;
3304:D  -404  2   lc    : string[4];
3305:D  -420  2   tempf : filname;
3306:D  -422  2   im    : shortint;
3307:S
3308:D        2   procedure checkassoc;
3309:C        3   begin
3310:C        3     assoc:=assoc and (vmodnum>0);
3311:C        3     if assoc then
3312:C        4     begin firstmodname:=fdirectory^[vmodnum].dtid;
3313:C        4           firstmodnum:=vmodnum;
3314:C        4           assoc:=assoc and (firstmodnum<>untilmodnum);
3315:C        4     end;
3316:C        3   end;
3317:D        2   procedure outoforder;
3318:C        3   begin errorline; write('module ',tempf,' out of order'); dobeep;
3319:C        3   end;
3320:D        2   procedure mnotfound;
3321:C        3   begin errorline; write('module ',tempf,' not found'); dobeep;
3322:C        3   end;
3323:S
3324:C        2 begin   { doedit }
3325:C        2   untilmodname := '(end of file)';
3326:C        2   untilmodnum  := fdirectory^[0].dnumfiles+1;
3327:C        2   if vmodnum=0 then firstmodname:='(none)'
3328:C        3                    else firstmodname := fdirectory^[vmodnum].dtid;
3329:C        2   firstmodnum  := vmodnum;
3330:C        2   assoc := true;
3331:C        2   fgotoxy(output,0,2); write(cteos);
3332:C        2   repeat
3333:C        3     fgotoxy(output,0,2);
3334:C        3     writeln('S  Stop editing');
3335:C        3     clear(2);
3336:C        3     if firstmodnum>0 then
3337:C        4        writeln('C  Copy First module upto Until module',cteol)
3338:C        4     else clear(1);
3339:C        3     writeln('F  First module: ',firstmodname,cteol);
3340:C        3     writeln('U  Until module: ',untilmodname,cteol);
3341:C        3     clear(1);
3342:C        3     writeln('A  Append module(s)'); clear(5);
3343:C        3     fgotoxy(output,0,18);
3344:C        3     write('M  input Module:  ');
3345:C        3     if vmodnum = 0 then begin none; clear(3); end
3346:C        4     else
3347:C        4     begin
3348:C        4       writeln(fdirectory^[vmodnum].dtid,cteol);
3349:C        4       if booting then      lc := 'boot'
3350:C        5       else if linking then lc := 'link'
3351:C        6                            else lc := 'copy';
3352:C        4       writeln;
3353:C        4       writeln('T  Transfer (',lc,') module',cteol);
3354:C        4       writeln('<space> to continue verifying',cteol);
3355:C        4     end;
3356:C        3     getcommandchar('Edit option?',commandchar);
3357:C        3     case commandchar of
3358:C        4     'A':begin
3359:C        4           oldfilename:= infilename;     { save current inputfile name }
3360:C        4           oldvmodnum := vmodnum;        { same current module number & name }
```

```
 3361:C        4           oldvmodname := fdirectory^[vmodnum].dtid;
 3362:C        4           fgotoxy(output,0,13); write('        Input  file:  ',cteol);
 3363:C        4           setstrlen(infilename,0);
 3364:C        4           openin;       { get new input file }
 3365:C        4           if strlen(infilename)>0 then begin       { 3.0 BUG FIX -- 4/11/84 }
 3366:C        5             { get list of modules and copy them }
 3367:C        5             fgotoxy(output,0,10);
 3368:C        5             writeln('enter list of modules or = for all');
 3369:C        5             readln(modlist); trim(modlist); upc(modlist);
 3370:C        5             if modlist='=' then
 3371:C        6             begin { all modules }
 3372:C        6               for im:=1 to fdirectory^[0].dnumfiles do
 3373:C        7               begin
 3374:C        7                 fgotoxy(output,0,11); write(fdirectory^[im].dtid,cteol);
 3375:C        7                 copymodule(im);
 3376:C        7               end;
 3377:C        6             end
 3378:C        6             else
 3379:C        6             while strlen(modlist)>0 do
 3380:C        7             begin
***WARNING: (line 3381): STRPOS does not conform to HP standard, see $SWITCH_STRPOS$
 3381:C        7               im:=strpos(',',modlist);
 3382:C        7               if im=0 then im:=strlen(modlist)+1;
 3383:C        7               try
 3384:C        8                 if im>sizeof(tempf) then escape(129)
 3385:C        9                                     else tempf:=str(modlist,1,im-1);
 3386:C        8                 if im>strlen(modlist) then setstrlen(modlist,0)
 3387:C        9                                       else strdelete(modlist,1,im);
 3388:C        8                 if strlen(tempf)>0 then
 3389:C        9                 begin     { find the module and copy it }
 3390:C        9                   findmod(tempf,vmodnum);
 3391:C        9                   if vmodnum>0 then copymodule(vmodnum) else escape(123);
 3392:C        9                   fgotoxy(output,0,11); write(modlist,cteol);
 3393:C        9                 end;
 3394:C        8               recover
 3395:C        8               begin
 3396:C        8                 im:=escapecode;  errorline;
 3397:C        8                 case im of
 3398:C        9                   123: writeln('module ',tempf,' not found');
 3399:C        9                   129: writeln('invalid module name');
 3400:C        9                   otherwise escape(im)
 3401:C        9                 end; { case im }
 3402:C        8                 dobeep; setstrlen(modlist,0);     { zap module list to force exit }
 3403:C        8               end;{ end recover}
 3404:C        7             end;  {while list not empty}
 3405:S
 3406:C        5             if not streaming then                 { 3.0 bug fix -- 4/9/84 jws }
 3407:C        6               repeat getcommandchar('Append done, <space> to continue',commandchar);
 3408:C        7               until commandchar=' ';
 3409:C        5           end;                                    { 3.0 BUG FIX -- 4/11/84 }
 3410:S
 3411:C        4           infilename := oldfilename;
 3412:C        4           openin;       { reopen the old file & find old input module }
 3413:C        4           if oldvmodnum=0 then vmodnum:=0
 3414:C        5           else
 3415:C        5           begin
 3416:C        5             findmod(oldvmodname,vmodnum);
 3417:C        5             if (vmodnum<>oldvmodnum) then
 3418:C        6             begin errorline;
 3419:C        6                   write('unable to find old input module ',oldvmodname);
```



```
 3420:C        6                   vmodnum:=0; dobeep;
 3421:C        6             end;
 3422:C        5           end;
 3423:C        4         end;
 3424:C        4     'C':if (firstmodnum>0) and (firstmodnum<untilmodnum) then
 3425:C        5         begin
 3426:C        5           for im:=firstmodnum to untilmodnum-1 do
 3427:C        6           begin fgotoxy(output,0,8);
 3428:C        6                 write('now copying ',fdirectory^[im].dtid,cteol);
 3429:C        6                 copymodule(im);
 3430:C        6           end;
 3431:C        5           if assoc then
 3432:C        6           begin
 3433:C        6             if untilmodnum>fdirectory^[0].dnumfiles
 3434:C        7                 then begin vmodnum := 0;
 3435:C        7                            firstmodname := '(none)';
 3436:C        7                      end
 3437:C        7                 else begin vmodnum := untilmodnum;
 3438:C        7                            firstmodname := fdirectory^[vmodnum].dtid;
 3439:C        7                      end;
 3440:C        6             firstmodnum  := vmodnum;
 3441:C        6             assoc:= assoc and (vmodnum>0);
 3442:C        6           end;
 3443:C        5         end
 3444:C        5         else dobeep;
 3445:C        4     'F':begin
 3446:C        4           fgotoxy(output,17,6); write(cteol);
 3447:C        4           setstrlen(tempf,0); findmod(tempf,im);
 3448:C        4           case im of
 3449:C        5           -1: mnotfound;
 3450:C        5            0: { no module name given, so use default }
 3451:C        5               begin if (vmodnum>0) and (vmodnum<=untilmodnum) then
 3452:C        6                 begin firstmodname := fdirectory^[vmodnum].dtid;
 3453:C        6                       firstmodnum  := vmodnum;
 3454:C        6                       assoc := assoc and (vmodnum<untilmodnum);
 3455:C        6                 end
 3456:C        6                 else dobeep;
 3457:C        5               end;
 3458:C        5           otherwise     { found the module }
 3459:C        5             if im<=untilmodnum then
 3460:C        6             begin firstmodname := tempf; firstmodnum:=im; assoc:=false;
 3461:C        6             end
 3462:C        6             else outoforder;
 3463:C        5           end;  { case im }
 3464:C        4         end;
 3465:C        4     'M': if fdirectory<>NIL then
 3466:C        5          begin openmod;
 3467:C        5            if vmodnum>0 then begin        { 3.0 BUG # 57   4/10/84 }
 3468:C        6              assoc := assoc and (vmodnum<untilmodnum);
 3469:C        6              if assoc then
 3470:C        7              begin
 3471:C        7                firstmodname := fdirectory^[vmodnum].dtid;
 3472:C        7                firstmodnum  := vmodnum;
 3473:C        7              end;
 3474:C        6            end                              { 3.0 BUG # 57   4/10/84 }
 3475:C        6            else dobeep                      { 3.0 BUG # 57   4/10/84 }
 3476:C        6          end
 3477:C        5          else dobeep;
 3478:C        4     'S':;
 3479:C        4     'T': if vmodnum>0 then
```

```
3480:C        5          begin xfer; checkassoc;
3481:C        5          end
3482:C        5          else dobeep;
3483:C        4      'U':begin
3484:C        4           fgotoxy(output,17,7); write(cteol);
3485:C        4           setstrlen(tempf,0); findmod(tempf,im);
3486:C        4           case im of
3487:C        5           -1: mnotfound;
3488:C        5            0: { no module name given, so default }
3489:C        5               begin untilmodname := '(end of file)';
3490:C        5                     untilmodnum  := fdirectory^[0].dnumfiles+1;
3491:C        5               end;
3492:C        5           otherwise     { found the module }
3493:C        5               if im>=firstmodnum then
3494:C        6               begin untilmodname := tempf; untilmodnum:=im;
3495:C        6               end
3496:C        6               else outoforder;
3497:C        5               assoc := assoc and (im>firstmodnum);
3498:C        5           end;  { case im }
3499:C        4         end;
3500:C        4      ' ': if vmodnum>0 then
3501:C        5           begin verifynext; checkassoc;
3502:C        5           end
3503:C        5           else dobeep;
3504:C        4      otherwise dobeep
3505:C        4      end;
3506:C        3    until commandchar='S';
3507:C        2  end;    (doedit)
3508:S
3509:D        1  procedure finishboot;
3510:C        2  begin
3511:C        2  close(outfile, 'LOCK');
3512:C        2  if ioresult<>0 then escape(126);
3513:C        2  outopen := false; booting := false;
3514:C        2  outmodnum := 0;
3515:C        2  end;
3516:S
3517:D        1  procedure menu;
3518:D    -6  2  var lc: string[4];
3519:C        2  begin
3520:C        2   fgotoxy(output, 0,2); write('Q  Quit'); clear(1);
3521:C        2
3522:C        2   write('P  Printout  ');
3523:C        2   if printopen then
3524:C        3     begin
3525:C        3     if printeron then write('ON   ')
3526:C        4                  else write('OFF  ');
3527:C        3     write(listfilename);
3528:C        3     clear(1);
3529:C        3     end
3530:C        3   else none;
3531:C        2
3532:C        2   if outmodnum > 0 then write('K  Keep o')
3533:C        3   else if (newmods = NIL) and not booting
3534:C        4                  then write('0       0')
3535:C        4                  else write('        o');
3536:C        2   write('utput file:  ');
3537:C        2   if outopen then writeln(outfilename,cteol)
3538:C        3              else none;
3539:C        2   if outopen then
```



```
3540:C        3    begin
3541:C        3    if booting then write('B  finish Boot')
3542:C        4    else if newmods <> NIL then write('L  finish Linking')
3543:C        5    else if linking then write('C  Copy')
3544:C        6    else write('L  Link');
3545:C        3    write(cteol); fgotoxy(output, 40,5);
3546:C        3    if booting then      begin
3547:C        4                         writeln('BOOTING');
3548:C        4                         lc := 'boot';
3549:C        4                         end
3550:C        4    else if linking then begin
3551:C        5                         writeln('LINKING');
3552:C        5                         lc := 'link';
3553:C        5                         end
3554:C        5    else                 begin
3555:C        5                         writeln('COPYING');
3556:C        5                         lc := 'copy';
3557:C        5                         end;
3558:C        3    end
3559:C        3   else
3560:C        3    begin
3561:C        3     writeln('B  write to Boot disk',cteol);
3562:C        3     writeln('H  file Header maximum size:  ',maxmodules,cteol);
3563:C        3    end;
3564:S
3565:C        2   if linking and outopen then
3566:C        3    begin
3567:C        3    write('N  Name of new module:  ');
3568:C        3    if newmodname.syp = NIL then none
3569:C        4    else writeln(newmodname.syp^,cteol);
3570:C        3    writeln('R  Relocation base:  ',startreloc:12,cteol);
3571:C        3    writeln('G  Global base:      ',startglobal:12,cteol);
3572:C        3    write  ('S  Space for patches:');
3573:C        3    if patchbytes > 0 then writeln(patchbytes:12)
3574:C        4    else clear(1);
3575:C        3    patchbytes := 0;
3576:C        3    write('D  output Def table?  ');
3577:C        3    if defsout then write('YES')
3578:C        4    else write('NO ');
3579:C        3    writeln(cteol);
3580:C        3    writeln('X  copyright notice:  ',copyright);
3581:C        3    end
3582:C        3   else clear(7);
3583:S
3584:C        2   fgotoxy(output, 0,13);
3585:C        2   write('I       Input  file:  ');
3586:C        2   if fdirectory = NIL then
3587:C        3    begin none; clear(7); end
3588:C        3   else
3589:C        3    begin
3590:C        3    writeln(infilename,cteol);
3591:C        3    if outopen then writeln('E  Edit') else clear(1);
3592:C        3    writeln('F  list File directory');
3593:C        3    if outopen then writeln('A  ',lc,' All modules') else clear(1);
3594:C        3    write('V  Verify modules');
3595:C        3    if verifying then
3596:C        4      begin
3597:C        4      fgotoxy(output, 40,17); writeln('VERIFYING');
3598:C        4      end
3599:C        4    else clear(1);
```

```
3600:C        3   write('M  input Module:  ');
3601:C        3   if vmodnum = 0 then
3602:C        4     begin none; clear(3); end
3603:C        4   else
3604:C        4     begin
3605:C        4     writeln(fdirectory^[vmodnum].dtid,cteol);
3606:C        4     writeln('U  Unassemble object');
3607:C        4     if outopen then writeln('T  Transfer (',lc,') module')
3608:C        5     else clear(1);
3609:C        4     if verifying then writeln('<space> to continue verifying')
3610:C        5     else clear(1);
3611:C        4     end;
3612:C        3   end;
3613:C        2 end; {menu}
3614:S
3615:C        1 procedure getcommand;
3616:C   -82  2 var err: string[80];
3617:C        2 begin
3618:C        2   repeat
3619:C        3   try
3620:C        4   menu;
3621:C        4   getcommandchar('command?',commandchar);
3622:C        4   case commandchar of
3623:C        5     'A': if (fdirectory<>NIL) and  outopen then copyfile else dobeep;
3624:C        5     'B': if booting then finishboot
3625:C        6          else if outopen then dobeep
3626:C        7          else openout(true);
3627:C        5     'C': if outopen and linking and (newmods = NIL)
3628:C        6          and not booting then copyon else dobeep;
3629:C        5     'D': if linking and outopen
3630:C        6          then defsout := not defsout else dobeep;
3631:C        5     'E': if (fdirectory<>NIL) and outopen then doedit else dobeep;
3632:C        5     'F': if fdirectory<>NIL then printdirectory else dobeep;
3633:C        5     'G': if linking and outopen then setglobal else dobeep;
3634:C        5     'H': if outopen then dobeep else setmaxmodules;
3635:C        5     'I': begin setstrlen(infilename,0); openin; end;
3636:C        5     'K': if (outmodnum > 0) and not booting
3637:C        6          then closeout else dobeep;
3638:C        5     'L': if booting then dobeep
3639:C        6          else if newmods <> NIL then link
3640:C        7          else if outopen and not linking
3641:C        8          then initlink else dobeep;
3642:C        5     'M': if fdirectory<>NIL then openmod else dobeep;
3643:C        5     'N': if linking and outopen then setname else dobeep;
3644:C        5     'O': if (outmodnum = 0) and (newmods=NIL) and not booting
3645:C        6          then openout(false) else dobeep;
3646:C        5     'P': toggleprinter;
3647:C        5     'Q': quit;
3648:C        5     'R': if linking and outopen then setreloc else dobeep;
3649:C        5     'S': if linking and outopen then makepatchspace else dobeep;
3650:C        5     'T': if (vmodnum > 0) and outopen then xfer else dobeep;
3651:C        5     'U': if (vmodnum > 0) then unassemble else dobeep;
3652:C        5     'V': if fdirectory<>NIL then verifymod else dobeep;
3653:C        5     'X': if linking and outopen then setcopyright else dobeep;
3654:C        5     ' ': if verifying then verifynext;
3655:S
3656:C        5   otherwise dobeep;
3657:C        5   end;
3658:S
3659:C        4    ecover
```



```
3660:C        4    begin
3661:C        4    if (escapecode <> -20) and (escapecode <> 123) and (escapecode<>128)
3662:C        5                                                        then errorline;
3663:C        4    if escapecode=-10 then begin getioerrmsg(err, ires); writeln(err); end
3664:C        5    else case escapecode of
3665:C        6     110: write('symbols defined recursively');
3666:C        6     111: write('improper link info format');
3667:C        6     112: write('not enough memory');
3668:C        6     113: write('output file full');
3669:C        6     114: write('error writing to boot disk, ioresult = ',ires:1);
3670:C        6     116: write('''', infilename, ''' is not a code file');
3671:C        6     118: write('printer or list file not on line');
3672:C        6     119: write('duplicate symbol definition');
3673:C        6     120: write('module being booted has external references');
3674:C        6     121: write('unexpected end of code');
3675:C        6     122: write(errors:1, ' errors during linking',cteol);
3676:C        6     123,128,129: {error message already printed};
3677:C        6     124: write('integer required');
3678:C        6     125: write('integer too large');
3679:C        6     126: write('unable to close output, ioresult = ',ires:1);
3680:C        6     127: write('file header full');
3681:C        6     otherwise escape(escapecode);
3682:C        6     end; {case escapecode}
3683:C        4     if streaming then escape(-1);
3684:C        4     if (escapecode-100) in [12,16] then closein;
3685:C        4     if (escapecode-100) in [10..13,19,22,26,28] then
3686:C        5       begin
3687:C        5       if newmods <> NIL then begin closein; closefiles; end;
3688:C        5       linking := false;     newmods := NIL;
3689:C        5       if outopen then close(outfile);
3690:C        5       outopen := false;     outmodnum := 0;
3691:C        5       booting := false;
3692:C        5       lowheap := lowheap0;
3693:C        5       end;
3694:C        4    end; {recover}
3695:C        3   until commandchar = 'Q';
3696:C        2 end {getcommand};
3697:S
3698:S
3699:D        1 procedure wrapup;
3700:C        2 begin
3701:C        2  pageeject;
3702:C        2  closein;
3703:C        2  closefiles;
3704:C        2  if (pagenum=0) and (linenum=0)
3705:C        3    then close(listing)
3706:C        3    else close(listing, 'lock');
3707:C        2 end; {wrapup}
3708:S
3709:C        1 begin {program linker}
3710:C        1   with linkerdate do
3711:C        2     begin day := 4; year := 84; month := 6; end;
3712:C        1   sysdate(todaysdate);
3713:S
3714:C        1   pagenum := 0;          linenum := 0;
3715:S
3716:C        1   fgotoxy(output, 0,0);
3717:C        1   printheader(output);
3718:C        1   fgotoxy(output, 0, 22);
3719:C        1   writeln('Copyright 1984 Hewlett-Packard Company.');
```

```
  3720:S
  3721:C        1   mark(lowheap.p); lowheap0 := lowheap;
  3722:C        1   highheap.a := lowheap.a + memavail - 5000;
  3723:C        1   release(highheap.p); highheap0 := highheap;
  3724:S
  3725:C        1   listfilename := 'PRINTER:LINK.ASC';
  3726:C        1   rewrite(listing, listfilename);
  3727:C        1   printopen := ioresult = 0;
  3728:C        1   printeron := false;
  3729:S
  3730:C        1   patchbytes := 0;
  3731:C        1   maxmodules := 38;     outdirectsize := 2;
  3732:C        1   linking := false;     newmods := NIL;
  3733:C        1   outopen := false;     outmodnum := 0;
  3734:C        1   booting := false;     verifying := false;
  3735:C        1   fdirectory := NIL;    vmodnum := 0;
  3736:C        1   loadfib.php := NIL;
  3737:S
  3738:C        1   try getcommand;
  3739:S
  3740:C        2   recover begin
  3741:C        2           esccode := escapecode;
  3742:C        2           wrapup;
  3743:C        2           escape(esccode);
  3744:C        2           end;
  3745:S
  3746:C        1   wrapup;
  3747:S
  3748:C        1 end.

No errors. 1 warnings.
            ***** Nonstandard language features enabled *****
```

Description

LIFDAM provides access to LIF format directories. The module also contains the code which installs itself in the system.

Usage

LIFDAM is called by the filesystem to operate on LIF format directories.

```
    1:S                                                 (*
    2:S
    3:S           (c) Copyright Hewlett-Packard Company, 1982.
    4:S          All rights are reserved.  Copying or other
    5:S          reproduction of this program except for archival
    6:S          purposes is prohibited without the prior
    7:S          written consent of Hewlett-Packard Company.
    8:S
    9:S
   10:S                       RESTRICTED RIGHTS LEGEND
   11:S
   12:S          Use, duplication, or disclosure by the Government
   13:S          is subject to restrictions as set forth in
   14:S          paragraph (b) (3) (B) of the Rights in Technical
   15:S          Data and Computer Software clause in
   16:S          DAR 7-104.9(a).
   17:S
   18:S          HEWLETT-PACKARD COMPANY
   19:D        0 Fort Collins, Colorado                        *)
   20:S
   21:S
   22:D        0 $MODCAL$
   23:D        0 $DEBUG OFF,  range off, ovflcheck off$
   24:D        0 program instlifdam;
   25:D        1 module lifmodule;
   26:D        1 import sysglobals, sysdevs, misc, fs;
   27:D        1 export
   28:D        1   procedure lifdam(anyvar f: fib; unum: unitnum; request: damrequesttype);
   29:D        1   procedure installlifdam;
   30:S
   31:D        1 implement
   32:D        1   const
   33:D        1     entrysize = 32;
   34:D        1   type
   35:D        1     vname   =packed array[1..6] of char;
   36:D        1     lifname = packed array[1..10] of char;
   37:D        1     bcd     = 0..15;
   38:D        1     word    = 0..65535;
   39:D        1     integer16 = -32768..32767;
   40:D        1     word15  = 0..32767;
   41:D        1     tdate   = packed array[1..12] of bcd;
   42:D        1     lvheader=packed record      {volume header sector 0}
   43:D        1                 discid   : word;
   44:D        1                 volname  : vname;
   45:D        1                 dstart   : integer;
   46:D        1                 dummy1   : integer16;
   47:D        1                 dummy2   : integer16;
   48:D        1                 dsize    : integer;
   49:D        1                 version  : integer16;
   50:D        1                 dummy3   : integer16;
   51:D        1                 tps      : integer;      {tracks/surface}
   52:D        1                 spm      : integer;      {surfaces/medium}
   53:D        1                 spt      : integer;      {sectors/track}
   54:D        1                 cdate    : tdate;        {volume create time}
   55:D        1                 filler   : packed array[21..123] of integer16;
   56:D        1                 sdate    : tdate;
   57:D        1                 dummy4   : integer16;
   58:D        1               end;
   59:D        1   direntry = packed record
   60:D        1                 fname    : lifname;
```



```
   61:D        1                 ftype    : integer16;
   62:D        1                 fstart   : integer;
   63:D        1                 fsize    : integer;
   64:D        1                 fdate    : tdate;
   65:D        1                 lastvol  : boolean;
   66:D        1                 volnumber: word15;
   67:D        1                 extension: integer;
   68:D        1               end;
   69:S
   70:D        1   spacerec  = record
   71:D        1                 sstart      : integer;
   72:D        1                 ssize       : integer;
   73:D        1                 here      : integer;
   74:D        1                 hole      : integer;
   75:D        1               end;
   76:D        1   catarray  = array[1..maxint] of catentry;
   77:D        1   dirfile   = file of direntry;
   78:D        1   var
   79:D    -4  1     dir        : ^dirfile;
   80:S
   81:D    -4  1 {*************************************************************************}
   82:D        1   procedure goodio;
   83:C        2   begin if ioresult<>ord(inoerror) then escape(0); end;
   84:S
   85:D    -4  1 {*************************************************************************}
   86:D        1   procedure badio(result : iorsltwd);
   87:C        2   begin ioresult := ord(result); escape(0); end;
   88:S
   89:D    -4  1 {*************************************************************************}
   90:D        1   procedure pactostr(anyvar pc: lifname; l:integer; var s:string);
   91:D    -4  2   var i : integer;
   92:C        2   begin
   93:C        2     setstrlen(s,l); for i:=1 to l do s[i] := pc[i];      i := l;
   94:C        2     while (i>1) and (s[i]=' ') do i:=i-1; setstrlen(s,i);
   95:C        2   end;
   96:S
   97:D    -4  1 {*************************************************************************}
   98:D        1   procedure strtopac(anyvar s:string255; l:integer;
   99:D        2                      anyvar pc:lifname;  sizechk:boolean);
  100:D    -8  2   var i,k : integer;
  101:C        2   begin
  102:C        2     if sizechk then
  103:C        3       if (strlen(s)>l) or (strlen(s)=0) then badio(ibadtitle);
  104:C        2     k:=strlen(s);
  105:C        2     for i:=1 to l do if i>k then pc[i] := ' ' else pc[i] := s[i];
  106:C        2   end;
  107:S
  108:D    -4  1 {*************************************************************************}
  109:D        1   procedure strtoany(var s:string; anyvar s2:string255);
  110:C        2   begin s2:=s; end;
  111:S
  112:D    -4  1 {*************************************************************************}
  113:D        1   procedure setdate(var d:tdate);
  114:D        2   var
  115:D    -6  2     doy:daterec;        tod:timerec;
  116:C        2   begin
  117:C        2     sysdate(doy);       systime(tod);
  118:C        2     with doy, tod do
  119:C        3     begin
  120:C        3       d[1]  := year div 10;   d[2]  := year mod 10;
```

```
   121:C        3       d[3]  := month div 10;  d[4]  := month mod 10;
   122:C        3       d[5]  := day div 10;    d[6]  := day mod 10;
   123:C        3       d[7]  := hour div 10;   d[8]  := hour mod 10;
   124:C        3       d[9]  := minute div 10; d[10] := minute mod 10;
   125:C        3       d[11] := (centisecond div 100) div 10;
   126:C        3       d[12] := (centisecond div 100) mod 10;
   127:C        3     end;
   128:C        2   end;
   129:S
   130:D    -4  1 (*****************************************************************************)
   131:D        1   procedure lifnametostr(anyvar ln :lifname ; var s:string);
   132:D        2   label 1;
   133:D        2   var
   134:D    -4  2     sl  : integer;
   135:D    -6  2     fk  : filekind;
   136:D    -7  2     found : boolean;
   137:S
   138:C        2   begin {lifname to str}
   139:C        2     pactostr(ln,10,s); sl := strlen(s);
   140:C        2     if sl=10 then
   141:C        3     begin
   142:C        3       if suffix(s)=datafile then
   143:C        4       begin     { rip underscores and try to add suffix }
   144:C        4         while (sl>=1) and (s[sl]='_') do sl := sl - 1;
   145:C        4         for fk:=untypedfile to lastfkind do
   146:C        5         begin
   147:C        5           if strlen(suffixtable^[fk])>0 then
   148:C        6             if suffixtable^[fk][1]=s[sl] then
   149:C        7             begin       { found suffix }
   150:C        7               { change last char to . then append suffix }
   151:C        7               setstrlen(s,sl); s[sl] := '.';
   152:C        7               s := s + suffixtable^[fk];  goto 1;
   153:C        7             end;
   154:C        5         end;
   155:C        4       end;      { for }
   156:C        3     end;
   157:C        2   1:end;  {lifname to str}
   158:S
   159:D    -4  1 (*****************************************************************************)
   160:D        1   procedure strtolifname(var s:string; var ln:lifname);
   161:D        2   var
   162:D    -8  2     sl, i       : integer;
   163:D  -252  2     stemp,temp2 : fid;  {31jan83  temp2 for case insensitive suffix)
   164:D  -254  2     fk          : filekind;
   165:S
   166:C        2   begin {str to lifname}
   167:C        2     sl := strlen(s);
   168:C        2     fk := suffix(s);
   169:C        2     if fk=datafile then
   170:C        3     begin       { data files have no suffix }
   171:C        3       if sl>10 then badio(ibadtitle);
   172:C        3       strtopac(s,10,ln,false);
   173:C        3     end
   174:C        3     else
   175:C        3     begin       { remove the suffix }
   176:C        3       sl := strlen(s)-strlen(suffixtable^[fk]);
   177:C        3       if sl>10 then badio(ibadtitle);
   178:C        3       strtopac(s,10,ln,false);  { pack the name }
   179:C        3       { replace dot with first char of suffix (to preserve uniqueness)}
   180:C        3       ln[sl] := suffixtable^[fk][1];    sl := sl+1;
```



```
   181:C        3       for i:=sl to 10 do ln[i] := '_';  { pad with _ }
   182:C        3     end;
   183:C        2     lifnametostr(ln,stemp);     { decompress the name as a final check }
   184:C        2     if stemp<>s then
   185:C        3     begin                       {31jan83 case insensitive suffix testing)
   186:C        3       temp2 := s;               { copy s, then remove given suffix }
   187:C        3       setstrlen(temp2,strlen(temp2)-strlen(suffixtable^[fk]));
   188:C        3       temp2:= temp2 + suffixtable^[fk];         { add suffix from table }
   189:C        3       if stemp<>temp2 then badio(ibadtitle);    { check again }
   190:C        3     end;
   191:C        2   end; {str to lifname}
   192:S
   193:D    -4  1 (*****************************************************************************)
   194:D    -4  1 (*****************************************************************************)
   195:D        1   procedure lifdam(anyvar f: fib; unum: unitnum; request: damrequesttype);
   196:D        2     var
   197:D  -256  2       vol      : lvheader;
   198:D  -274  2       volid    : vid;
   199:D  -277  2       ok, mediavalid ,anychange: boolean;
   200:D  -294  2       dindex, dlast, dend, vsize       : integer;
   201:D  -326  2       dentry   : direntry;
   202:S
   203:D  -326  2   $iocheck off$
   204:D  -326  2 (*****************************************************************************)
   205:D        2   function volsize:integer;
   206:C        3   begin
   207:C        3     if vsize=0 then vsize := ueovbytes(unum);
   208:C        3     volsize := vsize;
   209:C        3   end;
   210:S
   211:D  -326  2 (*****************************************************************************)
   212:D        2   procedure cleanup;
   213:C        3   begin
   214:C        3     if ioresult=ord(zmediumchanged) then mediavalid := false;
   215:C        3     unitable^[unum].umediavalid   := mediavalid;
   216:C        3     unitable^[unum].ureportchange := true;
   217:C        3   end;
   218:S
   219:D  -326  2 (*****************************************************************************)
   220:D        2   procedure checkftitle;
   221:C        3   begin
   222:C        3     if (strlen(f.ftitle)>tidleng) or (strlen(f.ftitle)=0) then badio(ibadtitle);
   223:C        3     f.ftid := f.ftitle;
   224:C        3   end;
   225:S
   226:D  -326  2 (*****************************************************************************)
   227:D        2   function vvname:boolean;
   228:D        3   var
   229:D    -4  3     i : integer;
   230:D    -5  3     b : boolean;
   231:C        3   begin
   232:C        3     vvname := true; b := true;
   233:C        3     for i := 1 to 6 do  {1feb83 allow all blank names}
   234:C        4     begin
   235:C        4       if b then b := vol.volname[i]>' ';
   236:C        4       if not b then     {1feb83 allow all blank names}
   237:C        5         if vol.volname[i]<>' ' then vvname := false;
   238:C        4     end
   239:C        4   end;
   240:S
```

```
241:D  -326  2 {*************************************************************************}
242:D        2   function lifvol:boolean;
243:D        3   var
244:D    -4  3     i : integer;          {31jan83 allow all blank volname}
245:C        3   begin { read and validate the volume header }
246:C        3     with fibp(dir)^, unitable^[unum] do
247:C        4     begin
248:C        4       fileid := 0; fpeof := maxint;   { initialize dir }
249:C        4       if uisblkd then
250:C        5       begin
251:C        5         call(tm,fibp(dir),readbytes,vol,sizeof(lvheader),0);
252:C        5         with vol do
253:C        6           ok := ((ioresult=ord(inoerror)) and (discid=32768) and
254:C        6                  (dummy1=4096) and (dummy2=0) and (dummy3=0) and
255:C        6                  (dstart>1) and (dsize>0) and vvname);
256:C        5       end
257:C        5       else ok := false;
258:C        4       ureportchange := true;    { now let TM report any mediachanges }
259:C        4       umediavalid   := true;
260:C        4       lifvol := ok;
261:C        4       if ok then
262:C        5       begin
263:C        5         if vol.volname[1]=' ' then      {31jan83 allow all blank volname}
264:C        6         begin
265:C        6           setstrlen(volid,6); for i:=1 to 6 do volid[i]:=' ';
266:C        6         end
267:C        6         else
268:C        6           pactostr(vol.volname,6,volid);
269:C        5         if volid<>uvid then
270:C        6         begin mediavalid := false; uvid := volid; end;
271:C        5       end
272:C        5       else setstrlen(uvid,0);
273:C        4       if (not ok) and (ioresult=ord(inoerror)) then ioresult:=ord(inodirectory);
274:C        4     end;
275:C        3   end;  { lifvol }
276:S
277:D  -326  2   $iocheck on$
278:D  -326  2 {*************************************************************************}
279:D        2   procedure opendir;
280:C        3   begin
281:C        3     if lifvol then
282:C        4     begin
283:C        4       dlast:=vol.dsize * 8;             { # entries in directory }
284:C        4       dend:=vol.dstart + vol.dsize;   { directory end sector + 1 }
285:C        4       with fibp(dir)^ do
286:C        5       begin   { initialize the fib ( fake OPEN )}
287:C        5         fisnew:=false;
288:C        5         freadable:=true;       fwriteable:=true;
289:C        5         freadmode:=false;      fbufvalid:=false;
290:C        5         feof:=false;           fmodified:=false;
291:C        5         fileid:=vol.dstart*256;
292:C        5         if vol.version>0 then vsize := vol.tps*vol.spm*vol.spt*256
293:C        6                          else vsize := 0;
294:C        5         fpeof:=vol.dsize*256;       { end of directory }
295:C        5         fleof:=fpeof;
296:C        5         fpos:=0;               am:=amtable^[datafile];
297:C        5         frepicnt:=0;           flastpos:=-1;
298:C        5         fbufchanged:=false;    fbuffered:=true;
299:C        5       end;
300:C        4       dindex:=1;   read(dir^,dentry);
```



```
301:C        4       goodio;
302:C        4     end
303:C        4     else escape(0);
304:C        3   end;  { opendir }
305:S
306:D  -326  2 {*************************************************************************}
307:D        2   procedure checkvolid;
308:C        3   begin
309:C        3     if f.fvid<>volid then badio(ilostunit);
310:C        3   end;
311:S
312:D  -326  2 {*************************************************************************}
313:D        2   procedure flushdir;
314:C        3   begin { june 83 fixed to account for ops which don't open the directory RDQ}
315:C        3   if fibp(dir)^.freadable
316:C        4         { directory open so flush thru AM }
317:C        4     then call(fibp(dir)^.am,fibp(dir),flush,ioresult,0,0)
318:C        4         { directory not open so flush thru TM }
319:C        4     else call(unitable^[unum].tm,fibp(dir),flush,ioresult,0,0);
320:C        3     anychange := false;
321:C        3   end;
322:S
323:D  -326  2 {*************************************************************************}
324:D        2   procedure getsdate(anyvar svdate:daterec);
325:C        3   begin
326:C        3     if lifvol then
327:C        4     with vol, svdate do
328:C        5     begin
329:C        5       if not ((sdate[5]=0) and (sdate[6]=0)) then
330:C        6       begin
331:C        6         year:=sdate[1]*10+sdate[2];
332:C        6         month:=sdate[3]*10+sdate[4];
333:C        6         day:=sdate[5]*10+sdate[6];
334:C        6       end;
335:C        5     end;
336:C        3   end;
337:S
338:D  -326  2 {*************************************************************************}
339:D        2   procedure setsdate(anyvar svdate:daterec);
340:D        3   var
341:D    -4  3     j   : integer;
342:C        3   begin
343:C        3     if lifvol then
344:C        4     with svdate, vol do
345:C        5     begin
346:C        5       sdate[1]:=year div 10; sdate[2]:=year mod 10;
347:C        5       sdate[3]:=month div 10; sdate[4]:=month mod 10;
348:C        5       sdate[5]:=day div 10; sdate[6]:=day mod 10;
349:C        5       for i:=7 to 12 do sdate[i]:=0;    {clear time of day}
350:C        5       with unitable^[unum] do
351:C        6       begin
352:C        6         call(tm,fibp(dir),writebytes,vol,sizeof(lvheader),0);
353:C        6         goodio;          anychange := true;
354:C        6       end;
355:C        5     end;
356:C        3   end;  { setsdate }
357:S
358:D  -326  2 {*************************************************************************}
359:D        2   procedure cleandir;
360:D        3   var
```

```
   361:D    -4  3     k           : integer;
   362:D   -36  3     tempd       : direntry;
   363:S
   364:C        3   begin {cleandir}
   365:C        3     k:=1;       seek(dir^,k);
   366:C        3     repeat
   367:C        4       read(dir^,tempd);
   368:C        4       if tempd.ftype=-1 then k:=dlast
   369:C        5       else
   370:C        5       if tempd.ftype<>0 then
   371:C        6       begin
   372:C        6         if (tempd.fdate[3]=9) and (tempd.fdate[4]=9) then
   373:C        7         begin { temp file so purge it }
   374:C        7           tempd.ftype:=0;
   375:C        7           writedir(dir^,k,tempd);       anychange := true;
   376:C        7         end;
   377:C        6       end;
   378:C        4       k:=k+1;
   379:C        4     until k>dlast;
   380:C        3     mediavalid := true;
   381:C        3   end;  {clean dir}
   382:S
   383:D  -326  2 {*****************************************************************************}
   384:D        2   procedure crunchv; { assumed to be called from procedure lifdam only 24jan83}
   385:D        3   var
   386:D   -12  3     frompos, topos, todindex    : integer;
   387:D   -24  3     bsize, filesize, movesize   : integer;
   388:D   -32  3     bufptr, heapmark            : windowp;
   389:D   -33  3     changed                     : boolean;
   390:D   -38  3     datafib                     : fibp;
   391:S
   392:C        3   begin
   393:C        3     opendir; checkvolid;
   394:C        3     if strlen(f.ftitle)<>0 then badio(ibadtitle);
   395:C        3     cleandir;
   396:C        3     { allocate the buffer space }
   397:C        3     MARK(heapmark);
   398:C        3     try
   399:C        4       bsize:=(memavail-(1024*5));
   400:C        4       if bsize>=256 then
   401:C        5       begin       { buffer in sector multiples }
   402:C        5         bsize:=(bsize div 256) * 256;
   403:C        5         newwords(bufptr,bsize div 2);
   404:C        5         new(datafib);   { set up the data fib }
   405:C        5         with datafib^ do
   406:C        6         begin
   407:C        6           funit:=fibp(dir)^.funit;
   408:C        6           fileid:=0;
   409:C        6           fpeof:=volsize;      fleof:=fpeof;
   410:C        6         end;
   411:C        5       end
   412:C        5       else escape(-2); { not enough room to run }
   413:C        4       { krunch it }
   414:C        4       dindex:=1;
   415:C        4       todindex:=0;       topos:=dend*256;        anychange:=false;
   416:C        4       repeat
   417:C        5         changed:=false;
   418:C        5         readdir(dir^,dindex,dentry);
   419:C        5         if dentry.ftype=-1 then dindex:=dlast
   420:C        6         else
```



```
   421:C        6         if dentry.ftype=efttable^[badfile] then
   422:C        7         begin   { bad sectors ; don't move this file }
   423:C        7           todindex:=todindex+1;
   424:C        7           if dindex<>todindex then changed:=true;       { move the entry }
   425:C        7           topos:=(dentry.fstart+dentry.fsize)*256;      { move topos }
   426:C        7         end
   427:C        7         else
   428:C        7         if dentry.ftype=0 then anychange := true        { found purged entry }
   429:C        8         else
   430:C        8         begin { move the file ? }
   431:C        8           todindex:=todindex+1;
   432:C        8           if dindex<>todindex then changed:=true;       { move the entry}
   433:S
   434:C        8           frompos:=dentry.fstart*256;
   435:C        8           if frompos<>topos then
   436:C        9             begin { move the data }
   437:C        9               filesize:=dentry.fsize*256;   { bytes to move}
   438:C        9               dentry.fstart:=topos div 256; { set start}
   439:C        9               changed:=true;
   440:C        9               with unitable^[datafib^.funit] do
   441:C       10               repeat
   442:C       11                 if filesize>bsize then movesize:=bsize
   443:C       12                                   else movesize:=filesize;
   444:C       11                 call(tm,datafib,readbytes,bufptr^,movesize,frompos);
   445:C       11                 frompos:=frompos+movesize; goodio;
   446:S
   447:C       11                 call(tm,datafib,writebytes,bufptr^,movesize,topos);
   448:C       11                 topos:=topos+movesize; goodio;
   449:S
   450:C       11                 filesize:=filesize-movesize;
   451:C       11               until filesize=0;
   452:C        9             end  { move the data }
   453:C        9           else topos:=topos+dentry.fsize*256;
   454:C        8         end;   { move the file ? }
   455:C        5         if changed then
   456:C        6         begin
   457:C        6           writedir(dir^,todindex,dentry);       anychange := true;
   458:C        6         end;
   459:C        5         dindex:=dindex+1;
   460:C        5       until dindex>dlast;
   461:C        4       if anychange then
   462:C        5       begin    { put end of directory mark }
   463:C        5         if todindex<dlast then
   464:C        6         begin
   465:C        6           dentry.ftype:=-1;
   466:C        6           writedir(dir^,todindex+1,dentry);
   467:C        6         end;
   468:C        5       end;
   469:C        4       call(unitable^[datafib^.funit].tm,datafib,flush,ioresult,0,0);
   470:C        4       RELEASE(heapmark);
   471:C        4     recover
   472:C        4       begin
   473:C        4         RELEASE(heapmark);
   474:C        4         frompos := ioresult;  topos := escapecode;      { save state 24jan83}
   475:C        4         if anychange then flushdir;                  { try to clean up 24jan83}
   476:C        4         ioresult := frompos;                      { restore the state 24jan83}
   477:C        4         escape(frompos);                                    {exit 24jan83}
   478:C        4       end;
   479:C        3   end;  {crunchv}
   480:S
```

```
481:D  -326  2 (*****************************************************************************)
482:D        2   procedure domakedirectory(anyvar cat:catentry);
483:D        3   var
484:D    -4  3     i      : integer;
485:D    -8  3     actualsize : integer;
486:D  -264  3     secbuf : packed array[0..63] of integer;
487:C        3   begin
488:C        3     if strlen(f.ftitle)>0 then badio(ibadrequest);
489:C        3     with vol, cat do
490:C        4     begin
491:C        4       if not lifvol then
492:C        5       begin
493:C        5         if (ioresult<>ord(inoerror)) and
494:C        6            (ioresult<>ord(inodirectory)) then escape(0);
495:C        5         ioresult := ord(inoerror);
496:C        5         { clear header fields }
497:C        5         discid := 32768;
498:C        5         dummy1 := 4096;   dummy2 := 0;   dummy3 := 0;
499:C        5         version := 0;
500:C        5       end
501:C        5       else checkvolid;
502:C        4                 { directory size checks }
503:C        4       dstart := 2;
504:C        4       dsize  := (((cextra1*entrysize)+255) div 256);
505:C        4       if dsize<=0 then dsize := 10;     { default directory size }
506:C        4                 { size checks }
507:C        4       actualsize := ueovbytes(unum);
508:C        4       if (cpsize>actualsize) or (cpsize<1024) then badio(inoroom);
509:C        4       if (dstart+dsize+1)*256>=cpsize then badio(inoroom);
510:C        4                 { fill in the pieces }
511:C        4       strtopac(cname,6,volname,true);   { volume name }
512:C        4       if version>0 then
513:C        5         if (tps*spm*spt*256)<>cpsize then version := 0;
514:S
515:C        4       setdate(cdate);   { fill in create date }
516:C        4       for i := 1 to 12 do sdate[i] := 0; { clear system date }
517:S
518:C        4       if version=0 then
519:C        5       begin     { create pseudo level 1 header }
520:C        5         version := 1;
521:C        5         tps := 1; spm := 1; spt := cpsize div 256;
522:C        5         for i:=21 to 123 do filler[i]:=0;
523:C        5       end;
524:C        4       dummy4 := 0;      { clear 250 maint. word }
525:C        4                 { write volume header }
526:C        4       with unitable^[unum] do
527:C        5       BEGIN
528:C        5         call(tm,fibp(dir),writebytes,vol,sizeof(lvheader),0); goodio;
529:C        5         for i:=0 to 63 do secbuf[i] := 0;
530:C        5         call(tm,fibp(dir),writebytes,secbuf,256,256);        {clear sector 1}
531:C        5       END;
532:C        4                 { write end of directory }
533:C        4       opendir;
534:C        4       dentry.ftype := -1;
535:C        4       writedir(dir^,1,dentry); flushdir;
536:C        4     end;
537:C        3   end;  {domakedirectory}
538:S
539:D  -326  2 (*****************************************************************************)
540:D        2   procedure liftofkind(lt:integer16; var fk:filekind);
```



```
541:C        3   begin
542:C        3     fk:=untypedfile;
543:C        3     while (fk<>lastfkind) and (efttable^[fk]<>lt) do fk:=succ(fk);
544:C        3     if efttable^[fk]<>lt then fk:=DATAFILE;
545:C        3   end;
546:S
547:D  -326  2 (*****************************************************************************)
548:D        2   procedure cvtdatetime(var fdate:tdate; var date:daterec; var time:timerec);
549:C        3   begin
550:C        3     date.year:=fdate[1]*10+fdate[2];
551:C        3     date.month:=fdate[3]*10+fdate[4];
552:C        3     date.day:=fdate[5]*10+fdate[6];
553:C        3     if (date.month=0) or (date.day=0) then
554:C        4       begin date.year:=0; date.month:=3; date.day:=1; end;
555:C        3     time.hour:=fdate[7]*10+fdate[8];
556:C        3     time.minute:=fdate[9]*10+fdate[10];
557:C        3     time.centisecond:=(fdate[11]*10+fdate[12])*100;
558:C        3   end;
559:S
560:D  -326  2 (*****************************************************************************)
561:D        2   procedure doopendirectory(anyvar cat:catentry);
562:C        3   begin
563:C        3     opendir;    checkvolid;
564:C        3     with cat do
565:C        4     begin      { volume info }
566:C        4       cname:=volid;
567:C        4       cstart:=(vol.dstart+vol.dsize)*256;      { start of data area }
568:C        4       cblocksize:=256;              { No. of bytes in allocation unit }
569:C        4       cpsize:=volsize;                 { physical size of the volume }
570:C        4       clsize:=cpsize-cstart;             { data space on the medium }
571:C        4       cextra1:=vol.dsize*8;               { number of possible files }
572:C        4       cextra2:=-1;                          { unused space available }
573:C        4       cvtdatetime(vol.sdate,clastdate,clasttime);     { system date }
574:C        4       cvtdatetime(vol.cdate,ccreatedate,ccreatetime);{ date created }
575:C        4       cinfo:='LIF level  '; cinfo[11]:=chr(vol.version+ord('0'));
576:C        4     end;       { volume info }
577:C        3   end;
578:S
579:D  -326  2 (*****************************************************************************)
580:D        2   procedure docat(anyvar cat : catarray);
581:D        3   var
582:D   -12  3     di, dstart, dnum    : integer;
583:D   -13  3     done        : boolean;
584:S
585:D        3     procedure zerodatetime(var dr:daterec; var tr:timerec);
586:C        4     begin
587:C        4       dr.year:=0; dr.month:=3; dr.day:=1;
588:C        4       tr.hour:=0; tr.minute:=0; tr.centisecond:=0;
589:C        4     end;
590:S
591:C        3   begin (docat)
592:C        3     opendir;    checkvolid;
593:C        3     dstart:=f.fpos;   dnum:=f.fpeof;   f.fpeof:=0;
594:C        3     di:=1;    seek(dir^,di);    done:=false;
595:C        3     while (f.fpeof<dnum) and (not done) do
596:C        4     begin      { get file info }
597:C        4       read(dir^,dentry);
598:C        4       if dentry.ftype=-1 then done:=true
599:C        5       else
600:C        5       { don't show temporary or purged files }
```

```
 601:C        5       if (dentry.ftype<>0) and
 602:C        6          ((dentry.fdate[3]<>9) or (dentry.fdate[4]<>9)) then
 603:C        6       begin   { count this entry }
 604:C        6         if dstart<=0 then     { skip to start index }
 605:C        7         begin { report this entry }
 606:C        7           f.fpeof := f.fpeof+1;
 607:C        7           with dentry, cat[f.fpeof] do
 608:C        8           begin
 609:C        8             lifnametostr(fname,cname);
 610:C        8             ceft    := ftype;   liftofkind(ceft,ckind);
 611:C        8             cpsize  := fsize*256;
 612:C        8             if ftype=-5622 (workstation data) then clsize:=extension
 613:C        9             else clsize := cpsize;
 614:C        8             cstart     := fstart*256;
 615:C        8             cblocksize := 256;
 616:C        8             zerodatetime(ccreatedate,ccreatetime);
 617:C        8             cvtdatetime(fdate,clastdate,clasttime);
 618:C        8             cextra1 := extension;  cextra2 := volnumber;
 619:C        8             if lastvol then cinfo:='' else cinfo := 'continued';
 620:C        8           end { with }
 621:C        8         end   { report this entry }
 622:C        7         else dstart := dstart-1;
 623:C        6       end;    { count this entry }
 624:C        4       di:=di+1;
 625:C        4       if di>dlast then done := true;
 626:C        4     end;        { while }
 627:C        3   end;  (docat)
 628:S
 629:D  -326  2 {*************************************************************************}
 630:D        2   function findfile(temporary:boolean; ftypecode:integer16):boolean;
 631:D        3   var
 632:D   -1   3     found  : boolean;
 633:D  -12   3     tempname    : lifname;
 634:S
 635:C        3   begin {find file}
 636:C        3     if not f.fanonymous then strtolifname(f.ftid,tempname);
 637:C        3     found:=false;       dindex:=1;      seek(dir^,dindex);
 638:C        3     repeat
 639:C        4       read(dir^,dentry);
 640:C        4       with dentry do
 641:C        5       begin
 642:C        5         if ftype=-1 then dindex:=dlast
 643:C        6         else
 644:C        6         if ftype<>0 then
 645:C        7         begin   { check this entry }
 646:C        7           if f.fanonymous then found:=(dentry.fstart*256 = f.fileid)
 647:C        8           else
 648:C        8           if (tempname=fname) then
 649:C        9             if ((ftypecode=0) or (ftypecode=ftype)) then
 650:C       10             begin
 651:C       10               if temporary then found:=(fdate[3]=9) and (fdate[4]=9)
 652:C       11                            else found:=(fdate[3]<>9) or (fdate[4]<>9);
 653:C       10             end;
 654:C        7         end;
 655:C        5       end;      { with }
 656:C        4       if not found then dindex:=dindex+1;
 657:C        4     until (dindex>dlast) or found;
 658:C        3     findfile:=found;
 659:C        3   end; {find file}
 660:S
```



```
 661:D  -326  2 {*************************************************************************}
 662:D        2   procedure purgef;
 663:C        3   begin
 664:C        3     dentry.ftype := 0; writedir(dir^,dindex,dentry); flushdir;
 665:C        3   end;
 666:S
 667:D  -326  2 {*************************************************************************}
 668:D        2   procedure dopurgename;
 669:C        3   begin
 670:C        3     opendir;    checkvolid;     checkftitle;
 671:C        3     if findfile(false,0) then purgef
 672:C        4                          else ioresult := ord(inofile);
 673:C        3   end;
 674:S
 675:D  -326  2   {*************************************************************************}
 676:D        2   procedure getspace(space:integer; var srec:spacerec);
 677:D        3   var
 678:D    -3  3     fixed, done, opening          : boolean;
 679:D   -12  3     lastused, lastopening         : integer;
 680:D   -16  3     dataspace                     : integer;
 681:D   -48  3     mostavail, nextavail          : spacerec;
 682:S
 683:D        3     procedure shuffle;
 684:D        4     var
 685:D   -32  4       tempentry            : direntry;
 686:D   -44  4       increment, here, hole : integer;
 687:S
 688:C        4     begin       { move directory entries to open a required space }
 689:C        4       here := mostavail.here;
 690:C        4       hole := mostavail.hole;
 691:C        4       if here<=hole then
 692:C        5       begin
 693:C        5         if hole<dlast then
 694:C        6         begin   { move logical eod if required }
 695:C        6           readdir(dir^,hole,tempentry);
 696:C        6           if tempentry.ftype=-1 then
 697:C        7           begin
 698:C        7             writedir(dir^,hole + 1,tempentry);
 699:C        7             anychange := true;
 700:C        7           end;
 701:C        6         end;
 702:C        5         increment := -1;
 703:C        5       end
 704:C        5       else
 705:C        5         increment := 1;
 706:C        4       while hole<>here do
 707:C        5       begin
 708:C        5         readdir(dir^,hole+increment,tempentry);
 709:C        5         writedir(dir^,hole,tempentry);      anychange := true;
 710:C        5         hole := hole + increment;
 711:C        5       end;
 712:C        4       tempentry.ftype:=0;
 713:C        4       writedir(dir^,here,tempentry); anychange := true;
 714:C        4     end;        { shuffle }
 715:S
 716:D        3     procedure allocate(var srec : spacerec; eod : boolean);
 717:C        4     begin
 718:C        4       srec.ssize  := dataspace;
 719:C        4       srec.sstart := lastused + 1;
 720:C        4       if eod then
```

```
   721:C        5       begin { end of directory allocation }
   722:C        5         if opening then srec.here := lastopening
   723:C        6                    else srec.here := dindex;
   724:C        5         srec.hole := srec.here;
   725:C        5       end
   726:C        5       else
   727:C        5       begin { middle of directory allocation }
   728:C        5         if opening then srec.here := lastopening
   729:C        6         else
   730:C        6           if lastopening>0 then srec.here := dindex-1
   731:C        7                            else srec.here := dindex;
   732:C        5         srec.hole := lastopening;
   733:C        5       end;
   734:C        4     end;        { allocate }
   735:S
   736:D        3     procedure checkspace(eod : boolean);
   737:D        4     var
   738:D    -4  4       temp      : integer;
   739:D    -5  4       check2    : boolean;
   740:C        4     begin
   741:C        4       if fixed and (dataspace>=space) then
   742:C        5       begin     { fixed space check }
   743:C        5         if (mostavail.ssize=0) { no space yet } or
   744:C        6            (dataspace=space) { exact fit } or
   745:C        6            { this space is bigger than previous good fit }
   746:C        6            (((dataspace-space)>(mostavail.ssize-space)) and
   747:C        6            (mostavail.ssize<>space)) then allocate(mostavail,eod);
   748:C        5         if mostavail.hole>0 then done := true;
   749:C        5       end;      { fixed space check }
   750:S
   751:C        4       if not fixed then
   752:C        5       begin     { biggest or 2nd biggest }
   753:C        5         check2 := true;
   754:C        5         if (dataspace>=mostavail.ssize) then
   755:C        6         begin   { check biggest space }
   756:C        6           if (dataspace>mostavail.ssize) then
   757:C        7           begin { new biggest space }
   758:C        7             nextavail := mostavail;       { demote to second biggest }
   759:C        7             allocate(mostavail,eod);
   760:C        7             check2    := false;
   761:C        7           end
   762:C        7           else
   763:C        7           if not eod then
   764:C        8           begin { same size space }
   765:C        8             if opening then temp := lastopening
   766:C        9             else
   767:C        9               if lastopening>0 then temp := dindex -1
   768:C       10                               else temp := dindex;
   769:C        8             if (abs(temp-lastopening)<=abs(mostavail.here-mostavail.hole)) or
   770:C        9                (lastopening=0) then
   771:C        9             with mostavail do
   772:C       10             begin       { this causes shorter shuffle }
   773:C       10               ssize  := dataspace;
   774:C       10               sstart := lastused + 1;
   775:C       10               here   := temp;
   776:C       10               hole   := lastopening;
   777:C       10             end;
   778:C        8           end;  { same size space }
   779:C        6         end;    { check biggest space }
   780:C        5         if check2 then
```



```
   781:C        6           if (dataspace>=nextavail.ssize) and (space<0) then
   782:C        7           begin   { check 2nd biggest space }
   783:C        7             if dataspace>nextavail.ssize then allocate(nextavail,eod)
   784:C        8             else
   785:C        8             if not eod then
   786:C        9             begin { same size space }
   787:C        9               if opening then temp := lastopening
   788:C       10               else
   789:C       10                 if lastopening>0 then temp := dindex -1
   790:C       11                                  else temp := dindex;
   791:C        9               if (abs(temp-lastopening)<=abs(nextavail.here-nextavail.hole)) or
   792:C       10                  (lastopening=0) then
   793:C       10               with nextavail do
   794:C       11               begin       { this causes shorter shuffle }
   795:C       11                 ssize  := dataspace;
   796:C       11                 sstart := lastused + 1;
   797:C       11                 here   := temp;
   798:C       11                 hole   := lastopening;
   799:C       11               end;
   800:C        9             end;
   801:C        7           end;    { same size space }
   802:C        5       end;      { biggest or 2nd biggest }
   803:C        4     end; {checkspace}
   804:S
   805:D        3     procedure checkentry;
   806:C        4     begin       { checkentry }
   807:C        4       if dentry.ftype=-1 then
   808:C        5       begin   { logical end of directory }
   809:C        5         dataspace := (volsize div 256) - lastused - 1;
   810:C        5         if dataspace>0 then checkspace(true);{check space at end of directory}
   811:C        5         { set lastopening so outer proc. won't think directory is full }
   812:C        5         if lastopening=0 then lastopening := dindex;
   813:C        5         if (mostavail.hole=0) and (mostavail.sstart>0) then
   814:C        6           mostavail.hole := lastopening;
   815:C        5         if (nextavail.hole=0) and (nextavail.sstart>0) then
   816:C        6           nextavail.hole := lastopening;
   817:C        5         done := true;   lastused := (volsize div 256) + 1;
   818:C        5       end
   819:C        5       else
   820:C        5       if dentry.ftype=0 then
   821:C        6       begin   { hole in the directory }
   822:C        6         if not opening then
   823:C        7         begin
   824:C        7           opening     := true;
   825:C        7           lastopening := dindex;
   826:C        7           if mostavail.sstart>0 then
   827:C        8           with mostavail do
   828:C        9           begin   { adjust fixed/biggest space }
   829:C        9             if hole=0 then hole := lastopening
   830:C       10             else
   831:C       10               if (abs(hole-here)>abs(lastopening-here)) then
   832:C       11               begin   { hole changed direction from entry }
   833:C       11                 here := here + 1;
   834:C       11                 hole := lastopening;
   835:C       11               end;
   836:C        9             if fixed then done := true;
   837:C        9           end; { with }
   838:S
   839:C        7           if (space<0) and (nextavail.sstart>0) then
   840:C        8           with nextavail do
```

```
 841:C        9           begin     { adjust second biggest space }
 842:C        9             if hole=0 then hole := lastopening
 843:C       10             else
 844:C       10               if (abs(hole-here)>abs(lastopening-here)) then
 845:C       11               begin     { hole changed direction from entry }
 846:C       11                 here := here + 1;
 847:C       11                 hole := lastopening;
 848:C       11               end;
 849:C        9           end;
 850:S
 851:C        7         end;
 852:C        6       end       { hole in the directory }
 853:C        6       else
 854:C        6       begin   { have file entry }
 855:C        6         dataspace := dentry.fstart - lastused - 1;
 856:C        6         if dataspace>0 then checkspace(false);
 857:C        6         { if no dataspace yet, move lastopening to end of series }
 858:C        6         if (mostavail.sstart=0) and opening then lastopening := dindex - 1;
 859:C        6         opening  := false;
 860:C        6         lastused := dentry.fstart + dentry.fsize - 1;
 861:C        6       end;
 862:C        4     end;  {checkentry}
 863:S
 864:C        3   begin {getspace}
 865:C        3     dindex := 1;
 866:C        3     seek(dir^,dindex);
 867:C        3     if space>0 then begin fixed:=true; space:=(space+255) div 256; end
 868:C        4                else fixed:=false;
 869:C        3     lastused := dend - 1 ;     lastopening     := 0;
 870:C        3     mostavail.sstart:= 0;      nextavail.sstart:= 0;
 871:C        3     mostavail.ssize := 0;      nextavail.ssize := 0;
 872:C        3     mostavail.hole  := 0;      nextavail.hole  := 0;
 873:C        3     mostavail.here  := 0;      nextavail.here  := 0;
 874:C        3     done        := false;      opening     := false;
 875:C        3     repeat
 876:C        4       read(dir^,dentry);
 877:C        4       checkentry;
 878:C        4       dindex := dindex + 1;
 879:C        4       if not done then done := dindex>dlast;
 880:C        4     until done;
 881:S
 882:C        3     if lastopening=0 then badio(idirfull)
 883:C        4     else
 884:C        4     begin       { have at least one directory opening }
 885:C        4       if (mostavail.sstart=0) or not fixed then
 886:C        5       begin
 887:C        5         dataspace := (volsize div 256) - lastused - 1;
 888:C        5         if dataspace>0 then
 889:C        6         begin
 890:C        6           dindex := dlast + 1; {insurance policy }
 891:C        6           if (not fixed and (dataspace>mostavail.ssize)) or
 892:C        7              ((space<0) and (dataspace>nextavail.ssize)) or
 893:C        7              (fixed and (dataspace>=space)) then checkspace(false);
 894:C        6         end;
 895:C        5       end;
 896:C        4       if mostavail.sstart=0 then badio(inoroom);
 897:S
 898:C        4       if fixed then mostavail.ssize := space
 899:C        5       else
 900:C        5       with mostavail do
```



```
 901:C        6       begin     { final decision for non fixed space }
 902:C        6         { watch for [*] type allocation }
 903:C        6         { biggest of (1/2 biggest or 2nd biggest) } { include any odd sector }
 904:C        6         if space<0 then ssize:=(ssize div 2) + (ssize mod 2);
 905:C        6         if nextavail.ssize>=ssize then mostavail:=nextavail;
 906:C        6       end;
 907:C        4       shuffle;  { allign dataspace and directory entry }
 908:C        4       srec := mostavail;
 909:C        4     end;
 910:C        3   end;  {getspace}
 911:S
 912:D   -326  2 {*****************************************************************************}
 913:D        2   procedure finishfib;
 914:C        3   begin
 915:C        3     f.fileid    := dentry.fstart*256;
 916:C        3     f.fpeof     := dentry.fsize*256;
 917:C        3     if f.fisnew  then f.fleof := 0
 918:C        4                  else f.fleof := f.fpeof;
 919:C        3     f.fmodified := f.fisnew;
 920:C        3     if not f.fbuffered then f.am := amtable^[UNTYPEDFILE]
 921:C        4     else
 922:C        4      if f.fistextvar then f.am := amtable^[f.fkind]
 923:C        5                      else f.am := amtable^[datafile];
 924:C        3   end;  {finishfib}
 925:S
 926:D   -326  2 {*****************************************************************************}
 927:D        2   procedure opennew;
 928:D        3   var
 929:D    -16  3     space       : spacerec;
 930:C        3   begin
 931:C        3     getspace(f.fpos,space);
 932:C        3     dindex := space.here;
 933:C        3     { fill in the lif directory entry }
 934:C        3     {file name}
 935:C        3     if f.fanonymous then dentry.fname := 'anonymous '
 936:C        4       else strtolifname(f.ftid,dentry.fname);
 937:C        3     dentry.ftype     := f.feft;
 938:C        3     dentry.fstart    := space.sstart;
 939:C        3     dentry.fsize     := space.ssize;
 940:C        3     setdate(dentry.fdate);
 941:C        3     dentry.fdate[3]  := 9; dentry.fdate[4] := 9; { mark as temporary }
 942:C        3     dentry.lastvol   := true;
 943:C        3     dentry.volnumber := 1;
 944:C        3     dentry.extension := 0;
 945:C        3     writedir(dir^,dindex,dentry); {write it out} flushdir;
 946:C        3     { finish the file fib }
 947:C        3     finishfib;
 948:C        3   end;  {opennew}
 949:S
 950:D   -326  2 {*****************************************************************************}
 951:D        2   procedure openold;
 952:D    -2  3   var fk:filekind;
 953:C        3   begin
 954:C        3     liftofkind(dentry.ftype,fk);  f.fkind:=fk;
 955:C        3     finishfib;
 956:C        3     with dentry do
 957:C        4     begin
 958:C        4       f.fstartaddress := 0;
 959:C        4       f.feft := ftype;
 960:C        4       if ftype=efttable^[datafile] then f.fleof := extension
```

```
   961:C        5                                  else f.fstartaddress := extension;
   962:C        4     end;
   963:C        3   end;
   964:S
   965:D  -326  2 (*****************************************************************************)
   966:D        2   procedure openf;
   967:C        3   begin
   968:C        3     opendir; checkvolid;
   969:C        3     if not (f.fanonymous and f.fisnew) then  checkftitle;
   970:C        3     if f.fisnew then
   971:C        4     begin   { new temp file }
   972:C        4       if not mediavalid then cleandir;
   973:C        4       if f.fanonymous then opennew
   974:C        5       else
   975:C        5       if findfile(f.fisnew,0) then ioresult:=ord(idupfile)
   976:C        6                              else opennew;
   977:C        4     end
   978:C        4     else    { existing permanent file }
   979:C        4     begin
   980:C        4       if findfile(f.fisnew,0) then
   981:C        5       begin
   982:C        5         openold;
   983:C        5         if not mediavalid then cleandir;
   984:C        5       end
   985:C        5       else ioresult:=ord(inofile);
   986:C        4     end;
   987:C        3   end;  { openf }
   988:S
   989:D  -326  2 (*****************************************************************************)
   990:D        2   procedure closef;
   991:D        3   var
   992:D   -4   3     temp : integer;
   993:S
   994:C        3   begin
   995:C        3     if f.fisnew then
   996:C        4     begin     { purge old file }
   997:C        4       temp:=dindex;
   998:C        4       if findfile(false,0) then purgef;
   999:C        4       dindex:=temp; readdir(dir^,dindex,dentry);
  1000:C        4     end;
  1001:C        3     if f.fmodified then
  1002:C        4     with dentry do
  1003:C        5     begin       { rewrite the directory entry }
  1004:C        5       if ftype=efttable^[datafile] then extension := f.fleof
  1005:C        6                                    else extension := f.fstartaddress;
  1006:C        5       temp:=(f.fleof+255) div 256;
  1007:C        5       if temp<fsize then fsize:=temp;
  1008:C        5       setdate(fdate);
  1009:C        5       writedir(dir^,dindex,dentry); flushdir;
  1010:C        5     end;
  1011:C        3   end;
  1012:S
  1013:D  -326  2 (*****************************************************************************)
  1014:D        2   procedure stretchf;
  1015:D        3   var
  1016:D    -2  3     found, eod  : boolean;
  1017:D   -18  3     tempindex, filestart, dataspace, reqsize  : integer;
  1018:S
  1019:C        3   begin {stretchf}
  1020:C        3     tempindex := dindex;  found := false;   eod := false;
```



```
  1021:C        3     filestart := dentry.fstart;
  1022:C        3     reqsize := (f.fpos + 255) div 256;  { round requested size 25jan83}
  1023:C        3     if reqsize>dentry.fsize then
  1024:C        4     begin
  1025:C        4       while (not found) and (not eod) do
  1026:C        5       with dentry do
  1027:C        6       begin
  1028:C        6         tempindex := tempindex + 1;
  1029:C        6         if tempindex>dlast then eod := true
  1030:C        7         else
  1031:C        7         begin
  1032:C        7           readdir(dir^,tempindex,dentry);
  1033:C        7           if ftype=-1 then eod := true
  1034:C        8           else
  1035:C        8           if ftype<>0 then
  1036:C        9           begin
  1037:C        9             found     := true;
  1038:C        9             dataspace := dentry.fstart - filestart;
  1039:C        9           end;
  1040:C        7         end;
  1041:C        6       end; { while with }
  1042:C        4       if eod then
  1043:C        5       begin     { dataspace is from begining of file to end of volume }
  1044:C        5         found:=true;                                  {25jan83}
  1045:C        5         dataspace := (volsize div 256) - filestart;   {25jan83}
  1046:C        5       end;
  1047:C        4       if found then
  1048:C        5       if dataspace>=reqsize then                    {25jan83}
  1049:C        6       begin   { will stretch }
  1050:C        6         readdir(dir^,dindex,dentry);
  1051:C        6         { allow requested space + half of excess space  25jan83 }
  1052:C        6         dentry.fsize := (reqsize + dataspace) div 2;  {25jan83}
  1053:C        6         writedir(dir^,dindex,dentry);  flushdir;
  1054:C        6         f.fpeof      := dentry.fsize * 256;
  1055:C        6       end;    { will stretch }
  1056:C        4     end;
  1057:C        3   end;  { stretchf }
  1058:S
  1059:D  -326  2 (*****************************************************************************)
  1060:D        2   procedure changefname(anyvar n:string255);
  1061:D        3   var
  1062:D    -4  3     tempindex : integer;
  1063:D    -5  3     ok        : boolean;
  1064:C        3   begin
  1065:C        3     if f.fanonymous then badio(ibadrequest);
  1066:C        3     opendir;    checkvolid;     checkftitle;
  1067:C        3     { find the original (permanent file) }
  1068:C        3     if not findfile(false,0) then badio(inofile);
  1069:C        3     { change the name }
  1070:C        3     tempindex := dindex;
  1071:C        3     if (strlen(n)=0) or (strlen(n)>tidleng) then badio(ibadtitle);
  1072:C        3     f.ftid := n;
  1073:C        3     if findfile(false,0) then badio(idupfile)
  1074:C        4     else
  1075:C        4     begin
  1076:C        4       readdir(dir^,tempindex,dentry);
  1077:C        4       strtolifname(f.ftid,dentry.fname);
  1078:C        4       writedir(dir^,tempindex,dentry); flushdir;
  1079:C        4     end
  1080:C        4   end;  { changefname }
```

```
1081:S
1082:D  -326  2 {*****************************************************************************}
1083:D        2   procedure dooverwritefile;
1084:C        3   begin
1085:C        3     opendir;  checkvolid;
1086:C        3     if f.fanonymous then badio(ibadrequest);
1087:C        3     checkftitle;
1088:C        3     f.fisnew := true;
1089:C        3     if findfile(false,0) then
1090:C        4     begin       { existing file }
1091:C        4       if not mediavalid then cleandir;
1092:C        4       openold;  f.fleof := 0;   { setup fib then reset logical eof }
1093:C        4       setdate(dentry.fdate);
1094:C        4       dentry.fdate[3] := 9; dentry.fdate[4] := 9; { now a temporary file }
1095:C        4       writedir(dir^,dindex,dentry);     anychange := true;
1096:C        4     end
1097:C        4     else
1098:C        4     begin       { new file }
1099:C        4       if not mediavalid then cleandir;
1100:C        4       opennew;
1101:C        4     end;
1102:C        3   end;  { dooverwritefile }
1103:S
1104:D  -326  2 {*****************************************************************************}
1105:D        2   procedure nowopen;
1106:C        3   begin
1107:C        3     opendir;
1108:C        3     if f.fvid<>volid then ioresult:=ord(ilostfile)
1109:C        4     else
1110:C        4     if not findfile(f.fisnew,f.feft) then ioresult:=ord(ilostfile);
1111:C        3     goodio;
1112:C        3   end;
1113:S
1114:D  -326  2 {*****************************************************************************}
1115:D  -326  2 {*****************************************************************************}
1116:C        2   begin {lifdam}
1117:C        2     lockup;
1118:C        2     mediavalid := unitable^[unum].umediavalid;
1119:C        2     { tell TM to keep quiet about media changes for a while }
1120:C        2     unitable^[unum].umediavalid   := true;
1121:C        2     unitable^[unum].ureportchange := false;
1122:C        2     fibp(dir)^.funit := unum; fibp(dir)^.freadable := false;
1123:C        2     ioresult          := ord(inoerror);
1124:C        2     anychange         := false;
1125:C        2     try
1126:C        3       case request of
1127:C        4         openfile  : begin f.fisnew := false; openf; end;
1128:C        4         createfile: begin f.fisnew := true; openf; end;
1129:C        4         overwritefile : dooverwritefile;
1130:C        4         closefile : if f.fmodified then begin nowopen; closef; end;
1131:C        4         purgefile : begin nowopen; purgef; end;
1132:C        4         stretchit : begin nowopen; stretchf; end;
1133:C        4         changename: changefname(f.fwindow^);
1134:C        4         getvolumename:
1135:C        4           begin
1136:C        4             ok := lifvol; strtoany(unitable^[unum].uvid,f);
1137:C        4           end;
1138:C        4         setvolumename:
1139:C        4           if lifvol then
1140:C        5           begin
```



```
1141:C        5             strtopac(f,6,vol.volname,true);
1142:C        5             with unitable^[unum] do
1143:C        6             begin
1144:C        6               call(tm,fibp(dir),writebytes,vol,sizeof(lvheader),0);
1145:C        6               goodio;   anychange := true;
1146:C        6             end;
1147:C        5           end;
1148:C        4         purgename     : dopurgename;
1149:C        4         getvolumedate: getsdate(f);
1150:C        4         setvolumedate: setsdate(f);
1151:C        4         crunch        : crunchv;
1152:C        4         catalog       : docat(f.fwindow^);
1153:C        4         opendirectory: doopendirectory(f.fwindow^);
1154:C        4         closedirectory:begin end;
1155:C        4         makedirectory: domakedirectory(f.fwindow^);
1156:C        4         openunit,
1157:C        4         openvolume    : begin
1158:C        4                           cleanup;
1159:C        4                           unblockeddam(f,unum,request);
1160:C        4                           mediavalid := unitable^[unum].umediavalid;
1161:C        4                         end;
1162:C        4         setunitprefix: if strlen(f.ftitle)>0 then badio(ibadtitle);
1163:C        4         stripname     : begin
1164:C        4                           checkftitle; setstrlen(f.ftitle,0);
1165:C        4                           strtolifname(f.ftid,dentry.fname);
1166:C        4                         end;
1167:C        4         otherwise
1168:C        4           ioresult := ord(ibadrequest);
1169:C        4       end;   { case request }
1170:C        3       if anychange then flushdir;
1171:C        3       cleanup;  { fix umediavalid and ureportchange }
1172:C        3     recover
1173:C        3     begin
1174:C        3       cleanup;  { fix umediavalid and ureportchange }
1175:C        3       if (escapecode<0) and (escapecode<>-10) then
1176:C        4         begin lockdown; escape(escapecode); end;
1177:C        3     end;
1178:C        2     lockdown;
1179:C        2   end;
1180:D        1   procedure installlifdam;
1181:C        2   begin
1182:C        2     if dir=nil then new(dir);
1183:C        2   end;
1184:C        1 end;  { module }
1185:D        1   import lifmodule,loader;
1186:C        1 begin { instlifdam}
1187:C        1   installlifdam;
1188:C        1   markuser;
1189:C        1 end.{ rev 2.2x1 }

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

LOCKMOD

Description

LOCKMOD provides the function: LOCK and the procedures: UNLOCK and WAITFORLOCK for the sharing of SRM files opened with the LOCKABLE option as the third parameter.

Usage

Only useful with SRM (Shared Resource Managment).

Notes

Supplied in SYSVOL:LIBRARY.

```
     1:S                                                    (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1983.
     4:S          All rights are reserved.  Copying or other
     5:S          reproduction of this program except for archival
     6:S          purposes is prohibited without the prior
     7:S          written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                       RESTRICTED RIGHTS LEGEND
    11:S
    12:S          Use, duplication, or disclosure by the Government
    13:S          is subject to restrictions as set forth in
    14:S          paragraph (b) (3) (B) of the Rights in Technical
    15:S          Data and Computer Software clause in
    16:S          DAR 7-104.9(a).
    17:S
    18:S          HEWLETT-PACKARD COMPANY
    19:D        0 Fort Collins, Colorado                        *)
    20:S
    21:S
    22:D        0 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'$
    23:S
    24:D        0 $sysprog$
    25:D        0 module lockmodule;
    26:S
    27:D        1 import sysglobals;
    28:D        1 export
    29:S
    30:D        1 function  lock          (anyvar f : fib) : boolean;
    31:D        1 procedure unlock        (anyvar f : fib);
    32:D        1 procedure waitforlock   (anyvar f : fib);
    33:S
    34:D        1 implement
    35:S
    36:D        1 (*************************************************************************)
    37:D        1 function  lock          (anyvar f : fib) : boolean;
    38:C        2 begin
    39:C        2   ioresult := ord(inoerror);
    40:C        2   with f do
    41:C        3     if not flockable then
    42:C        4       ioresult := ord(inotlockable)
    43:C        4     else
    44:C        4       if flocked then
    45:C        5         ioresult := ord(ifilelocked)
    46:C        5       else
    47:C        5         begin
    48:C        5           fwaitonlock := false;
    49:C        5           call(unitable^[funit].dam,f,funit,lockfile);
    50:C        5         end;
    51:C        2   if ioresult = ord(inoerror) then
    52:C        3     lock := true
    53:C        3   else
    54:C        3     begin
    55:C        3       lock := false;
    56:C        3       if ioresult <> ord(ifilelocked) then
    57:C        4         escape(-10);
    58:C        3     end;
    59:C        2 end;
    60:S
```



```
    61:D        1 (*************************************************************************)
    62:D        1 procedure unlock        (anyvar f : fib);
    63:C        2 begin
    64:C        2   ioresult := ord(inoerror);
    65:C        2   with f do
    66:C        3     if not flockable then
    67:C        4       ioresult := ord(inotlockable)
    68:C        4     else
    69:C        4       if not flocked then
    70:C        5         ioresult := ord(ifileunlocked)
    71:C        5       else
    72:C        5         call(unitable^[funit].dam,f,funit,unlockfile);
    73:C        2   if ioresult <> ord(inoerror) then
    74:C        3     escape(-10);
    75:C        2 end;
    76:S
    77:D        1 (*************************************************************************)
    78:D        1 procedure waitforlock   (anyvar f : fib);
    79:C        2 begin
    80:C        2   ioresult := ord(inoerror);
    81:C        2   with f do
    82:C        3     if not flockable then
    83:C        4       ioresult := ord(inotlockable)
    84:C        4     else
    85:C        4       if flocked then
    86:C        5         ioresult := ord(ifilelocked)
    87:C        5       else
    88:C        5         begin
    89:C        5           fwaitonlock := true;
    90:C        5           call(unitable^[funit].dam,f,funit,lockfile);
    91:C        5         end;
    92:C        2   if ioresult <> ord(inoerror) then
    93:C        3     escape(-10);
    94:C        2 end;
    95:S
    96:C        1 end.  (lockmodule)

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

M68KSYS

Description

M68KSYS provides the command interpreter for the Pascal operating system.

Requirements

SYSGLOBALS, FS, LOADER, ASM, MISC, SYSDEVS, and LDR.

```
     1:S          (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1984.
     4:S          All rights are reserved.  Copying or other
     5:S          reproduction of this program except for archival
     6:S          purposes is prohibited without the prior
     7:S          written consent of Hewlett-Packard Company.
     8:S
     9:S                        RESTRICTED RIGHTS LEGEND
    10:S
    11:S          Use, duplication, or disclosure by the Government
    12:S          is subject to restrictions as set forth in
    13:S          paragraph (b) (3) (B) of the Rights in Technical
    14:S          Data and Computer Software clause in
    15:S          DAR 7-104.9(a).
    16:S
    17:S          HEWLETT-PACKARD COMPANY
    18:S          Fort Collins, Colorado                          *)
    19:D        0
    20:S
    21:S
    22:S
    23:D        0 $modcal$
    24:D        0 $iocheck off$ $debug off$ $range off$ $ovflcheck off$
    25:D        0 $heap_dispose off$
    26:D        0 $search 'INITLOAD', 'ASM', 'INIT', 'SYSDEVS'$
    27:S
    28:D        0 program cmd(input,output,keyboard);
    29:D        1
    30:D        1 module ci;
    31:S
    32:D        1 import
    33:D        1   sysglobals,fs,loader,asm,misc,sysdevs,ldr;
    34:S
    35:D        1 export
    36:S
    37:S
    38:D        1 type
    39:D        1   sysfiles = (assembler,compiler,editor,filer,librarian,library);
    40:D        1   sysfilevols  = array [sysfiles] of string[6];
    41:D        1   sysfilenames = array [sysfiles] of fid;
    42:S
    43:D        1
    44:D        1   inforec = record                                    (*FILE INFORMATION*)
    45:D        1              errsym,errblk,errnum: integer;          (*ERROR STUFF IN EDIT*)
    46:D        1              gotsym,gotcode: boolean;               (*TITLES ARE MEANINGFUL*)
    47:D        1              workfid,symfid,codefid,errfid: fid;      (*PERM&CUR WORKFILES*)
    48:D        1            end (*INFOREC*) ;
    49:S
    50:D        1   inforecptr = ^inforec;
    51:D        1   cmdprocedure = procedure;
    52:D        1   cmdprocptr= ^cmdprocedure;
    53:S
    54:D    -4  1 var streamfib: ^text;                              (*FOR SYSTEM STREAM FILE*)
    55:D  -736  1     filename: sysfilenames;
    56:D  -740  1     tioresult:integer; {to save previous ioresult}
    57:D  -862  1     chainfile: fid;
    58:D  -864  1     chaining: (nochain,progchain,streamchain);
    59:D  -868  1     userinfo:  inforecptr;
    60:D  -869  1     versionup: boolean;
```



```
    61:D  -874  1     cmdcharhook : cmdprocptr;
    62:D  -875  1     ci_idle: boolean;
    63:D  -876  1     ci_cmd: char;
    64:D  -877  1     keystream: boolean; { stream file is original file }
    65:S
    66:D  -877  1   procedure homecursor  ;
    67:D  -877  1   procedure clearscreen  ;
    68:D  -877  1   procedure clearline  ;
    69:D  -877  1   procedure prompt (pl: string80);
    70:D  -877  1   function  getchar (flushit: boolean): char;
    71:D  -877  1   function  uppercase (ch: char): char;
    72:D  -877  1   procedure chain(filename: fid);
    73:D  -877  1   procedure startstream(filename: fid);
    74:D  -877  1   function  streaming: boolean;
    75:S
    76:S
    77:D  -877  1 procedure systemstartup;
    78:S
    79:D  -877  1 implement
    80:S
    81:D  -877  1 type
    82:D  -877  1   monthtype = array [0..15] of packed array [1..3] of char;
    83:D  -877  1 const
    84:D  -877  1   months = monthtype [ '???','Jan','Feb','Mar','Apr','May','Jun',
    85:D  -877  1                        'Jul','Aug','Sep','Oct','Nov','Dec','???',
    86:D  -877  1                        '???','???' ];
    87:S
    88:D        1 function streaming: boolean;
    89:C        2 begin with fibp(streamfib)^ do
    90:C        3   streaming := freadable and (fpos < fleof);
    91:C        2 end;
    92:S
    93:D  -122  1 procedure startstream(filename: fid);
    94:C        2 begin
    95:C        2   chainfile := filename;
    96:C        2   chaining := streamchain;
    97:C        2   escape(0);
    98:C        2 end;
    99:S
   100:D  -122  1 procedure chain(filename: fid);
   101:C       2 begin
   102:C       2   chainfile := filename;
   103:C       2   fixname(chainfile, codefile);
   104:C       2   chaining := progchain;
   105:C       2   escape(0);
   106:C       2 end;
   107:S
   108:D       1 procedure disptime;
   109:D    -4 2 var time: timerec;
   110:D    -6 2     date: daterec;
   111:D   -14 2     x,y: integer;
   112:D   -18 2     second, dayy: shortint;
   113:C       2 begin
   114:C       2   second := -1;
   115:C       2   dayy := -1;
   116:C       2   setrunlight(chr(idle));
   117:C       2   with fibp(gfiles[0])^ do
   118:C       3     repeat
   119:C       4       call(am, fibp(gfiles[0]), unitstatus, gfiles[0], 0, 0);
   120:C       4       if fbusy and versionup then
```

```
   121:C        5       begin
   122:C        5       systime(time);
   123:C        5       sysdate(date);
   124:C        5       with time, date do
   125:C        6         if (centisecond div 100 <> second) or (day <> dayy) then
   126:C        7           begin
   127:C        7           second := centisecond div 100;
   128:C        7           fgetxy(output, x, y);
   129:C        7           fgotoxy(output, 25, 3);
   130:C        7           write(hour:2,':',minute:2,':',second:2);
   131:C        7           if day <> dayy then
   132:C        8             begin
   133:C        8             dayy := day;
   134:C        8             fgotoxy(output, 25, 2);
   135:C        8             writeln(day:2,'-',months[month],'-',year:2);
   136:C        8             end;
   137:C        7           fgotoxy(output, x, y);
   138:C        7           end;
   139:C        5       end;
   140:C        4       if fbusy and ci_idle then call(kbdwaithook);
   141:C        4     until not fbusy or (chaining<>nochain);
   142:C        2   ci_idle:=false;
   143:C        2 end;
   144:S
   145:D        1 procedure dummycmdchar;
   146:C        2 begin end;
   147:S
   148:D        1 procedure initdate;
   149:D        2 var
   150:D    -6  2   thedatetime   : datetimerec;
   151:D   -10  2   ltime         : timerec;
   152:C        2 begin
   153:C        2   with thedatetime, date do
   154:C        3     begin
   155:C        3       sysdate(date);
   156:C        3       if (year=0) and (month=3) and (day=1) then
   157:C        4         begin
   158:C        4           systime(ltime);
   159:C        4           time := ltime;
   160:C        4           call (unitable^[sysunit].dam, thedatetime, sysunit, getvolumedate);
   161:C        4           if ioresult = ord(inoerror) then
   162:C        5             begin
   163:C        5               setsysdate(date);
   164:C        5               with ltime do
   165:C        6                 if (hour = 0) and (minute = 0) then
   166:C        7                   setsystime(time);
   167:C        5             end;
   168:C        4         end;
   169:C        3     end;
   170:C        2 end; (*INITDATE*)
   171:S
   172:D        1 procedure dateset;
   173:D        2 var
   174:D  -82   2   gs: string80;
   175:D  -83   2   changed: boolean;
   176:D  -88   2   i: integer;
   177:D  -92   2   clocktime: timerec;
   178:D  -94   2   clockdate: daterec;
   179:D -100   2   clockdatetime: datetimerec;
   180:S
```



```
   181:D        2   procedure identify;
   182:C        3   begin
   183:C        3     clearscreen;  writeln(output);
   184:C        3     writeln(output);
   185:C        3     writeln(output,'  System date is         ');
   186:C        3     writeln(output,'  Clock time is          ');
   187:C        3     writeln(output);
   188:C        3     writeln(output,'  Workstation            ',fsidc);
   189:C        3     writeln;
   190:C        3     writeln(output,'  Available Global Space ',eglobal-(a5-32768):1,' bytes');
   191:C        3     writeln(output,'  Total Available Memory ',
   192:C        3       eglobal-integer(eheap):1,' bytes');
   193:C        3     writeln;
   194:C        3     writeln(output,'  System  volume:  ',syvid,':');
   195:C        3     writeln(output,'  Default volume:  ',dkvid,':');
   196:C        3     writeln; writeln;
   197:C        3     writeln;
   198:C        3     writeln('Copyright 1984 Hewlett-Packard Company.');
   199:C        3     writeln('All rights are reserved.  Copying or other');
   200:C        3     writeln('reproduction of this program except for archival');
   201:C        3     writeln('purposes is prohibited without the prior');
   202:C        3     writeln('written consent of Hewlett-Packard Company.');
   203:S
   204:C        3     versionup := true;
   205:S
   206:C        3   end;  {IDENTIFY}
   207:S
   208:D        2   function readnumericfield
   209:D        3        (llimit,hlimit:integer; var field:integer): boolean;
   210:D        3   label 1;
   211:D    -7  3   var  gotnum: boolean;  i: integer;  ch: char;
   212:C        3   begin  gotnum := false;  i := 0;
   213:C        3     while strlen(gs) > 0 do
   214:C        4       begin  ch := gs[1];
   215:C        4         if (ch>='0') and (ch<='9') then
   216:C        5           begin  gotnum := true;
   217:C        5             i := 10*i+ord(ch)-ord('0');
   218:C        5             if i>hlimit then i := hlimit+1;
   219:C        5           end
   220:C        5         else
   221:C        5           if gotnum then goto 1;
   222:C        4         strdelete(gs,1,1);
   223:C        4       end;
   224:C        3 1:    if gotnum and (i>=llimit) and (i<=hlimit) then
   225:C        4       begin  readnumericfield := true; field := i  end
   226:C        4     else
   227:C        4       readnumericfield := false;
   228:C        3   end; {READNUMERICFIELD}
   229:S
   230:D        2   function readmonthabbrev (var monthnumber:integer): boolean;
   231:D        3   label 1;
   232:D    -4  3   var  s3: packed array[1..3] of char;
   233:D    -9  3       i: integer;  ch: char;
   234:C        3   begin  i := 0;  s3 := '   ';
   235:C        3     while strlen(gs) > 0 do
   236:C        4       begin  ch := gs[1];
   237:C        4         if ch in ['A'..'Z','a'..'z'] then
   238:C        5           begin  i := i+1;
   239:C        5             if (ch>='A') and (ch<='Z') then
   240:C        6               ch := chr(ord(ch)-ord('A')+ord('a'));
```

```
241:C        5             if i<4 then s3[i] := ch;
242:C        5           end
243:C        5         else
244:C        5           if i>0 then goto 1;
245:C        4         strdelete(gs,1,1);
246:C        4       end;
247:C        3 1:    s3[i] := uppercase(s3[1]);  readmonthabbrev := false;
248:C        3     for i := 1 to 12 do
249:C        4       if months[i] = s3 then
250:C        5         begin  readmonthabbrev := true; monthnumber := i  end;
251:C        3   end;  {READMONTHABBREV}
252:S
253:C        2 begin {DATESET}
254:C        2   changed := false;
255:C        2   sysdate(clockdate);
256:C        2   systime(clocktime);
257:C        2   identify;
258:C        2   prompt('New system date ? '); disptime;
259:C        2   readln(input,gs);
260:C        2   if strlen(gs)>0 then with clockdate do
261:C        4     if readnumericfield(1,31,i) then
262:C        5       begin
263:C        5       changed := true;
264:C        5       day := i;
265:C        5       if readmonthabbrev(i) then month := i;
266:C        5       if readnumericfield(0,99,i) then year := i;
267:C        5       setsysdate(clockdate);
268:C        5       end;
269:C        2   prompt('New system clock time ? '); disptime;
270:C        2   readln(input,gs);
271:C        2   if strlen(gs)>0 then with clocktime do
272:C        4     if readnumericfield(0,23,i) then
273:C        5       begin
274:C        5       changed := true;
275:C        5       hour := i;
276:C        5       if readnumericfield(0,59,i) then minute := i
277:C        6                                   else minute := 0;
278:C        5       if readnumericfield(0,59,i) then centisecond := i*100
279:C        6                                   else centisecond := 0;
280:C        5       setsystime(clocktime);
281:C        5       end;
282:C        2   if changed then
283:C        3     with clockdatetime do
284:C        4       begin
285:C        4         date    := clockdate;
286:C        4         time    := clocktime;
287:C        4         call (unitable^[sysunit].dam, clockdatetime, sysunit, setvolumedate);
288:C        4         identify;
289:C        4       end;
290:C        2 end; {DATESET}
291:S
292:D        1 procedure disableuserisrs;
293:D    -4  2 var i:integer;
294:C        2 begin
295:C        2     call(cleariohook);
296:C        2     interrupttable:=perminttable;
297:C        2 end;
298:S
299:D        1 function uppercase ((CH: CHAR): CHAR);
300:C        2   begin
```



```
301:C        2     if (ch>='a') and (ch<='z')
302:C        3       then uppercase := chr(ord(ch)-32)
303:C        3       else uppercase := ch
304:C        3   end;
305:S
306:D        1 procedure streamdvr(fp: fibp; request: amrequesttype; anyvar buffer: window;
307:D        2                                                       bufsize, position: integer);
308:D        2 type str    =  record s: string255 end;
309:D        2      strptr = ^str;
310:S
311:D    -4  2 var buf: charptr;
312:D    -5  2     c: char;
313:D    -8  2     i: shortint;
314:S
315:D        2   procedure checkctrl(var c:char);
316:D        3   var
317:D    -1  3     ceoln : boolean;
318:C        3   begin { check for control chars in keystream files }
319:C        3     if (c=chr(255)) and keystream then
320:C        4     begin
321:C        4       ceoln:=eoln(streamfib^);
322:C        4       read(streamfib^,c);
323:C        4       if ceoln then c:=chr(13);
324:C        4       c:=chr(ord(c) mod 32);
325:C        4     end;
326:C        3   end;  { checkctrl }
327:S
328:D        2   procedure closedown;
329:C        3   begin
330:C        3     if keystream then close(streamfib^) { if keystream keep the file }
331:C        4                  else close(streamfib^, 'PURGE');
332:C        3     with fp^ do
333:C        4     begin
334:C        4       am := serialtextamhook;
335:C        4       call(am, fp, request, buf^, bufsize, position);
336:C        4     end;
337:C        3   end;
338:S
339:C        2 begin   {STREAMDVR}
340:C        2   ioresult := ord(inoerror);
341:C        2   buf := addr(buffer);
342:C        2   if eof(streamfib^) then closedown
343:C        3   else
344:C        3   with fp^, unitable^[funit] do
345:C        4    case request of
346:C        5    readbytes: while bufsize > 0 do
347:C        6      begin
348:C        6      feoln := eoln(streamfib^);
349:C        6      read(streamfib^, buf^); checkctrl(buf^);
350:C        6      if uisinteractive then
351:C        7        if feoln then call(tm, fp, writeeol,   buf^, 1, 0)
352:C        8                 else call(tm, fp, writebytes, buf^, 1, 0); {echo}
353:C        6      bufsize := bufsize - 1;
354:C        6      if bufsize > 0 then
355:C        7        begin
356:C        7        buf := addr(buf^ , 1);
357:C        7        if eof(streamfib^) then begin
358:C        8                                  closedown;
359:C        8                                  bufsize := 0;
360:C        8                                end;
```

```
 361:C        7        end;
 362:C        6      end;
 363:C        5    readtoeol: with strptr(buf)^ do
 364:C        6      begin
 365:C        6      setstrlen(s, bufsize);
 366:C        6      i := 0;
 367:C        6      while (i < bufsize) and not eoln(streamfib^) do
 368:C        7         begin i := i + 1; read(streamfib^, s[i]); checkctrl(s[i]); end;
 369:C        6      setstrlen(s, i);
 370:C        6      {note there is no need to echo, since readtoeol isn't used interactively}
 371:C        6      if i < bufsize then
 372:C        7       if eof(streamfib^) then
 373:C        8         begin
 374:C        8         buf := addr(s[i]);
 375:C        8         c := s[i];
 376:C        8         bufsize := bufsize - i;
 377:C        8         closedown;
 378:C        8         setstrlen(s, i+ord(s[i]));
 379:C        8         s[i] := c;                        {note that i cannot be 0!}
 380:C        8         end;
 381:C        6      end;
 382:C        5    unitstatus: fbusy := false;
 383:C        5    otherwise call(tm, fp, request, buffer, bufsize, position);
 384:C        5    end;
 385:C        2 end;    {STREAMDVR}
 386:S
 387:D   -82  1 procedure streamopen(sfile: string80; report:boolean);
 388:D   -82  2 const
 389:D   -82  2   parindex = ['0'..'9','A'..'Z'];
 390:D   -82  2 type
 391:D   -82  2   stringptr = ^string255;
 392:D   -82  2 var
 393:D  -254  2   parptr: array['0' .. 'Z'] of stringptr;
 394:D  -262  2   lastior,i: integer;
 395:D  -266  2   heap: ipointer;
 396:D  -930  2   f: text;
 397:D  -948  2   v:vid;
 398:D -1070  2   t:fid;
 399:D -1074  2   segs:integer;
 400:D -1076  2   fk:filekind;
 401:S
 402:S
 403:D        2 function streamsyntax : boolean;
 404:D        3 label
 405:D        3   1;
 406:D        3 var
 407:D    -2  3   c, uc : char;
 408:D  -166  3   s, instr : string80;
 409:D  -169  3   ciseoln, needc, notendofparms : boolean;
 410:S
 411:D        3   procedure getc;
 412:C        4   begin
 413:C        4     ciseoln := eoln(f);
 414:C        4     read(f,c);
 415:C        4     if strlen(instr) = 80 then strdelete(instr,1,10);
 416:C        4     setstrlen(instr,strlen(instr)+1);
 417:C        4     instr[strlen(instr)] := c;
 418:C        4   end;
 419:S
 420:D        3   procedure testioresult;
```



```
 421:C        4   begin
 422:C        4     if ioresult <> ord(inoerror) then
 423:C        5       begin
 424:C        5         lastior := ioresult;
 425:C        5         writeln('Can''t create ',sfile);
 426:C        5         printerror(-10,lastior);
 427:C        5         goto 1;
 428:C        5       end;
 429:C        4   end;
 430:S
 431:C        3 begin   {STREAMSYNTAX}
 432:C        3   mark(heap);
 433:C        3   for uc := '0' to 'Z' do parptr[uc] := nil;
 434:S
 435:C        3   streamsyntax := false;
 436:C        3   notendofparms := true;
 437:S
 438:C        3   if not keystream then  { if keystream then forget this whole operation }
 439:C        4   while not eof(f) do
 440:C        5   begin
 441:S
 442:C        5     instr := '';
 443:C        5     s := '';
 444:C        5     getc;
 445:C        5     needc := false;
 446:S
 447:C        5     if (c = '=') and notendofparms then       {parm line}
 448:C        6     begin
 449:C        6       getc;
 450:C        6       uc := uppercase(c);
 451:C        6       readln(f,s);
 452:C        6       if uc in parindex then
 453:C        7       begin
 454:C        7         writeln(output,s);
 455:C        7         newbytes(parptr[uc],sizeof(string255));
 456:C        7         readln(input,parptr[uc]^);
 457:C        7       end
 458:C        7       else
 459:C        7       begin
 460:C        7         writeln(output,instr,s);
 461:C        7         writeln(output,'':strlen(instr)-1
 462:C        7                         mod syscom^.crtinfo.width,'^');
 463:C        7         printerror(-24,0);
 464:C        7         goto 1;
 465:C        7       end;
 466:C        6     end
 467:C        6     else                                      {line to process}
 468:C        6     begin
 469:C        6       if notendofparms then
 470:C        7       begin
 471:C        7         notendofparms := false;
 472:C        7         rewrite(streamfib^,sfile,'exclusive');
 473:C        7         testioresult;
 474:C        7       end;
 475:S
 476:C        6       repeat
 477:C        7         if needc then getc;
 478:S
 479:C        7         if c = chr(255) then               {control char}
 480:C        8         begin
```

```
481:C        8           getc;
482:C        8           if ciseoln then c := chr(13);
483:C        8           c := chr(ord(c) mod 32);
484:C        8           write(streamfib^,c);
485:C        8         end
486:C        8         else if c = '@' then                {macro expansion}
487:C        9         begin
488:C        9           getc;
489:C        9           if ciseoln then
490:C       10           begin                          {error: no char after @}
491:C       10             writeln(output,instr);
492:C       10             writeln(output,'':strlen(instr)-1
493:C       10                             mod syscom^.crtinfo.width,'^');
494:C       10             printerror(-25,0);
495:C       10             goto 1;
496:C       10           end;
497:C        9           uc := uppercase(c);
498:C        9           if uc in parindex then
499:C       10             if parptr[uc] <> nil then write(streamfib^,parptr[uc]^)
500:C       11             else
501:C       11             begin
502:C       11               if not ciseoln then readln(f,s);
503:C       11               writeln(output,instr,s);
504:C       11               writeln(output,'':(strlen(instr)-1)
505:C       11                             mod syscom^.crtinfo.width,'^');
506:C       11               printerror(-25,0);
507:C       11               goto 1;
508:C       11             end
509:C       11           else           {write char as is}
510:C       10             write(streamfib^,c);
511:C        9         end
512:C        9         else if ciseoln then                {char is eoln}
513:C       10         begin
514:C       10           if not eof(f) then writeln(streamfib^);
515:C       10         end
516:S
517:C       10         else                                {normal char}
518:C       10           write(streamfib^,c);
519:S
520:C        7         testioresult;
521:C        7         needc := true;
522:S
523:C        7       until ciseoln or eof(f);
524:S
525:C        6     end; {line to process}
526:S
527:C        5   end; {while not eof(f)}
528:S
529:C        3   testioresult;
530:C        3   streamsyntax := true;
531:S
532:C        3   1: release(heap);
533:C        3 end;    {STREAMSYNTAX}
534:S
535:S
536:C        2 begin {STREAMOPEN}
537:C        2   reset (f,sfile,'shared');                          {OPEN THE STREAM FILE}
538:C        2   if ioresult = ord (inoerror) then                  {SUCCESSFUL OPEN}
539:C        3   begin                                    {OPEN THE SYSTEM STREAM FILE}
540:C        3     if scantitle(sfile,v,t,segs,fk) then keystream:=segs<>0
```



```
541:C        4                                    else keystream:=false;
542:C        3     if not keystream then sfile := '*STREAM';
543:C        3     if streamsyntax then                  {SYNTAX AND WRITE TO SYSTEM FILE}
544:C        4     begin                                 {AVOID HOGGING THE DISK}
545:C        4       if not keystream then close(streamfib^, 'CRUNCH');
546:C        4       reset(streamfib^, sfile, 'shared');
547:C        4       fibp(gfiles[0])^.am := streamdvr;       {INPUT    }
548:C        4       fibp(gfiles[2])^.am := streamdvr;       {KEYBOARD}
549:C        4     end
550:C        4     else close (streamfib^, 'PURGE');         {REMOVE SYSTEM STREAM FILE}
551:C        3     close (f);
552:C        3   end
553:C        3   else if report then writeln(output,'Can''t open file ',sfile);
554:C        2 end; {STREAMOPEN}
555:S
556:C        1 procedure homecursor;  begin write(homechar); end;
557:S
558:C        1 procedure clearscreen; begin write(clearscr); versionup := false; end;
559:S
560:C        1 procedure clearline;   begin write(cteol   ); end;
561:S
562:D   -82  1 procedure prompt (*PL: STRING80*);
563:C        2   begin  homecursor; clearline; write(output,pl)  end;
564:S
565:D        1 procedure zaptypeahead;
566:D    -4  2 var x: integer;
567:C        2 begin call (fibp(gfiles[2])^.am, fibp(gfiles[2]), clearunit, x, 0, 0);
568:C        2       reset(input); reset(gfiles[2]^ {KEYBOARD});
569:C        2 end;
570:S
571:D        1 function getchar(flushit: boolean): char;
572:D    -1  2 var ch: char;
573:C        2 begin
574:C        2   if flushit then zaptypeahead;
575:C        2   read(input,ch);
576:C        2   getchar := uppercase (ch);
577:C        2 end (*GETCHAR*) ;
578:S
579:D        1 procedure initfnames;
580:S
581:D        2 const sf = sysfilenames
582:D        2                [ 'ASSEMBLER','COMPILER','EDITOR','FILER','LIBRARIAN','LIBRARY'];
583:D        2       sysvolname = sysfilevols
584:D        2                [ 'ASM','CMP','ACCESS','ACCESS','ACCESS','SYSVOL']; { js 8/5/83 }
585:S
586:D    -2  2 var   f: sysfiles;
587:D    -6  2       find:  set of sysfiles;
588:D    -8  2       lunit: unitnum;
589:S
590:D        2   procedure findem(var volume: vid);
591:D    -2  3   var f: sysfiles;
592:D  -668  3       l: file of integer;
593:D  -750  3       ltitle: string80;
594:C        3   begin
595:C        3     for f := assembler to library do
596:C        4       if f in find then
597:C        5         begin
598:C        5           ltitle := volume+':'+sf[f];
599:C        5           reset(l, ltitle,'shared');
600:C        5           if ioresult = ord(inoerror) then
```

```
601:C        6          begin
602:C        6          filename[f] := ltitle;
603:C        6          find := find - [f];
604:C        6          close(l);
605:C        6          end;
606:C        5        end;
607:C        3    end;  {findem}
608:S
609:C        2 begin   { initfnames }
610:C        2    find := [assembler..library];
611:C        2    findem(syvid);
612:C        2    lunit := 1;
613:C        2    while (lunit <= maxunit) and (find <> []) do with unitable^[lunit] do
614:C        4    begin
615:C        4      call (dam, uvid, lunit, getvolumename);
616:C        4      if uisblkd and (uvid <> '') and (uvid <> syvid) then findem(uvid);
617:C        4      lunit := lunit+1
618:C        4    end;
619:C        2    for f := assembler to library do
620:C        3      if f in find then filename[f] := sysvolname[f]+':'+sf[f];
621:C        2    syslibrary := filename[library];
622:C        2 end (*INITFNAMES*) ;
623:S
624:D        1    procedure initworkfile;
625:D   -82  2    var ltitle: string80;
626:D  -748  2        workfile: file of integer;
627:C        2    begin
628:C        2      with userinfo^ do
629:C        3        begin                                          (*INITIALIZE WORK FILES*)
630:C        3          errnum := 0; errblk := 0; errsym := 0;
631:C        3          symfid := ''; codefid := ''; workfid := ''; errfid := '';
632:S
633:C        3          ltitle := '*WORK.TEXT';
634:C        3          reset(workfile,ltitle,'shared');
635:C        3          gotsym := ioresult = ord(inoerror);
636:C        3          if gotsym then symfid := ltitle;
637:C        3          close(workfile);
638:S
639:C        3          ltitle := '*WORK.CODE';
640:C        3          reset(workfile,ltitle,'shared');
641:C        3          gotcode := ioresult = ord(inoerror);
642:C        3          if gotcode then codefid := ltitle;
643:C        3          close(workfile);
644:C        3        end;
645:C        2    end (*INITWORKFILE*) ;
646:S
647:D        1 procedure updatesysunit;
648:C        2 begin
649:C        2    initdate;
650:C        2    initfnames;
651:C        2    initworkfile;
652:C        2 end;
653:S
654:D        1 procedure whatfiles;
655:S
656:D   -14  2 var e: string[12];
657:D   -16  2     f: sysfiles;
658:D   -18  2     update: boolean; c: char;
659:D   -22  2     i: integer;
660:S
```



```
661:D        2   procedure edit(f: sysfiles);
662:D  -122  3   var s: fid;
663:C        3   begin
664:C        3   fgotoxy(output, 12, 3+ord(f)); write(cteol); readln(s);
665:C        3   fixname(s, codefile);
666:C        3   if s <> '' then filename[f] := s;
667:C        3   fgotoxy(output, 12, 3+ord(f)); write(filename[f], cteol);
668:C        3   end;
669:S
670:D        2   procedure volname(sysvol: boolean);
671:D    -4  3   var i: integer;
672:D  -126  3       s: fid;
673:C        3   begin
674:C        3   fgotoxy(output, 19, 11-ord(sysvol)); write(cteol); readln(s);
675:C        3   zapspaces(s);
676:C        3   if s<>'' then
677:C        4     begin
678:C        4     if sysvol then doprefix(s, syvid, sysunit, true)
679:C        5               else doprefix(s, dkvid, i,       false);
680:C        4     if ioresult<>ord(inoerror) then
681:C        5       begin
682:C        5       getioerrmsg(s,ioresult); fgotoxy(output, 0, 13);
683:C        5       writeln(bellchar, s, cteol);
684:C        5       end
685:C        5     else if sysvol then begin updatesysunit; update := true; end;
686:C        4     end;
687:C        3   fgotoxy(output, 19, 11-ord(sysvol));
688:C        3   if sysvol then write(syvid) else write(dkvid);
689:C        3   write(':',cteol);
690:C        3   end;
691:D   -22  2
692:C        2 begin   { whatfiles}
693:C        2  filename[library] := syslibrary; update := true;
694:C        2  repeat
695:C        3    if update then
696:C        4      begin
697:C        4      page;
698:C        4      writeln; writeln; writeln;
699:C        4      for f := assembler to library do
700:C        5          begin
701:C        5          e := '';
702:C        5          strwrite(e, 1, i, f);
703:C        5          writeln(e, '':12-strlen(e), filename[f]);
704:C        5          end;
705:C        4      writeln;
706:C        4      writeln('* System  volume:  ', syvid,':');
707:C        4      writeln(': Default volume:  ', dkvid,':');
708:C        4      update := false;
709:C        4      end;
710:C        3    writeln(homechar,
711:C        3            'Assembler  Compiler  Editor  Filer  Librarian',, cteol);
712:C        3    write  (    'liBrary  System volume  Default volume   Quit ', cteol);
713:C        3    c := getchar(false);
714:C        3    fgotoxy(output, 0, 13); write(cteol);
715:C        3    case c of
716:C        4      'A':  edit(assembler);
717:C        4      'B':  edit(library);
718:C        4      'C':  edit(compiler);
719:C        4      'D':  volname(false);      (prefix;)
720:C        4      'E':  edit(editor);
```

```
 721:C        4      'F':  edit(filer);
 722:C        4      'L':  edit(librarian);
 723:C        4      'S':  volname(true);       {sysvol;}
 724:C        4      'Q':  ;
 725:C        4      otherwise write(bellchar);
 726:C        4    end;
 727:C        3  until c = 'Q';
 728:C        2  syslibrary := filename[library];
 729:C        2 end;    { whatfiles }
 730:S
 731:D        1 procedure ramdriver(fp: fibp; request: amrequesttype; anyvar buffer: window;
 732:D        2                                                    length, position: integer);
 733:C        2 begin
 734:C        2 with fp^, unitable^[funit] do
 735:C        3  case request of
 736:C        4    flush:        ;
 737:C        4    writebytes: if length > 0 then
 738:C        5      fastmove(addr(buffer), ipointer(byteoffset + position + fileid), length);
 739:C        4    readbytes:  if length > 0 then
 740:C        5      fastmove(ipointer(byteoffset + position + fileid), addr(buffer), length);
 741:C        4    otherwise ioresult := ord(ibadrequest);
 742:C        4    end;
 743:C        2 end;    { ramdriver }
 744:S
 745:D        1 function getvalue(low,high: integer; var value: integer;
 746:D        2                                                numsign, opt: boolean): boolean;
 747:D   -82  2 var s: string80;
 748:D   -86  2     i: integer;
 749:C        2 begin
 750:C        2   readln(s);
 751:C        2   if (strlen(s)=0) and opt then s := '0';
 752:C        2   strread(s,1,i,value);
 753:C        2   if ioresult<>ord(inoerror) then
 754:C        3    if i <= strlen(s) then
 755:C        4     if (s[i]='#') and numsign then
 756:C        5       begin
 757:C        5       strread(s,i+1,i,value);
 758:C        5       if ioresult<>0 then writeln(#7'integer required')
 759:C        6       end
 760:C        5     else writeln(#7'integer required');
 761:C        2   if ioresult=ord(inoerror) then
 762:C        3     if (value<low) or (value>high) then
 763:C        4       begin
 764:C        4       writeln(#7'value must be between ',
 765:C        4        low:1, ' and ', high:1);
 766:C        4       ioresult := ord(ibadformat);
 767:C        4       end;
 768:C        2   getvalue := ioresult=0;
 769:C        2 end;    { getvalue }
 770:S
 771:D        1 procedure makeramvol;
 772:S
 773:D        2 const zero = direntry[
 774:D        2         dfirstblk: 0,          dlastblk: 6,
 775:D        2         dfkind: untypedfile,
 776:D        2         dvid:   'RAM',         deovblk: 6,
 777:D        2         dnumfiles: 0,          dloadtime: 0,
 778:D        2         dlastboot:  daterec    [year: 0,
 779:D        2         day: 0,                month: 0]];
 780:S
```



```
 781:S
 782:D    -4  2 var  volsize:   integer;
 783:D    -8  2      untnumb:   integer;
 784:D   -12  2      entries:   integer;
 785:D   -16  2      membytes:  integer;
 786:D   -16  2      trick: record case integer of
 787:D   -16  2         2: (ip: ipointer);
 788:D   -16  2         3: (i:  integer);
 789:D   -20  2         end;
 790:D  -682  2      f: fib;
 791:D  -762  2      cat: catentry;
 792:S
 793:C        2 begin   { makeramvol }
 794:C        2  writeln('*** CREATING A MEMORY VOLUME ***');
 795:C        2  writeln;
 796:C        2  write('What unit number?  ');
 797:C        2  if getvalue(7,maxunit,untnumb,true,false) then
 798:C        3    begin
 799:C        3    write('How many 512 byte BLOCKS?  ');
 800:C        3    if getvalue(1,maxint,volsize,false,false) then
 801:C        4     begin
 802:C        4     write('How many entries in directory?  ');
 803:C        4     if getvalue(0,maxint,entries,false,true) then
 804:C        5       begin
 805:C        5       releaseuser;
 806:C        5       membytes := blocksize*volsize;
 807:C        5       if integer(eheap)+membytes > userstack then escape(-2);
 808:C        5       newbytes(trick.ip, membytes);
 809:C        5       unitable^[untnumb] := unitable^[0];  {handles most fields correctly}
 810:C        5       with unitable^[untnumb] do  {fill in the rest}
 811:C        6         begin
 812:C        6           fm := ramdriver;
 813:C        6           byteoffset := trick.i;
 814:C        6           uvid := 'RAM';
 815:C        6           offline := false;
 816:C        6           umaxbytes := volsize*fblksize;
 817:S
 818:C        6           with f, cat do
 819:C        7             begin
 820:C        7             fvid := ''; funit := untnumb; ftitle := '';
 821:C        7             fwindow := addr(cat);
 822:C        7             cname := 'RAM';
 823:C        7             cextral := entries;
 824:C        7             cpsize := umaxbytes;
 825:C        7             call (dam,  f, untnumb, makedirectory);
 826:C        7             end;
 827:C        6         end;
 828:S
 829:C        5       if ioresult = ord(inoerror) then
 830:C        6         begin
 831:C        6         markuser;
 832:C        6         writeln;
 833:C        6         writeln('#',untnumb:1, ':  (RAM:)  zeroed');
 834:C        6         end
 835:C        6       else printerror(-10, ioresult);
 836:C        5       end
 837:C        5     end;
 838:C        3   end
 839:C        3 end;    { makeramvol }
 840:S
```

```
 841:D        1 procedure newsysunit;
 842:D     -8 2 var newunit,i: integer;
 843:D   -130 2     name: fid;
 844:C        2 begin
 845:C        2   write('What new system unit number?  ');
 846:C        2   if getvalue(1, maxunit, newunit, true, false) then
 847:C        3     begin
 848:C        3     name := '#'; strwrite(name, 2, i, newunit:1, ':');
 849:C        3     doprefix(name, syvid, sysunit, true);
 850:C        3     if ioresult <> ord(inoerror) then printerror(-10,ioresult)
 851:C        4     else updatesysunit;
 852:C        3     end;
 853:C        2 end;    { newsysunit }
 854:S
 855:D        1 procedure osinit;
 856:D     -2 2 var  iu: 1..maxunit;
 857:D     -6 2      esccode: integer;
 858:C        2 begin (*OSINIT*)
 859:C        2   esccode := sysescapecode;
 860:C        2   locklevel := 0; actionspending := 0;
 861:C        2   zaptypeahead;
 862:C        2   if keystream then close(streamfib^)
 863:C        3                     else close(streamfib^,'PURGE') ;
 864:C        2   for iu := 1 to maxunit do  (* force directory cleanups on all vols *)
 865:C        3     unitable^[iu].umediavalid := false;
 866:C        2   sysescapecode := esccode;
 867:C        2 end (*OSINIT*) ;
 868:S
 869:D        1 procedure go_prog (debugging: boolean);
 870:D     -1 2 var  stopgoing: boolean;
 871:D     -6 2      lastioresult:integer;
 872:D     -8 2      esccode:shortint;
 873:D    -12 2      userheap: anyptr;
 874:D    -16 2      modptr: moddescptr;
 875:D    -17 2      done:   boolean;
 876:C        2 begin {GO_PROG}
 877:C        2   repeat
 878:C        3     clearscreen;
 879:C        3     writeln(output);
 880:C        3     modptr := entrypoint;
 881:C        3     repeat
 882:C        4       done := modptr^.lastmodule;
 883:C        4       if modptr^.startaddr<>0 then
 884:C        5         begin
 885:C        5         call(debugger,1,entrypoint^.startaddr,ord(debugging));
 886:C        5         mark(userheap);
 887:C        5         userprogram(modptr^.startaddr,userstack); (*** ALL PROGRAMS ARE ENTERED HERE ****)
 888:C        5         esccode := escapecode; lastioresult:=ioresult;
 889:C        5         release(userheap);
 890:C        5         call(debugger,2,0,0);
 891:C        5         stopgoing := true;
 892:C        5         openfiles;
 893:C        5         if esccode <> 0 then
 894:C        6           begin
 895:C        6             done := true;
 896:C        6             disableuserisrs;
 897:C        6             osinit;                              {shuts off stream files}
 898:C        6             if (esccode <> -1) and (esccode<>-20) then
 899:C        7               begin
 900:C        7               printerror(esccode,lastioresult);
```



```
 901:C        7               prompt('Restart with debugger ? ');
 902:C        7               stopgoing := getchar(false) <> 'Y';
 903:C        7               end;
 904:C        6             debugging := not stopgoing;
 905:C        6           end;
 906:C        5         end;
 907:C        4       modptr := modptr^.link;
 908:C        4     until done;
 909:C        3   until stopgoing;
 910:C        2 end;  (GO_PROG)
 911:S
 912:D        1 procedure loadandgo (var filetogo:fid; permanent, debugging: boolean);
 913:D        2 label 1;
 914:D  -146  2 var vol: vid; name: fid;  segs: integer; kind: filekind;
 915:D  -168  2     modp: moddescptr; upcname: tid;
 916:C        2 begin
 917:C        2   if not permanent then
 918:C        3    if scantitle(filetogo,vol,name,segs,kind) then
 919:C        4     begin
 920:C        4     if strlen(name)<=tidleng then upcname := name else upcname := '';
 921:C        4     upc(upcname);
 922:C        4     modp := sysdefs;
 923:C        4     while modp <> nil do with modp^ do
 924:C        6       begin
 925:C        6       if startaddr<>0 then
 926:C        7        if (name = progname) or (ucase and (upcname = progname)) then
 927:C        8         begin
 928:C        8         if entrypoint<>modp then releaseuser;
 929:C        8         entrypoint := modp;
 930:C        8         go_prog(debugging);
 931:C        8         goto 1;
 932:C        8         end;
 933:C        6       modp := link;
 934:C        6       end;
 935:C        4     end;
 936:C        2   load(filetogo, permanent);
 937:C        2   if permanent then markuser
 938:C        3   else if entrypoint <> nil then go_prog(debugging);
 939:C        2 1: end;  (LOADANDGO)
 940:S
 941:D        1 procedure initheap;
 942:D     -4 2 var   marker: anyptr;
 943:C        2 begin                                          (*BASIC FILE AND HEAP SETTUP*)
 944:C        2   new(userinfo);
 945:C        2   new(streamfib);
 946:C        2   mark(marker);
 947:C        2   if integer(marker) > userstack then escape(-2);
 948:C        2   markuser;
 949:C        2 end (*INITHEAP*) ;
 950:S
 951:D        1 procedure initunits;
 952:D        2 var
 953:D     -2 2   lunit: unitnum;
 954:D   -664 2   f: fib;
 955:C        2 begin
 956:C        2  f.fileid := 0;
 957:C        2  for lunit := 1 to maxunit do
 958:C        3    with unitable^[lunit] do
 959:C        4      begin
 960:C        4      offline := false;
```

```
 961:C        4      umediavalid := false;
 962:C        4      f.funit := lunit;
 963:C        4      call (tm, addr(f), clearunit, f, 0, 0);
 964:C        4      offline := uisblkd and (ioresult<>0);
 965:C        4      end;
 966:C        2 end; (*INITUNITS*)
 967:S
 968:S
 969:D        1 procedure command;
 970:D        2 type  prompttype=string[79];
 971:D        2 const
 972:D        2       prompt1=prompttype
 973:D        2 ['Command: Cmplr Edit File Init Libr Run Xcut Ver ?'];
 974:D        2       prompt2=prompttype
 975:D        2 ['Command: Asm Dbg Memv New Perm Stream User What ?'];
 976:S
 977:D        2       lprompt1=prompttype
 978:D        2 ['Command: Compiler Editor Filer Initialize Librarian Run eXecute Version ?'];
 979:D        2       lprompt2=prompttype
 980:D        2 ['Command: Assembler Debugger Memvol Newsysvol Permanent Stream User What ?'];
 981:S
 982:D   -1   2 var   skipping: boolean;
 983:D   -6   2       i : integer;
 984:D  -10   2       pl : ^prompttype;
 985:D  -11   2       plfirst : boolean;
 986:S
 987:D        2   procedure execute(permanent: boolean; debugging: boolean);
 988:D -122   3   var  title: fid;
 989:C        3   begin
 990:C        3     if permanent then
 991:C        4           prompt('Load what code file? ')
 992:C        4     else if debugging then
 993:C        5           prompt('Debug what file? ')
 994:C        5     else prompt('Execute what file? ');
 995:C        3     readln(input,title);
 996:C        3     fixname(title, codefile);
 997:C        3     if strlen(title) > 0 then
 998:C        4       begin
 999:C        4       if strlen(title) > (sizeof(fid)-6)
1000:C        5       then setstrlen(title, sizeof(fid)-6);
1001:C        4       loadandgo(title,permanent,debugging);
1002:C        4       end;
1003:C        3   end (*EXECUTE*) ;
1004:S
1005:D        2   procedure compileandedit;
1006:C        3   begin (COMPILEANDEDIT)
1007:C        3     with userinfo^ do
1008:C        4       begin
1009:C        4         errnum := 0; errblk := 0;
1010:C        4         loadandgo(filename[compiler],false,false);
1011:C        4         if entrypoint <> nil then if errnum <> 0 then
1012:C        6           loadandgo(filename[editor],false,false);
1013:C        4       end;
1014:C        3   end;  (COMPILEANDEDIT)
1015:S
1016:D        2   procedure runworkfile (debugging:boolean);
1017:D -122   3   var title: fid;
1018:C        3   begin with userinfo^ do
1019:C        4    if not (gotsym or gotcode) then execute(false,debugging)
1020:C        5    else
```



```
1021:C        5      begin
1022:C        5      if not gotcode then compileandedit;
1023:C        5      if gotcode then
1024:C        6        begin
1025:C        6        title:=codefid;
1026:C        6        loadandgo(title,false,debugging);
1027:C        6        end;
1028:C        5      end;
1029:C        3   end (*RUNWORKFILE*) ;
1030:S
1031:D        2   procedure stream;
1032:D        3   var
1033:D   -87   3     sfile: string80;  i: integer;  done: boolean;
1034:C        3   begin
1035:C        3     if chaining = streamchain then
1036:C        4       begin
1037:C        4       chaining := nochain;
1038:C        4       sfile := chainfile;
1039:C        4       end
1040:C        4     else
1041:C        4       begin
1042:C        4       prompt('Stream what file ? ');
1043:C        4       readln(input,sfile);
1044:C        4       end;
1045:C        3     if strlen(sfile) <= 70 then             {too long of name can crash system}
1046:C        4       fixname(sfile,textfile)               { since SFILE is of type STRING80 }
1047:C        4     else
1048:C        4       begin
1049:C        4         sfile := '';
1050:C        4         writeln(output,'Stream file name too long');
1051:C        4       end;
1052:C        3     if strlen (sfile) > 0 then                  {GOT VALID NAME}
1053:C        4       begin
1054:C        4         if keystream then close(streamfib^)
1055:C        5                      else close(streamfib^, 'PURGE');
1056:C        4         streamopen(sfile,true);                 {TRY TO OPEN THE STREAM FILE}
1057:C        4       end;                                      {EXIT IF NO NAME GIVEN}
1058:C        3   end; (*STREAM*)
1059:S
1060:C        2 begin (COMMAND)
1061:C        2   skipping := false;
1062:C        2   plfirst:=true;
1063:C        2   ci_cmd:=' ';
1064:C        2   call(cmdcharhook^);
1065:C        2   repeat
1066:C        3    if chaining=progchain then
1067:C        4      begin
1068:C        4      chaining := nochain;
1069:C        4      loadandgo(chainfile,false,false);
1070:C        4      end
1071:C        4    else if chaining=streamchain then stream
1072:C        5    else
1073:C        5      begin
1074:C        5      if skipping then
1075:C        6        begin
1076:C        6          while streaming and skipping do
1077:C        7            begin
1078:C        7              if eoln(input) then
1079:C        8                begin
1080:C        8                get(input);
```

```
  1081:C        8                if streaming then skipping := input^ = '*';
  1082:C        8                end
  1083:C        8              else get(input);
  1084:C        7            end;
  1085:C        6            for i := 1 to 80000 do {nothing};
  1086:C        6            skipping := false;
  1087:C        6          end;
  1088:S
  1089:C        5        if syscom^.crtinfo.width>=80 then
***WARNING: (line 1090): 'ADDR' of a constant may not be supported on other implementations
***WARNING: (line 1090): 'ADDR' of a constant may not be supported on other implementations
  1090:C        6          if plfirst then pl:=addr(lprompt1) else pl:=addr(lprompt2)
  1091:C        7        else
***WARNING: (line 1092): 'ADDR' of a constant may not be supported on other implementations
***WARNING: (line 1092): 'ADDR' of a constant may not be supported on other implementations
  1092:C        6          if plfirst then pl:=addr(prompt1) else pl:=addr(prompt2);
  1093:S
  1094:C        5        if ci_cmd<>chr(0) then prompt(pl^);
  1095:S
  1096:C        5        ci_idle:=true;
  1097:C        5        disptime;
  1098:C        5        if chaining=nochain then
  1099:C        6          begin
  1100:C        6            ci_cmd := getchar(false); call(cmdcharhook^);
  1101:C        6            if ci_cmd<>chr(0) then
  1102:C        7            begin clearscreen; writeln(output); end;
  1103:C        6          end;
  1104:S
  1105:C        5        if ci_cmd = chr(3) then ci_cmd := 'X';
  1106:S
  1107:C        5        setrunlight(ci_cmd);{set the run light to indicate command}
  1108:S
  1109:C        5        if not ((ci_cmd=' ') or (ci_cmd=chr(0))) then
  1110:C        6         case ci_cmd of
  1111:C        7    '?':  plfirst := not plfirst;
  1112:C        7    'A':  loadandgo(filename[assembler],false,false);
  1113:C        7    'C':  compileandedit;
  1114:C        7    'D':  runworkfile(true);
  1115:C        7    'E':  begin
  1116:C        7            userinfo^.errnum := 0;
  1117:C        7            userinfo^.errblk := 0;
  1118:C        7            loadandgo(filename[editor],false,false);
  1119:C        7          end;
  1120:C        7    'F':  loadandgo(filename[filer],false,false);
  1121:S
  1122:C        7    'I':  begin
  1123:C        7            lockup;
  1124:C        7            releaseuser;
  1125:C        7            lockfiles;
  1126:C        7            initunits;
  1127:C        7            openfiles;
  1128:C        7            lockdown;
  1129:C        7          end;
  1130:C        7    'L':  loadandgo(filename[librarian],false,false);
  1131:C        7    'M':  makeramvol;
  1132:C        7    'N':  newsysunit;
  1133:C        7    'P':  execute(true,false);
  1134:C        7    'R':  runworkfile(false);
  1135:C        7    'S':  stream;
  1136:C        7    'U':  if entrypoint <> nil then go_prog(false)
```



```
  1137:C        8          else execute(false, false);
  1138:C        7    'V':  dateset;
  1139:C        7    'W':  whatfiles;
  1140:C        7    'X':  execute(false,false);
  1141:C        7          otherwise
  1142:C        7            if streaming then
  1143:C        8              if ci_cmd = '*' then
  1144:C        9                begin
  1145:C        9                  skipping := true;
  1146:C        9                  write(output,'*');
  1147:C        9                end
  1148:C        9              else
  1149:C        9                begin
  1150:C        9                osinit;
  1151:C        9                if (ci_cmd > ' ') and (ord(ci_cmd)<127) then
  1152:C       10                  write(output,'',ci_cmd,'"')
  1153:C       10                else
  1154:C       10                  write(output,'Character #',ord(ci_cmd));
  1155:C        9                write(output,' is not a command.');
  1156:C        9                end;
  1157:S
  1158:C        7        end; {CASES}
  1159:C        5      end;
  1160:C        3    until false;
  1161:C        2  end;  {COMMAND}
  1162:S
  1163:S
  1164:D        1  procedure systemstartup;
  1165:D    -1  2  var done: boolean;
  1166:C        2  begin
  1167:C        2  initheap;                  (* point of final allocation of heap space *)
  1168:C        2  repeat
  1169:C        3  try
  1170:C        4   call(debugger,2,0,0);  { log in with debugger }
  1171:C        4   chaining := nochain;
  1172:C        4   versionup := false;
  1173:C        4   ci_idle:=false;
  1174:C        4   if cmdcharhook=nil then begin
  1175:C        5     new(cmdcharhook); markuser;
  1176:C        5     cmdcharhook^ := dummycmdchar;
  1177:C        5   end;
  1178:C        4   initworkfile;
  1179:C        4   initfnames;
  1180:C        4   streamopen('*AUTOSTART',false);  {open autostart stream file before dateset}
  1181:C        4   if ioresult<>ord(inoerror) then streamopen('*AUTOKEYS[*]',false);
  1182:C        4   initdate;
  1183:C        4   dateset;
  1184:S
  1185:C        4   repeat
  1186:C        5     try
  1187:C        6       command
  1188:C        6     recover
  1189:C        6       repeat
  1190:C        7         try
  1191:C        8           tioresult:=ioresult;                    {save it}
  1192:C        8           call(debugger,2,0,0);
  1193:C        8           osinit;                                 {shut off stream files}
  1194:C        8           if escapecode <> -1 then
  1195:C        9             begin
  1196:C        9             disableuserisrs;
```

```
 1197:C        9             clearscreen;  writeln(output);
 1198:C        9             printerror(escapecode,tioresult);
 1199:C        9             writeln(output,'Trapped by outer level of OS.');
 1200:C        9             end;
 1201:C        8           done := true;
 1202:C        8         recover done := false;
 1203:C        7       until done;
 1204:C        5  until false;
 1205:C        4 recover printerror(escapecode, ioresult);
 1206:C        3 until false;
 1207:C        2 end; (*systemstartup*)
 1208:S
 1209:C        1 end (*MODULE CI*);
 1210:S
 1211:D        1 import ci,asm;
 1212:S
 1213:C        1 begin
 1214:C        1   ci_switch;
 1215:C        1   systemstartup;
 1216:C        1 end. (*COMMAND INTERPRETER*)
 1217:S

No errors. 4 warnings.
             ***** Nonstandard language features enabled *****
```

Description

MINIT contains the Pascal portion of all the device-dependent drivers in MEDIAINIT. Having no real source of its own, this short file includes the files:

- MIDECS: Mediainit declarations.
- MMINIT: Internal minifloppy initialization drivers.
- HMINIT: Supported Amigo device initialization drivers.
- QMINIT: CS80 initialization drivers.
- XMINIT: Unsupported Amigo device initialization drivers.

Notes

Includes MMINIT, HMINIT, QMINIT, XMINIT and MIDECS.

```
Pascal [Rev 3.0M  6/ 4/84] MINIT.TEXT              26-Dec-84 22:58:49 Page 1   *** File name: MINIT ***

     1:D        0 $modcal$
     2:D        0 $debug off, range off, ovflcheck off$
     3:D        0 $stackcheck off, iocheck off$
     4:S
     5:D        0 $search 'IOLIB:KERNEL', 'INIT:DRVASM', 'INIT:DISCHPIB',
     6:D        0         'INIT:AMIGO', 'INIT:CS80'$
     7:S
     7:D        0 $include 'MIDECS'$;
     8:S
     9:S
    10:D        0 module midecs; {mediainit declarations}
    11:S
    12:D        1 import
    13:D        1   sysglobals;
    14:S
    15:D        1 export
    16:S
    17:D        1   type
    18:S
    19:D        1     interleave_data =
    20:D        1       record
    21:D        1         min: 0..255;    {minimum interleave factor}
    22:D        1         max: 0..255;    {maximum interleave factor}
    23:D        1         def: 0..255;    {default interleave factor}
    24:D        1       end;
    25:S
    26:D        1     physical_data = {(for LIF volume labels)}
    27:D        1       record
    28:D        1         ntps: integer;  {number of tracks per surface}
    29:D        1         nspm: integer;  {number of surfaces per media}
    30:D        1         nspt: integer;  {number of sectors per track}
    31:D        1       end;
    32:S
    33:S
    34:D        1   var
    35:D    -1  1     extended_features_mode: boolean;
    36:S
    37:D    -1  1   function yes(prompt: string255): boolean;
    38:S
    39:D    -1  1 implement  {midecs}
    40:S
    41:D  -256  1 function yes(prompt: string255): boolean;
    42:D  -256  2   var
    43:D  -257  2     answer: char;
    44:C        2   begin {yes}
    45:C        2     write(prompt, ' (Y/N) ');
    46:C        2     repeat
    47:C        3       read(keyboard, answer);
    48:C        3     until answer in ['Y', 'y', 'N', 'n'];
    49:C        2     writeln(answer);
    50:C        2     yes := (answer='Y') or (answer='y');
    51:C        2   end; {yes}
    52:S
    53:C        1 end   {midecs}
    54:S
    55:C        2 $include 'MIDECS'$;
    55:D        1 $include 'BMINIT'$;
    56:D        1 MODULE BMINIT;
    57:D        1 IMPORT SYSGLOBALS,MIDECS;
    58:D        1 EXPORT
```

```
Pascal [Rev 3.0M  6/ 4/84] BMINIT.TEXT             26-Dec-84 22:58:49 Page 2   *** File name: MINIT ***

    59:D        1   PROCEDURE BMINITIALIZE(UN:INTEGER);
    60:D        1   FUNCTION BPHYDATA(UN:INTEGER):PHYSICAL_DATA;
    61:D        1 IMPLEMENT
    62:D        1   PROCEDURE BMINITIALIZE(UN:INTEGER);
    63:D        2   VAR
    64:D    -4  2     VFILE  : FIBP;
    65:D    -8  2     HP   : ANYPTR;
    66:D   -12  2     I    : INTEGER;
    67:D  -268  2     SECTOR : PACKED ARRAY[0..255] OF CHAR;
    68:C        2   BEGIN
    69:C        2     MARK(HP);
    70:C        2     TRY
    71:C        3       NEW(VFILE);
    72:C        3       WITH VFILE^, UNITABLE^[UN]  DO
    73:C        4       BEGIN
    74:C        4         { INITIALIZE FIB FIELDS }
    75:C        4         FUNIT := UN;
    76:C        4         FPEOF := UMAXBYTES;
    77:C        4         FILEID := 0;
    78:C        4         { RESET THE UNIT }
    79:C        4         CALL(IM,VFILE,CLEARUNIT,SECTOR,0,0);
    80:C        4         IF IORESULT<>ORD(INOERROR) THEN ESCAPE(-10);
    81:S
    82:C        4         { WRITE BLANKS ALL OVER THE DEVICE }
    83:C        4         FOR I := 0 TO 255 DO SECTOR[I]:=' ';       { BLANK FILL }
    84:C        4         FOR I := 0 TO (UMAXBYTES DIV 256)-1 DO
    85:C        5         BEGIN
    86:C        5           CALL(IM,VFILE,WRITEBYTES,SECTOR,256,I*256);
    87:C        5           IF IORESULT<>ORD(INOERROR) THEN ESCAPE(-10);
    88:C        5         END;
    89:C        4       END;
    90:C        3       RELEASE(HP);
    91:C        3     RECOVER
    92:C        3       BEGIN RELEASE(HP); ESCAPE(ESCAPECODE); END;
    93:C        2   END;
    94:S
    95:D        1   FUNCTION BPHYDATA(UN:INTEGER):PHYSICAL_DATA;
    96:D        2   VAR
    97:D   -12  2     PDATA : PHYSICAL_DATA;
    98:C        2   BEGIN
    99:C        2     PDATA.NTPS := 1; PDATA.NSPM := 1;
   100:C        2     PDATA.NSPT := UNITABLE^[UN].UMAXBYTES DIV 256;
   101:C        2     BPHYDATA := PDATA;
   102:C        2   END;
   103:S
   104:C        1 END
   105:C        2 $include 'BMINIT'$;
   105:D        1 $include 'MMINIT'$;
```

```
106:D        1 $page$
107:S
108:D        1 module Mminit;
109:S
110:D        1 import
111:D        1   sysglobals, fs, asm, sysdevs, mini, midecs;
112:S
113:D        1 export
114:D        1   function Mintdata: interleave_data;
115:D        1   function Mphydata: physical_data;
116:D        1   procedure Mminitialize(interleave: shortint);
117:S
118:D        1 implement {Mminit}
119:S
120:S
121:D        1 function Mintdata: interleave_data;
122:D        2   const
123:D        2     intdata = interleave_data
124:D        2       [ min: 1,          {minimum interleave factor}
125:D        2         max: 15,         {maximum interleave factor}
126:D        2         def: 1 ];        {default interleave factor}
127:C        2   begin {Mintdata}
128:C        2     Mintdata := intdata;
129:C        2   end; {Mintdata}
130:S
131:S
132:D        1 function Mphydata: physical_data;
133:D        2   const
134:D        2     phydata = physical_data
135:D        2       [ ntps: (2*35-4) div 2,   {number of tracks per surface}
136:D        2         nspm: 2,                {number of surfaces per media}
137:D        2         nspt: 16 ];             {number of sectors per track}
138:C        2   begin {Mphydata}
139:C        2     Mphydata := phydata;
140:C        2   end; {Mphydata}
```



```
141:D        1 $page$
142:S
143:D        1 procedure Mminitialize(interleave: shortint);
144:S
145:D        2   { NOTE:  the BOOTROM global that selects drive 0 or 1 must already be setup! }
146:S
147:D        2   var
148:D    -8  2     x,y: integer;
149:D   -12  2     crtptr: anyptr;
150:D   -14  2     saved_escapecode: shortint;
151:S
152:C        2   begin {Mminitialize}
153:S
154:C        2     call(maskopshook,fhimask,0);        {enable kbd fasthandshake}
155:S
156:C        2     writeln;
157:C        2     fgetxy(output,x,y);
158:C        2     with syscom^.crtinfo do
159:C        3       crtptr := anyptr(crtmemaddr+(y-1)*2*width);
160:S
161:C        2     try
162:C        3       floyinit(crtptr,interleave);
163:C        3       call(maskopshook,0,fhimask);    {disable kbd fasthandshake}
164:S
165:C        3     recover
166:C        3       begin
167:C        3         call(maskopshook,0,fhimask);    {disable kbd fasthandshake}
168:C        3         if iorval<>inoerror then ioresc(iorval);
169:C        3         saved_escapecode := escapecode;
170:C        3         try
171:C        4           f_pwr_on;
172:C        4         recover
173:C        4           {ignore any errors};
174:C        3         escape(saved_escapecode);
175:C        3       end; {recover}
176:S
177:C        2   end; {Mminitialize}
178:S
179:C        1 end {Mminit}
180:S
181:C        2 $include 'MMINIT'$;
181:D        1 $include 'HMINIT'$;
```

```
  182:D        1 $page$
  183:S
  184:D        1 module IRamigo; {amigo initialization routines}
  185:S
  186:D        1 import
  187:D        1   sysglobals, bkgnd, discHPIB, CSamigo;
  188:S
  189:D        1 export
  190:D        1   procedure verify          (uep: uep_type; nsectors: unsigned16);
  191:D        1   procedure init_with_d_bits(uep: uep_type);
  192:D        1   procedure format          (uep: uep_type; intlve, d_byte: byte; override: boolean);
  193:S
  194:D        1 implement {IRamigo}
  195:S
  196:D        1 procedure verify(uep: uep_type; nsectors: unsigned16);
  197:D        2   var
  198:D        2     verify_cmd_buf:
  199:D        2       packed record
  200:D        2         ftcb: ftcb_type;
  201:D        2         sector_count: unsigned16;
  202:D    -4  2       end;
  203:C        2   begin {verify}
  204:C        2     verify_cmd_buf.sector_count := nsectors;
  205:C        2     issue_cmd(uep, verify_cmd, verify_cmd_buf.ftcb);
  206:C        2   end; {verify}
  207:S
  208:D        1 procedure init_with_d_bits(uep: uep_type);
  209:D        2   var
  210:D    -2  2     init_cmd_buf: ftcb_type;
  211:D    -4  2     dummy_byte: packed record  b: byte;  end;
  212:D    -4  2   const
  213:D    -4  2     receive_data_sec = 0;
  214:C        2   begin {init_with_d_bits}
  215:C        2     issue_cmd(uep, init_d_cmd, init_cmd_buf);
  216:C        2     HPIBwait_for_ppol(uep);
  217:C        2     dummy_byte.b := 0;
  218:C        2     HPIBshort_msge_out(uep, receive_data_sec, addr(dummy_byte), sizeof(dummy_byte));
  219:C        2   end; {init_with_d_bits}
  220:S
  221:D        1 procedure format(uep: uep_type; intlve, d_byte: byte; override: boolean);
  222:D        2   var
  223:D        2     format_cmd_buf:
  224:D        2       packed record
  225:D        2         ftcb: ftcb_type;
  226:D        2         f: boolean;       { override old format bit }
  227:D        2         ttttttt: 0..127; { wanted format type      }
  228:D        2         int: byte;       { interleave factor       }
  229:D        2         db: byte;        { data byte               }
  230:D    -6  2       end;
  231:C        2   begin {format}
  232:C        2     with format_cmd_buf do
  233:C        3       begin
  234:C        3         f := override;
  235:C        3         ttttttt := 2;   {HP format always}
  236:C        3         int := intlve;
  237:C        3         db := d_byte
  238:C        3       end; {with}
  239:C        2     issue_cmd(uep, format_cmd, format_cmd_buf.ftcb);
  240:C        2   end; {format}
  241:S
```



```
  242:C        1 end; {IRamigo}
  243:S
  244:S
  245:D        1 module Hminit;
  246:S
  247:D        1 import
  248:D        1   sysglobals, bkgnd, discHPIB, CSamigo, IRamigo, midecs;
  249:S
  250:D        1 export
  251:D        1   function Hintdata: interleave_data;
  252:D        1   function Nintdata(uep: uep_type): interleave_data;
  253:D        1   function Hphydata(uep: uep_type): physical_data;
  254:D        1   procedure Hminitialize(uep: uep_type; intlve: shortint);
  255:S
  256:D        1 implement {Hminit}
  257:S
  258:S
  259:D        1 function Hintdata: interleave_data;
  260:D        2   const
  261:D        2     intdata = interleave_data
  262:D        2       [ min: 1,          {minimum interleave factor}
  263:D        2         max: 29,         {maximum interleave factor}
  264:D        2         def: 2 ];        {default interleave factor}
  265:C        2   begin {Hintdata}
  266:C        2     Hintdata := intdata;
  267:C        2   end; {Hintdata}
  268:S
  269:S
  270:D        1 function Nintdata(uep: uep_type): interleave_data;
  271:D        2   const
  272:D        2     intdata = interleave_data
  273:D        2       [ min: 1,          {minimum interleave factor}
  274:D        2         max: 15,         {maximum interleave factor}
  275:D        2         def: 3 ];        {default interleave factor}
  276:C        2   begin {Nintdata}
  277:C        2     Nintdata := intdata;
  278:C        2     if uep^.devid<>0 then  {this is actually a Sparrow...}
  279:C        3       Nintdata.def := 2;   {which performs best at this interleave}
  280:C        2   end; {Nintdata}
  281:S
  282:S
  283:D        1 function Hphydata(uep: uep_type): physical_data;
  284:D        2   var
  285:D    -6  2     map: map_type;
  286:D   -18  2     phydata: physical_data;
  287:C        2   begin {Hphydata}
  288:C        2     get_map(uep, map);
  289:C        2     with map, phydata do
  290:C        3       begin
  291:C        3         nspm := trk_per_cyl;    {number of surfaces per medium}
  292:C        3         ntps := cyl_per_med;    {number of tracks/surface}
  293:C        3         nspt := sec_per_trk;    {number of sectors per track}
  294:C        3       end; {with}
  295:C        2     Hphydata := phydata;
  296:C        2   end; {Hphydata}
```

```
297:D        1 $page$
298:S
299:D        1 procedure Hminitialize(uep: uep_type; intlve: shortint);
300:S
301:D        2   var
302:D    -4  2     status_bytes: status_type;
303:D   -10  2     map: map_type;
304:D   -12  2     tracks_per_medium: shortint;
305:S
306:D        2   procedure confirm_preeoi;
307:C        3     begin {confirm_preeoi}
308:C        3       with bip_type(uep^.dvrtemp)^ do
309:C        4         if (escapecode=-10) and (iores=zbadhardware)
310:C        5           then iores := inoerror
311:C        5           else escape(escapecode);
312:C        3     end; {confirm_preeoi}
313:D   -12  2
314:D        2   procedure check_status;
315:C        3     begin {check_status}
316:C        3       status(uep, status_bytes);
317:C        3       case status_bytes.s1 of
318:C        4         {"acceptable" errors}
319:C        4           uncorrectable_data_error,
320:C        4           head_sector_compare_error,
321:C        4           access_not_ready_during_data_operation:
322:C        4             {acceptable; do not escape};
323:C        4         {error requiring status 2 check}
324:C        4           drive_attention:
325:C        4             if not status_bytes.c then ioresc_bkgnd(uep, zinitfail);
326:C        4         {other errors}
327:C        4           otherwise ioresc_bkgnd(uep, zinitfail)
328:C        4       end; {case}
329:C        3     end; {check_status}
330:S
331:S
332:D        2   procedure init_9895_913X_Chinook;
333:D        3     const
334:D        3       testpasses = 4;                {number of test passes       }
335:D        3       max_bt = 4;                    {maximum number of bad tracks}
336:D        3     type
337:D        3       test_pattern_type = packed array[1..testpasses] of byte;
338:D        3     const
339:D        3       test_pattern = test_pattern_type
340:D        3         [ 198,    {$C6 or 1100011011000110}
341:D        3           99,     {$63 or 0110001101100011}
342:D        3           219,    {$DB or 1101101111011011}
343:D        3           136 ];  {$88 or 1000100010001000}
344:D        3     var
345:D    -2  3       bt_count: shortint;
346:D    -4  3       test_pass: shortint;
347:D    -6  3       verify_starting_track: shortint;
348:D    -7  3       verify_pass_complete: boolean;
349:D    -8  3       bt_already_logged: boolean;
350:D   -10  3       bad_track: shortint;
351:D   -12  3       bt_index: shortint;
352:D   -20  3       bt_table: array[1..max_bt] of shortint;
```



```
353:D   -20  3 $page$
354:S
355:C        3     begin {init_9895_913X_Chinook}
356:C        3       bt_count := 0;
357:C        3       for test_pass := 1 to testpasses do
358:C        4         begin
359:C        4           format(uep, 1, test_pattern[test_pass], true);
360:C        4           HPIBwait_for_ppol(uep);
361:C        4           if dsj(uep)<>0 then ioresc_bkgnd(uep, zinitfail);
362:C        4           verify_starting_track := 0;
363:C        4           repeat
364:C        5             seek(uep, verify_starting_track*map.sec_per_trk);
365:C        5             HPIBwait_for_ppol(uep);
366:C        5             verify(uep, (tracks_per_medium+max_bt-verify_starting_track)*map.sec_per_trk);
367:C        5             HPIBwait_for_ppol(uep);
368:C        5             if dsj(uep)=0 then
369:C        6               verify_pass_complete := true
370:C        6             else
371:C        6               begin
372:C        6                 check_status;
373:C        6                 bad_track := logical_addr(uep) div map.sec_per_trk;
374:C        6                 bt_already_logged := false;  {initial assumption}
375:C        6                 for bt_index := 1 to bt_count do
376:C        7                   if bt_table[bt_index]=bad_track then bt_already_logged := true;
377:C        6                 if not bt_already_logged then  {log it}
378:C        7                   begin
379:C        7                     if bt_count>=max_bt then ioresc_bkgnd(uep, zinitfail);
380:C        7                     bt_count := bt_count+1;
381:C        7                     bt_table[bt_count] := bad_track;
382:C        7                   end;
383:C        6                 verify_starting_track := bad_track+1;
384:C        6                 verify_pass_complete := verify_starting_track>=tracks_per_medium+max_bt;
385:C        6               end {else}
386:C        6           until verify_pass_complete;
387:C        4         end; {for}
388:S
389:C        3       for bt_index := 1 to bt_count do {set all d-bits on the bad tracks}
390:C        4         begin
391:C        4           seek(uep, bt_table[bt_index]*map.sec_per_trk);
392:C        4           HPIBwait_for_ppol(uep);
393:C        4           init_with_d_bits(uep);
394:C        4           HPIBwait_for_ppol(uep);
395:C        4           if dsj(uep)<>0 then ioresc_bkgnd(uep, zinitfail);
396:C        4         end;
397:S
398:C        3       format(uep, intlve, 0, false);
399:C        3       HPIBwait_for_ppol(uep);
400:C        3       if dsj(uep)<>0 then ioresc_bkgnd(uep, zinitfail);
401:C        3       seek(uep, 0);
402:C        3       HPIBwait_for_ppol(uep);
403:C        3       verify(uep, tracks_per_medium*map.sec_per_trk);
404:C        3       HPIBwait_for_ppol(uep);
405:C        3       if dsj(uep)<>0 then ioresc_bkgnd(uep, zinitfail);
406:C        3     end; {init_9895_913X_Chinook}
```

```
407:D   -12  2 $page$
408:S
409:D        2   procedure init_913X_ABC;
410:S
411:D        3     var
412:D     -2 3       verify_pass: shortint;
413:D     -4 3       csvp: shortint;  {consecutive successful verify passes}
414:D     -6 3       verify_starting_track: shortint;
415:D     -7 3       verify_pass_complete: boolean;
416:S
417:D     -7 3     const
418:D     -7 3       min_vp = 5;        {minimum # of verify passes}
419:D     -7 3       max_vp = 15;       {maxmimum # of verify passes}
420:D     -7 3       min_csvp = 2;      {minimum # of consecutive successful verify passes}
421:S
422:C        3     begin {init_913X_ABC}
423:S
424:C        3       format(uep, intlve, 0, true);
425:C        3       HPIBwait_for_ppol(uep);
426:C        3       if dsj(uep)<>0 then ioresc_bkgnd(uep, zinitfail);
427:S
428:C        3       verify_pass := 0;
429:C        3       csvp := 0;
430:S
431:C        3       repeat
432:S
433:C        4         verify_pass := verify_pass+1;
434:C        4         verify_starting_track := 0;
435:S
436:C        4         repeat
437:C        5           seek(uep, verify_starting_track*map.sec_per_trk);
438:C        5           HPIBwait_for_ppol(uep);
439:C        5           verify(uep, (tracks_per_medium-verify_starting_track)*map.sec_per_trk);
440:C        5           HPIBwait_for_ppol(uep);
441:C        5           if dsj(uep)=0 then
442:C        6             verify_pass_complete := true
443:C        6           else
444:C        6             begin
445:C        6               check_status;
446:C        6               verify_starting_track := logical_addr(uep) div map.sec_per_trk+1;
447:C        6               init_with_d_bits(uep);
448:C        6               HPIBwait_for_ppol(uep);
449:C        6               if dsj(uep)<>0 then ioresc_bkgnd(uep, zinitfail);
450:C        6               verify_pass_complete := verify_starting_track>=tracks_per_medium;
451:C        6             end {else}
452:C        6         until verify_pass_complete;
453:S
454:C        4         if verify_starting_track=0 {i.e., successful verify pass}
455:C        5           then csvp := csvp+1
456:C        5           else csvp := 0;
457:S
458:C        4       until ((verify_pass>=min_vp) and (csvp>=min_csvp)) or (verify_pass>=max_vp);
459:S
460:C        3       if csvp<min_csvp then ioresc_bkgnd(uep, zinitfail);
461:S
462:C        3     end; {init_913X_ABC}
```



```
463:D   -12  2 $page$
464:S
465:C        2 begin {Hminitialize}
466:C        2   try
467:C        3     allocate_bkgnd_info(uep);
468:S
469:C        3     try
470:C        4       status(uep, status_bytes);
471:C        4     recover
472:C        4       begin
473:C        4         confirm_preeoi;
474:C        4         if dsj(uep)=2 then ioresc_bkgnd(uep, zmediumchanged)
475:C        5         else ioresc_bkgnd(uep, zbadhardware);
476:C        4       end; {recover}
477:S
478:C        3     with status_bytes do
479:C        4       begin
480:C        4         if f then ioresc_bkgnd(uep, zmediumchanged);
481:C        4         if ss<>0 then ioresc_bkgnd(uep, znomedium);
482:C        4         if w then ioresc_bkgnd(uep, zprotected);
483:C        4       end; {with}
484:S
485:C        3     get_map(uep, map);
486:C        3     with map do
487:C        4       tracks_per_medium := trk_per_cyl*cyl_per_med;
488:S
489:C        3     case device(uep) of
490:C        4       HP9895, HP8290X:          init_9895_913X_Chinook;
491:C        4       HP913X_A..HP913X_C:       init_913X_ABC;
492:C        4       otherwise                 ioresc_bkgnd(uep, zcatchall);
493:C        4     end; {case}
494:S
495:C        3     deallocate_bkgnd_info(uep);
496:S
497:C        3   recover
498:C        3     begin
499:C        3       abort_bkgnd_process(uep);
500:C        3       ioresult := uep^.dvrtemp;
501:C        3       uep^.dvrtemp := ord(inoerror);  {report the error only once}
502:C        3       escape(-10);
503:C        3     end; {recover}
504:S
505:C        2 end; {Hminitialize}
506:S
507:C        1 end {Hminit}
508:S
509:S
510:C        2 $include 'HMINIT'$;
510:D        1 $include 'QMINIT'$;
```

```
511:D        1 $page$
512:S
513:D        1 module CS80ir; {Command Set '80 initialization routines}
514:S
515:D        1 import
516:D        1   sysglobals, bkgnd, discHPIB, CS80;
517:S
518:D        1 export
519:D        1   type
520:D        1     tva_type = {three-vector address (6 bytes)}
521:D        1       packed record
522:D        1         cyln: unsgn24;
523:D        1         head: unsgn8;
524:D        1         sect: signed16;
525:D        1       end;
526:D        1
527:D        1     de_log_type = {the two distinct types of data error logs}
528:D        1       (ert, runtime);
529:D        1
530:D        1     log_entry_type = {data error log entry}
531:D        1       packed record
532:D        1         physical_cyln:  signed16;
533:D        1         physical_head:  unsgn8;
534:D        1         physical_sect:  unsgn8;
535:D        1         logical_cyln:   signed16;
536:D        1         logical_head:   unsgn8;
537:D        1         logical_sect:   unsgn8;
538:D        1         error_byte:     unsgn8;
539:D        1         occurrences:    unsgn8;
540:D        1       end;
541:D        1
542:D        1     log_info_type = {info returned by the READ_DATA_ERROR_LOGS command}
543:D        1       packed record
544:D        1         my_special_pad: unsgn8;   {internal allignment only; not sent by the device!!!}
545:D        1         log_entries:    unsgn8;
546:D        1         other_info:     array[1..5+2+1] of char;
547:D        1         entry:          array[1..106] of log_entry_type;
548:D        1       end;
549:D        1
550:D        1     ert_area_type =
551:D        1       (sector_ert, track_ert, cylinder_ert, surface_ert, volume_ert);
552:D        1
553:S            {
554:S              NOTE: the following functions each perform a COMPLETE transaction. They:
555:S                    . issue a (device or transparent) command          (Command message)
556:S                    . transfer data if applicable                      (Execution message)
557:S                    . return the resulting QSTAT                       (Reporting message)
558:D        1   }
559:D        1   function formatting_option  (uep: uep_type; option: unsgn8): unsgn8;
560:D        1   function initiate_diagnostic(uep: uep_type; unit: unsgn4; loops: signed16; section: unsgn8): unsgn8;

561:D        1   function initialize_media   (uep: uep_type; options, interleave: unsgn8): unsgn8;
562:D        1   function preset_drive       (uep: uep_type): unsgn8;
563:D        1   function clear_logs         (uep: uep_type; logcode: unsgn8): unsgn8;
564:D        1   function set_3V_address     (uep: uep_type; cyln: unsgn24; head: unsgn8; sect: signed16): unsgn8;
565:D        1   function data_error_log     (uep: uep_type; de_log: de_log_type; head: unsgn8;
566:D        2                                 var log_bytes: log_info_type): unsgn8;
567:D        1   function pattern_ert        (uep: uep_type; ert_area: ert_area_type; loops: unsgn8;
568:D        2                                 var ert_message: boolean; var log_entry: log_entry_type): unsgn8;
569:D        1   function spare_block        (uep: uep_type; sparingmode: unsgn8): unsgn8;
```



```
570:D        1 $page$
571:S
572:D        1 implement {CS80ir}
573:S
574:S
575:D        1 type
576:D        1   setunitvol_type = {SET_UNIT/SET_VOLUME command pair}
577:D        1     packed record
578:D        1       setunit: CMD_type;
579:D        1       setvol: CMD_type;
580:D        1     end;
581:S
582:S
583:D        1 function suv_CMD_pair(uep: uep_type): setunitvol_type;
584:C        2   begin {suv_CMD_pair}
585:C        2     suv_CMD_pair.setunit := CMD_type(signed16(CMDset_unit_0)+uep^.du);
586:C        2     suv_CMD_pair.setvol  := CMD_type(signed16(CMDset_vol_0)+uep^.dv);
587:C        2   end; {suv_CMD_pair}
588:S
589:S
590:D        1 function formatting_option(uep: uep_type; option: unsgn8): unsgn8;
591:S            {
592:S              issue the following command sequence:
593:S                . SET_UNIT
594:S                . SET_VOLUME
595:S                . SET_FORMAT_OPTION (Subset/80 only)
596:S              issue the option byte in an execution message
597:S              return the QSTAT byte
598:D        2   }
599:D        2   var
600:D        2     fo: {the 5 bytes in the command message}
601:D        2       packed record
602:D        2         setunitvol: setunitvol_type;
603:D        2         initutil: CMD_type;
604:D        2         setformatoption: unsgn8;
605:D        2         parameter: unsgn8;
606:D    -6  2       end;
607:D    -6  2     em: {the single byte execution message}
608:D    -6  2       packed record
609:D    -6  2         optionbyte: unsgn8;
610:D    -8  2       end;
611:C        2   begin {formatting_option}
612:C        2     fo.setunitvol      := suv_CMD_pair(uep);
613:C        2     fo.initutil        := CMDinit_util_REM;
614:C        2     fo.setformatoption := 243;
615:C        2     fo.parameter       := 95;
616:C        2     HPIBshort_msge_out(uep, command_sec, addr(fo), sizeof(fo));
617:C        2     HPIBwait_for_ppol(uep);
618:C        2     em.optionbyte := option;
619:C        2     HPIBshort_msge_out(uep, execution_sec, addr(em), sizeof(em));
620:C        2     HPIBwait_for_ppol(uep);
621:C        2     formatting_option := qstat(uep);
622:C        2   end; {formatting_option}
```

```
623:D        1 $page$
624:S
625:D        1 function initiate_diagnostic(uep: uep_type; unit: unsgn4; loops: signed16; section: unsgn8): unsgn8;
626:S            {
627:S              set specified unit & issue the INITIATE_DIAGNOSTIC command
628:S              return the QSTAT byte
629:D        2   }
630:D        2   var
631:D        2     id: {the 5 bytes in the initiate diagnostic command message}
632:D        2       packed record
633:D        2         setunit: CMD_type;
634:D        2         initdiag: CMD_type;
635:D        2         loops: signed16;
636:D        2         section: unsgn8;
637:D    -6  2       end;
638:C        2   begin {initiate_diagnostic}
639:C        2     id.setunit  := CMD_type(signed16(CMDset_unit_0)+unit);
640:C        2     id.initdiag := CMDinit_diagnostic;
641:C        2     id.loops    := loops;
642:C        2     id.section  := section;
643:C        2     HPIBshort_msge_out(uep, command_sec, addr(id), sizeof(id));
644:C        2     HPIBwait_for_ppol(uep);
645:C        2     initiate_diagnostic := qstat(uep);
646:C        2   end; {initiate_diagnostic}
647:S
648:S
649:D        1 function initialize_media(uep: uep_type; options, interleave: unsgn8): unsgn8;
650:S            {
651:S              issue the following command sequence:
652:S                . SET_UNIT
653:S                . SET_VOLUME
654:S                . INITIALIZE_MEDIA
655:S              return the QSTAT byte
656:D        2   }
657:D        2   var
658:D        2     im: {the 5 bytes in the command message}
659:D        2       packed record
660:D        2         setunitvol: setunitvol_type;
661:D        2         initmedia: CMD_type;
662:D        2         options: unsgn8;
663:D        2         interleave: unsgn8;
664:D    -6  2       end;
665:C        2   begin {initialize_media}
666:C        2     im.setunitvol := suv_CMD_pair(uep);
667:C        2     im.initmedia  := CMDinit_media;
668:C        2     im.options    := options;
669:C        2     im.interleave := interleave;
670:C        2     HPIBshort_msge_out(uep, command_sec, addr(im), sizeof(im));
671:C        2     HPIBwait_for_ppol(uep);
672:C        2     initialize_media := qstat(uep);
673:C        2   end; {initialize_media}
```



```
674:D        1 $page$
675:S
676:D        1 function preset_drive(uep: uep_type): unsgn8;
677:S            {
678:S              issue the following command sequence:
679:S                . SET_UNIT
680:S                . SET_VOLUME
681:S                . PRESET_DRIVE
682:S              return the QSTAT byte
683:D        2   }
684:D        2   var
685:D        2     pd: {the 4 bytes in the command message}
686:D        2       packed record
687:D        2         setunitvol: setunitvol_type;
688:D        2         initutil: CMD_type;
689:D        2         presetdrive: unsgn8;
690:D    -4  2       end;
691:C        2   begin {preset_drive}
692:C        2     pd.setunitvol  := suv_CMD_pair(uep);
693:C        2     pd.initutil    := CMDinit_util_NEM;
694:C        2     pd.presetdrive := 206;
695:C        2     HPIBshort_msge_out(uep, command_sec, addr(pd), sizeof(pd));
696:C        2     HPIBwait_for_ppol(uep);
697:C        2     preset_drive := qstat(uep);
698:C        2   end; {preset_drive}
699:S
700:S
701:D        1 function clear_logs(uep: uep_type; logcode: unsgn8): unsgn8;
702:S            {
703:S              issue the following command sequence:
704:S                . SET_UNIT
705:S                . SET_VOLUME
706:S                . CLEAR_LOGS
707:S              return the QSTAT byte
708:D        2   }
709:D        2   var
710:D        2     cl: {the 5 bytes in the command message}
711:D        2       packed record
712:D        2         setunitvol: setunitvol_type;
713:D        2         initutil: CMD_type;
714:D        2         clearlogs: unsgn8;
715:D        2         logcode: unsgn8;
716:D    -6  2       end;
717:C        2   begin {clear_logs}
718:C        2     cl.setunitvol := suv_CMD_pair(uep);
719:C        2     cl.initutil   := CMDinit_util_NEM;
720:C        2     cl.clearlogs  := 205;
721:C        2     cl.logcode    := logcode;
722:C        2     HPIBshort_msge_out(uep, command_sec, addr(cl), sizeof(cl));
723:C        2     HPIBwait_for_ppol(uep);
724:C        2     clear_logs := qstat(uep);
725:C        2   end; {clear_logs}
```

```
   726:D        1 $page$
   727:S
   728:D        1 function data_error_log(uep: uep_type; de_log: de_log_type; head: unsgn8;
   729:D        2                         var log_bytes: log_info_type): unsgn8;
   730:S            {
   731:S              issue the appropriate READ_DATA_ERROR_LOG command
   732:S              place the returned data error log info in the passed variable 'log_bytes'
   733:S              return the QSTAT byte
   734:D        2   }
   735:D        2   type
   736:D        2     read_de_log_micro_op_type =
   737:D        2       array[de_log_type] of unsgn8;
   738:D        2   const
   739:D        2     read_de_log_micro_op = read_de_log_micro_op_type
   740:D        2       [ {ert} 198, {runtime} 197 ];
   741:D        2   var
   742:D        2     del: {the 5 bytes in the command message}
   743:D        2       packed record
   744:D        2         setunitvol: setunitvol_type;
   745:D        2         initutil: CMD_type;
   746:D        2         readlogs: unsgn8;
   747:D        2         head: unsgn8;
   748:D    -6  2       end;
   749:C        2   begin {data_error_log}
   750:C        2     del.setunitvol := suv_CMD_pair(uep);
   751:C        2     del.initutil   := CMDinit_util_SEM;
   752:C        2     del.readlogs   := read_de_log_micro_op[de_log];
   753:C        2     del.head       := head;
   754:C        2     HPIBshort_msge_out(uep, command_sec, addr(del), sizeof(del));
   755:C        2     HPIBwait_for_ppol(uep);
   756:C        2     try
   757:C        3       HPIBshort_msge_in(uep, execution_sec, addr(log_bytes, 1), sizeof(log_bytes)-1);
   758:C        3     recover
   759:C        3       with bip_type(uep^.dvrtemp)^ do  {confirm the "premature" eoi}
   760:C        4         begin
   761:C        4           if (escapecode<>-10) or (iores<>zbadhardware) then
   762:C        5             escape(escapecode);
   763:C        4           iores := inoerror;
   764:C        4         end; {recover}
   765:C        2     HPIBwait_for_ppol(uep);
   766:C        2     data_error_log := qstat(uep);
   767:C        2   end; {data_error_log}
```



```
   768:D        1 $page$
   769:S
   770:D        1 function pattern_ert(uep: uep_type; ert_area: ert_area_type; loops: unsgn8;
   771:D        2                       var ert_message: boolean; var log_entry: log_entry_type): unsgn8;
   772:S            {
   773:S              issue the appropriate ERROR_RATE_TEST command
   774:S              if specified, request an execution message and set var boolean accordingly
   775:S              return the QSTAT byte
   776:D        2   }
   777:D        2   var
   778:D        2     ert: {the 9 bytes in the command message}
   779:D        2       packed record
   780:D        2         setunitvol: setunitvol_type;
   781:D        2         initutil: CMD_type;
   782:D        2         patternert: unsgn8;
   783:D        2         loops: unsgn8;
   784:D        2         offset: signed8;
   785:D        2         report: unsgn8;
   786:D        2         testarea: unsgn8;
   787:D        2         datasource: unsgn8;
   788:D   -10  2       end;
   789:C        2   begin {pattern_ert}
   790:C        2     ert.setunitvol := suv_CMD_pair(uep);
   791:C        2     if ert_message
   792:C        3       then ert.initutil := CMDinit_util_SEM     {send execution message}
   793:C        3       else ert.initutil := CMDinit_util_NEM;    {no execution message}
   794:C        2     ert.patternert      := 200;                 {pattern ert micro opcode}
   795:C        2     ert.loops           := loops;               {as specified}
   796:C        2     ert.offset          := 0;                   {no offset}
   797:C        2     ert.report          := 0;                   {data error log info only}
   798:C        2     ert.testarea        := ord(ert_area);       {as specified}
   799:C        2     ert.datasource      := 0;                   {internal pattern table}
   800:C        2     HPIBshort_msge_out(uep, command_sec, addr(ert), sizeof(ert));
   801:C        2     if ert_message then
   802:C        3       try
   803:C        4         HPIBwait_for_ppol(uep);
   804:C        4         HPIBshort_msge_in(uep, execution_sec, addr(log_entry), sizeof(log_entry));
   805:C        4       recover
   806:C        4         with bip_type(uep^.dvrtemp)^ do  {confirm the "premature" eoi}
   807:C        5           begin
   808:C        5             if (escapecode<>-10) or (iores<>zbadhardware) then
   809:C        6               escape(escapecode);
   810:C        5             iores := inoerror;
   811:C        5             ert_message := false;
   812:C        5           end; {recover}
   813:C        2     HPIBwait_for_ppol(uep);
   814:C        2     pattern_ert := qstat(uep);
   815:C        2   end; {pattern_ert}
```

```
816:D        1 $page$
817:S
818:D        1 function set_3V_address(uep: uep_type; cyln: unsgn24; head: unsgn8; sect: signed16): unsgn8;
819:S            {
820:S              issue the following command sequence:
821:S                . SET_UNIT
822:S                . SET_VOLUME
823:S                . SET_ADDRESS
824:S              return the QSTAT byte
825:D        2     }
826:D        2     var
827:D        2       s3va: {the 10 bytes in the command message}
828:D        2         packed record
829:D        2           setunitvol: setunitvol_type;
830:D        2           nop: CMD_type;
831:D        2           setadd: CMD_type;
832:D        2           tva: tva_type;
833:D   -10  2         end;
834:C        2     begin {set_3V_address}
835:C        2       s3va.setunitvol := suv_CMD_pair(uep);
836:C        2       s3va.nop        := CMDno_op;
837:C        2       s3va.setadd     := CMDset_address_3V;
838:C        2       s3va.tva.cyln   := cyln;
839:C        2       s3va.tva.head   := head;
840:C        2       s3va.tva.sect   := sect;
841:C        2       HPIBshort_msge_out(uep, command_sec, addr(s3va), sizeof(s3va));
842:C        2       HPIBwait_for_ppol(uep);
843:C        2       set_3V_address := qstat(uep);
844:C        2     end; {set_3V_address}
845:S
846:S
847:D        1 function spare_block(uep: uep_type; sparingmode: unsgn8): unsgn8;
848:S            {
849:S              issue the following command sequence:
850:S                . SET_UNIT
851:S                . SET_VOLUME
852:S                . SPARE_BLOCK
853:S              return the QSTAT byte
854:D        2     }
855:D        2     var
856:D        2       sb: {the 4 bytes in the command message}
857:D        2         packed record
858:D        2           setunitvol: setunitvol_type;
859:D        2           spareblock: CMD_type;
860:D        2           sparingmode: unsgn8;
861:D    -4  2         end;
862:C        2     begin {spare_block}
863:C        2       sb.setunitvol   := suv_CMD_pair(uep);
864:C        2       sb.spareblock   := CMDspare_block;
865:C        2       sb.sparingmode  := sparingmode;
866:C        2       HPIBshort_msge_out(uep, command_sec, addr(sb), sizeof(sb));
867:C        2       HPIBwait_for_ppol(uep);
868:C        2       spare_block := qstat(uep);
869:C        2     end; {spare_block}
870:S
871:S
872:C        1 end; {CS80ir}
```



```
873:D        1 $page$
874:S
875:D        1 module Qminit;
876:S
877:D        1 import
878:D        1   sysglobals, bkgnd, discHPIB, CS80, CS80dsr, CS80ir, midecs;
879:S
880:D        1 export
881:D        1   function Qdevicename(uep: uep_type; var saved_ioresult: integer): string255;
882:D        1   function Qintdata(uep: uep_type): interleave_data;
883:D        1   function Qphydata(uep: uep_type): physical_data;
884:D        1   procedure Qgetinitparms;
885:D        1   procedure Qminitialize(uep: uep_type; interleave_factor: shortint);
886:S
887:D        1 implement {Qminit}
888:S
889:S
890:D        1 var
891:D   -38  1   describe_bytes: describe_type;
892:D   -40  1   formatting_option_byte: unsgn8;
893:D   -41  1   formatting_option_supported: boolean;
894:D   -42  1   diagnostic_to_be_run: boolean;
895:D   -44  1   initialize_options_byte: unsgn8;
896:D   -46  1   volume_ert_passes: unsgn8;
897:D   -48  1   clear_logs_option_byte: unsgn8;
898:S
899:S
900:D   -48  1 const
901:D   -48  1   default_initialize_options_byte = 0;
902:D   -48  1   default_volume_ert_passes = 2;
903:D   -48  1   default_clear_logs_option_byte = 1;  {clear ert logs only}
904:S
905:S
906:D        1 function Qintdata(uep: uep_type): interleave_data;
907:C        2   begin {Qintdata}
908:C        2     with describe_bytes do
909:C        3       begin
910:C        3         Qintdata.min := 1;                {minimum}
911:C        3         if formatting_option_supported
912:C        4           then Qintdata.max := 255        {maximum subject to format specified}
913:C        4           else Qintdata.max := mif;       {true maximum}
914:C        3         Qintdata.def :=  currentif;       {default}
915:C        3       end; {with}
916:C        2   end; {Qintdata}
917:S
918:S
919:D        1 function Qphydata(uep: uep_type): physical_data;
920:C        2   begin {Qphydata}
921:C        2     with describe_bytes do
922:C        3       begin
923:C        3         Qphydata.nspm := maxhadd+1;          {number of surfaces per medium}
924:C        3         Qphydata.ntps := maxcadd+1;          {number of tracks/surface}
925:C        3         if maxsadd>0                         {number of sectors per track}
926:C        4           then Qphydata.nspt := maxsadd+1                          {actual}
927:C        4           else Qphydata.nspt := (maxsvadd.lfb+1)*nbpb div 256;  {pseudo}
928:C        3       end; {with}
929:C        2   end; {Qphydata}
```

```
   930:D   -48  1 $page$
   931:S
   932:D        1 function Qdevicename(uep: uep_type; var saved_ioresult: integer): string255;
   933:S
   934:D        2   var
   935:D    -1  2     retry_required: boolean;
   936:D  -258  2     devicename: string255;
   937:D  -262  2     index: integer;
   938:D  -262  2     bcd_product_number: (within the describe bytes)
   939:D  -262  2       packed record case integer of
   940:D  -262  2         0: (dn:  unsgn24);
   941:D  -262  2         1: (bcd: packed array[1..6] of unsgn4);
   942:D  -266  2       end;
   943:D  -270  2     product_number: integer;
   944:D  -270  2     fixed_vol_byte:
   945:D  -270  2       packed record case integer of
   946:D  -270  2         0: (byte: unsgn8);
   947:D  -270  2         1: (bit:  packed array[0..7] of boolean);
   948:D  -272  2       end;
   949:S
   950:C        2   begin {Qdevicename}
   951:S
   952:C        2     ioresult := ord(inoerror);
   953:S
   954:C        2     try
   955:C        3       allocate_bkgnd_info(uep);
   956:S
   957:C        3       if HPIBamigo_identify(uep) div 256<>2 then
   958:C        4         ioresc_bkgnd(uep, znodevice);
   959:S
   960:C        3       repeat
   961:C        4         retry_required := false;
   962:C        4         if set_unitvol(uep)<>0 then
   963:C        5           handle_bad_status(uep, true, retry_required);
   964:C        4       until not retry_required;
   965:S
   966:C        3       repeat
   967:C        4         retry_required := false;
   968:C        4         if describe(uep, describe_bytes)<>0 then
   969:C        5           handle_bad_status(uep, true, retry_required);
   970:C        4       until not retry_required;
   971:S
   972:C        3       formatting_option_supported := false;  {until proven otherwise}
   973:C        3       if describe_bytes.ct.subset80 then  {see if multiple formats supported}
   974:C        4         try
   975:C        5           formatting_option_byte := 255;  {test value}
   976:C        5           repeat
   977:C        6             retry_required := false;
   978:C        6             if formatting_option(uep, formatting_option_byte)<>0 then
   979:C        7               handle_bad_status(uep, true, retry_required);
   980:C        6           until not retry_required;
   981:C        5           formatting_option_supported := true;
   982:C        5           formatting_option_byte := 0;  {device's primary format}
   983:C        5         recover
   984:C        5           with bip_type(uep^.dvrtemp)^ do  {confirm the command rejection}
   985:C        6             begin
   986:C        6               if (escapecode<>-10) or (iores<>zbadmode) then
   987:C        7                 escape(escapecode);
   988:C        6               iores := inoerror;
   989:C        6             end; {recover}
```



```
   990:S
   991:C        3       deallocate_bkgnd_info(uep);
   992:C        3     recover
   993:C        3       begin
   994:C        3         deallocate_bkgnd_info(uep);
   995:C        3         if escapecode<>-10 then escape(escapecode);
   996:C        3         ioresult := uep^.dvrtemp;  {flag problem with describe}
   997:C        3         if saved_ioresult=ord(inoerror) then  {report to miui}
   998:C        4           saved_ioresult := ioresult;
   999:C        3       end; {recover}
  1000:S
  1001:C        2     if ioresult=ord(inoerror) then
  1002:C        3       with describe_bytes do
  1003:C        4         begin
  1004:S
  1005:C        4           bcd_product_number.dn := dn;
  1006:C        4           product_number := 0;
  1007:C        4           for index := 1 to 5 do
  1008:C        5             product_number := product_number*10+bcd_product_number.bcd[index];
  1009:S
  1010:C        4           devicename := 'HP';
  1011:C        4           strwrite(devicename, 3, index, product_number:1);
  1012:S
  1013:C        4           case dt of
  1014:C        5             0:        devicename := devicename+' fixed disc';
  1015:C        5             1:        begin
  1016:C        5                         fixed_vol_byte.byte := describe_bytes.fvb;
  1017:C        5                         if fixed_vol_byte.bit[7-uep^.dv]
  1018:C        6                           then devicename := devicename+' fixed disc'
  1019:C        6                           else devicename := devicename+' removeable disc';
  1020:C        5                       end;
  1021:C        5             2:        devicename := devicename+' tape';
  1022:C        5             otherwise {can't further describe the generic device type};
  1023:S
  1024:C        5           end; {case}
  1025:C        4         end {with}
  1026:C        4     else
  1027:C        3       devicename := '<inaccessible CS80 device>';
  1028:S
  1029:C        2     Qdevicename := devicename;
  1030:S
  1031:C        2   end; {Qdevicename}
```

```
1032:D   -48  1 $page$
1033:S
1034:D  -256  1 procedure allow_modification(prompt: string255; var parm: unsgn8; maxparm: unsgn8);
1035:D  -256  2   var
1036:D  -512  2     response: string255;
1037:D  -520  2     newparm, i: integer;
1038:D  -521  2     modification_ok: boolean;
1039:C        2   begin {allow_modification}
1040:C        2     repeat
1041:C        3       write(prompt, ' (defaults to ', parm:1, ') ');
1042:C        3       readln(response);
1043:C        3       modification_ok := strlen(response)=0;
1044:C        3       if not modification_ok then
1045:C        4         try
1046:C        5           strread(response, 1, i, newparm);
1047:C        5           modification_ok := (i=strlen(response)+1) and (newparm>=0) and (newparm<=maxparm);
1048:C        5           if modification_ok then
1049:C        6             parm := newparm;
1050:C        5         recover
1051:C        5           if not (-escapecode in [8, 10]) then escape(escapecode);
1052:C        3     until modification_ok;
1053:C        2   end; {allow_modification}
1054:S
1055:S
1056:D        1 procedure Qgetinitparms;
1057:D        2   var
1058:D    -1  2     ch: char;
1059:C        2   begin {Qgetinitparms}
1060:C        2     with describe_bytes do
1061:C        3       begin
1062:S
1063:C        3         diagnostic_to_be_run    := not ct.subset80 and (dt<=1);  {CS80 discs only}
1064:C        3         initialize_options_byte := default_initialize_options_byte;
1065:C        3         volume_ert_passes       := default_volume_ert_passes;
1066:C        3         clear_logs_option_byte  := default_clear_logs_option_byte;
1067:S
1068:C        3         if extended_features_mode then
1069:C        4           begin {extended_features_mode stuff}
1070:C        4             writeln;
1071:C        4             if diagnostic_to_be_run then
1072:C        5               diagnostic_to_be_run := not yes('Suppress running the initial diagnostic?');
1073:C        4             allow_modification('Initialize options byte?', initialize_options_byte, 255);
1074:C        4             if not ct.subset80 and (dt<=1) then
1075:C        5               begin
1076:C        5                 allow_modification('Volume ert passes?', volume_ert_passes, 255);
1077:C        5                 allow_modification('Clear logs option byte?', clear_logs_option_byte, 255);
1078:C        5               end; {if}
1079:C        4           end {extended_features_mode stuff}
1080:C        4         else if dt=2 then  {it's a tape; user may want to force re-certification}
1081:C        5           begin
1082:C        5             writeln;
1083:C        5             writeln('Medium is a tape: do you wish to force');
1084:C        5             if yes('  re-certification if already certified?') then
1085:C        6               initialize_options_byte := 1;  {force re-certification}
1086:C        5           end; {if}
1087:S
1088:C        3         if formatting_option_supported then
1089:C        4           begin
1090:C        4             writeln;
1091:C        4             allow_modification('Formatting option?', formatting_option_byte, 239);
```



```
1092:C        4           end; {if}
1093:S
1094:C        3       end; {with}
1095:C        2   end; {Qgetinitparms}
```

```
1096:D   -48  1 $page$
1097:S
1098:D        1 procedure Qminitialize(uep: uep_type; interleave_factor: shortint);
1099:S
1100:D        2 var
1101:D    -1  2   retry_required: boolean;
1102:D    -4  2   diag_unit: unsgn4;
1103:D    -5  2   ert_msge: boolean;
1104:D    -8  2   de_log: de_log_type;
1105:D   -10  2   head: unsgn8;
1106:D -1080  2   log_bytes: log_info_type;
1107:D -1082  2   log_index: unsgn8;
1108:D -1086  2   this_entry_ptr: ^log_entry_type;
1109:D -1087  2   sparing_required: boolean;
1110:D -1090  2   sparing_tries: signed16;
1111:S
1112:D -1090  2 const
1113:D -1090  2   sector_ert_passes = 20;       {suspect sector scans}
1114:D -1090  2   track_ert_passes  = 20;       {spared track scans}
1115:D -1090  2   max_sparing_tries = 5;        {maximum attempts to spare a sector/track}
1116:S
1117:C        2 begin {Qminitialize}
1118:S
1119:C        2   try
1120:C        3     allocate_bkgnd_info(uep);
1121:C        3
1122:C        3     if diagnostic_to_be_run then
1123:C        4       begin
1124:C        4         diag_unit := uep^.du;  {try the explicit unit; may get illegal opcode}
1125:C        4         repeat
1126:C        5           try
1127:C        6             repeat  {make one pass of the complete diagnostic}
1128:C        7               retry_required := false;
1129:C        7               if initiate_diagnostic(uep, diag_unit, {loops} 1, {section} 0)<>0 then
1130:C        8                 handle_bad_status(uep, true, retry_required);
1131:C        7             until not retry_required;
1132:C        6           recover
1133:C        6             with bip_type(uep^.dvrtemp)^ do  {confirm the illegal opcode}
1134:C        7               if (escapecode<>-10) or (iores<>zbadmode) then
1135:C        8                 escape(escapecode)  {probably diagnostic_result}
1136:C        8               else if diag_unit=15 then  {unit 15 was rejected!!!}
1137:C        9                 ioresc_bkgnd(uep, zcatchall)
1138:C        9               else  {explicit unit diagnostic not supported; use unit 15}
1139:C        9                 begin
1140:C        9                   iores := inoerror;
1141:C        9                   diag_unit := 15;  {original CS/80 allows unit 15 only}
1142:C        9                   retry_required := true;
1143:C        9                 end; {if}
1144:C        5           until not retry_required;
1145:C        4         end; {if}
1146:S
1147:C        3     if formatting_option_supported then
1148:C        4       repeat  {set the user-specified formatting option}
1149:C        5         retry_required := false;
1150:C        5         if formatting_option(uep, formatting_option_byte)<>0 then
1151:C        6           handle_bad_status(uep, true, retry_required);
1152:C        5       until not retry_required;
1153:S
1154:C        3     invalidate_stateinfo(uep);  {formatting may change the blocksize!}
1155:S
```



```
1156:C        3     repeat  {initialize the medium}
1157:C        4       retry_required := false;
1158:C        4       if initialize_media(uep, initialize_options_byte, interleave_factor)<>0 then
1159:C        5         handle_bad_status(uep, true, retry_required);
1160:C        4     until not retry_required;
1161:S
1162:C        3     if not describe_bytes.ct.subset80 and (describe_bytes.dt<=1) then
1163:C        4       begin  {CS80 disc; perform testing and sparing}
1164:S
1165:C        4         repeat  {preset the drive to force logging of runtime data errors}
1166:C        5           retry_required := false;
1167:C        5           if preset_drive(uep)<>0 then
1168:C        6             handle_bad_status(uep, true, retry_required);
1169:C        5         until not retry_required;
1170:S
1171:C        4         repeat  {clear the ert logs}
1172:C        5           retry_required := false;
1173:C        5           if clear_logs(uep, clear_logs_option_byte)<>0 then
1174:C        6             handle_bad_status(uep, true, retry_required);
1175:C        5         until not retry_required;
1176:S
1177:C        4         repeat  {set address 0}
1178:C        5           retry_required := false;
1179:C        5           if set_3V_address(uep, {cyln} 0, {head} 0, {sect} 0)<>0 then
1180:C        6             handle_bad_status(uep, true, retry_required);
1181:C        5         until not retry_required;
1182:S
1183:C        4         if volume_ert_passes>0 then
1184:C        5           repeat  {run a complete volume ert the specified number of passes}
1185:C        6             retry_required := false;
1186:C        6             ert_msge := false;  {do not request an execution message}
1187:C        6             if pattern_ert(uep, volume_ert, volume_ert_passes, ert_msge, log_bytes.entry[1])<>0 then
1188:C        7               handle_bad_status(uep, true, retry_required);
1189:C        6           until not retry_required;
1190:S
1191:C        4         for de_log := ert to runtime do
1192:C        5           for head := 0 to describe_bytes.maxhadd do
1193:C        6             begin
1194:S
1195:C        6               repeat  {read the appropriate data error log}
1196:C        7                 retry_required := false;
1197:C        7                 if data_error_log(uep, de_log, head, log_bytes)<>0 then
1198:C        8                   handle_bad_status(uep, true, retry_required);
1199:C        7               until not retry_required;
1200:S
1201:C        6               for log_index := 1 to log_bytes.log_entries do
1202:C        7                 begin
1203:C        7                   this_entry_ptr := addr(log_bytes.entry[log_index]);
1204:C        7                   with this_entry_ptr^ do
1205:C        8                     begin
1206:S
1207:C        8                       repeat  {set address to the suspect sector}
1208:C        9                         retry_required := false;
1209:C        9                         if set_3V_address(uep, logical_cyln, logical_head, logical_sect)<>0 then
1210:C       10                           handle_bad_status(uep, true, retry_required);
1211:C        9                       until not retry_required;
1212:S
1213:C        8                       repeat  {run a sector ert; a message back implies sparing required}
1214:C        9                         retry_required := false;
1215:C        9                         sparing_required := true; {request an execution message}
```

```
1216:C  9                          if pattern_ert(uep, sector_ert, sector_ert_passes,
1217:C 10                                         sparing_required, this_entry_ptr^)<>0 then
1218:C 10                            handle_bad_status(uep, true, retry_required);
1219:C  9                        until not retry_required;
1220:S
1221:C  8                        sparing_tries := 0;
1222:C  8                        while sparing_required and (sparing_tries<max_sparing_tries) do
1223:C  9                          begin
1224:C  9                            sparing_tries := sparing_tries+1;
1225:S
1226:C  9                            repeat  {address the bad sector; may be different after the track ert!}
1227:C 10                              retry_required := false;
1228:C 10                              if set_3V_address(uep, logical_cyln, logical_head, logical_sect)<>0 then
1229:C 11                                handle_bad_status(uep, true, retry_required);
1230:C 10                            until not retry_required;
1231:S
1232:C  9                            repeat  {spare without retaining data}
1233:C 10                              retry_required := false;
1234:C 10                              if spare_block(uep, (sparingmode) 1)<>0 then
1235:C 11                                handle_bad_status(uep, true, retry_required);
1236:C 10                            until not retry_required;
1237:S
1238:C  9                            repeat  {set address to the beginning of the affected track}
1239:C 10                              retry_required := false;
1240:C 10                              if set_3V_address(uep, logical_cyln, logical_head, 0)<>0 then
1241:C 11                                handle_bad_status(uep, true, retry_required);
1242:C 10                            until not retry_required;
1243:S
1244:C  9                            repeat  {run a track ert; a message back implies sparing required again!!!}
1245:C 10                              retry_required := false;
1246:C 10                              sparing_required := true; {request an execution message}
1247:C 10                              if pattern_ert(uep, track_ert, track_ert_passes,
1248:C 11                                             sparing_required, this_entry_ptr^)<>0 then
1249:C 11                                handle_bad_status(uep, true, retry_required);
1250:C 10                            until not retry_required;
1251:S
1252:C  9                          end; {while}
1253:S
1254:C  8                        if sparing_required then ioresc_bkgnd(uep, zinitfail);
1255:S
1256:C  8                      end; {with this_entry_ptr^}
1257:S
1258:C  7                  end; {for log_index}
1259:S
1260:C  6              end; {for head}
1261:S
1262:C  4        end; {if device is a CS/80 disc}
```



```
1263:C  3 $page$
1264:S
1265:C  3     repeat  {describe volume again to update the media parameters}
1266:C  4       retry_required := false;
1267:C  4       if describe(uep, describe_bytes)<>0 then
1268:C  5         handle_bad_status(uep, true, retry_required);
1269:C  4     until not retry_required;
1270:S
1271:C  3     deallocate_bkgnd_info(uep);
1272:S
1273:C  3   recover
1274:C  3     begin
1275:C  3       abort_bkgnd_process(uep);
1276:C  3       ioresult := uep^.dvrtemp;
1277:C  3       uep^.dvrtemp := ord(inoerror);  {report the error only once}
1278:C  3       escape(-10);
1279:C  3     end; {recover}
1280:S
1281:C  2 end; {Qminitialize}
1282:S
1283:C  1 end {Qminit}
1284:S
1285:C  2 $include 'QMINIT'$;

No errors. No warnings.
          ***** Nonstandard language features enabled *****
```

Description

MIUI is the "mediainit user interface" and contains the device-independent portion of mediainit.

```
 1:D        0 $modcal, stackcheck off, iocheck off, debug off, range off, ovflcheck off$
 2:D        0
 3:D        0 $search 'IOLIB:KERNEL', 'INIT:DRVASM', 'INIT:DISCHPIB',
 4:D        0         'INIT:AMIGO', 'INIT:CS80', 'FMINIT', 'MINIT'$
 5:S
 6:S
 7:S
 8:D        0 program miui(input,output,listing);  {mediainit user interface program}
 9:S
10:D        1 import
11:D        1   sysglobals, misc, sysdevs, fs, iodeclarations, midecs, bkgnd,
12:D        1   Mminit, Hminit, Fminit, Xminit, Qminit, Bminit;
13:S
14:D        1 type
15:D        1   sup_dev_type = (M,N,U,V,W,H,F,C,D,P,X,B,Q); {supported devices}
16:S
17:D        1 var
18:D    -4  1   lun: integer;
19:D    -8  1   uep: uep_type;
20:D   -10  1   device: sup_dev_type;
21:S
22:S
23:D  -256  1 procedure fatal_ioresult(message: string255; iores: integer);
24:C        2   begin {fatal_ioresult}
25:C        2     writeln(message);
26:C        2     getioerrmsg(message, iores);
27:C        2     writeln(' ',message);
28:C        2     escape(-1);
29:C        2   end; {fatal_ioresult}
30:S
31:S
32:D        1 function on_same_medium(flun: unitnum): boolean;
33:C        2   begin {on_same_medium}
34:C        2     with unitable^[flun] do
35:C        3       on_same_medium := (sc=uep^.sc) and (ba=uep^.ba) and
36:C        3                         (du=uep^.du) and (dv=uep^.dv) and
37:C        3                         (letter=uep^.letter);
38:C        2   end; {on_same_medium}
```



```
39:D   -10  1 $page$
40:S
41:D        1 procedure write_program_header;
42:S
43:D        2   const
44:D        2     previd = daterec[year:84, day:4, month:6];
45:D        2     prevno = '3.0';
46:D        2     programname = 'Mediainit';          {title}
47:D        2     copyright1  = '      Copyright 1984 Hewlett-Packard Company.';
48:D        2     copyright2 =  '                     All rights reserved.';
49:D        2     monthstr = 'JanFebMarAprMayJunJulAugSepOctNovDec';
50:S
51:D        2   var
52:D    -2  2     date: daterec;
53:D    -6  2     time: timerec;
54:D   -10  2     i: integer;
55:D   -14  2     seconds: integer;
56:D   -24  2     todaysdate: string[9];
57:D   -34  2     timestring: string[8];
58:S
59:C        2   begin {write_program_header}
60:C        2
61:C        2     sysdate(date);
62:C        2     systime(time);
63:C        2
64:C        2     with date do
65:C        3       if (month in [1..12]) and (day>0) and (year<100) then
66:C        4         begin
67:C        4           todaysdate := '  -'+str(monthstr, month*3-2, 3)+'-  ';
68:C        4           if day>9 then todaysdate[1] := chr(day div 10 + ord('0'));
69:C        4           todaysdate[2] := chr(day mod 10 + ord('0'));
70:C        4           todaysdate[8] := chr(year div 10 + ord('0'));
71:C        4           todaysdate[9] := chr(year mod 10 + ord('0'));
72:C        4         end {then}
73:C        4       else
74:C        4         todaysdate := '         ';
75:C        2
76:C        2     setstrlen(timestring,0);
77:C        2     i := 1;
78:C        2     if time.hour<10 then
79:C        3       strwrite(timestring,i,i,'0');
80:C        2     strwrite(timestring,i,i,time.hour:1,':');
81:C        2     if time.minute<10 then
82:C        3       strwrite(timestring,i,i,'0');
83:C        2     strwrite(timestring,i,i,time.minute:1,':');
84:C        2     seconds := time.centisecond div 100;
85:C        2     if seconds<10 then
86:C        3       strwrite(timestring,i,i,'0');
87:C        2     strwrite(timestring,i,i,seconds:1);
88:S
89:C        2     writeln(programname,' [Rev ',prevno,' ',previd.month:2,'/',previd.day:2,
90:C        2             '/',previd.year:2,']',todaysdate:10,' ',timestring);
91:C        2     writeln;
92:C        2     writeln(copyright1);
93:C        2     writeln(copyright2);
94:C        2     writeln;
95:S
96:C        2   end; {write_program_header}
```

```
    97:D   -10  1 $page$
    98:S
    99:D        1 procedure request_vol;
   100:S
   101:D        2   type
   102:D        2     string27 = string[27];
   103:D        2     dev_name_type = array [M..B] of string27;                    {rdq}
   104:S
   105:D        2   const
   106:D        2     dev_name = {device names}
   107:D        2       dev_name_type
   108:D        2         [ {M}  'internal minifloppy',
   109:D        2           {N}  'HP8290X series minifloppy',
   110:D        2           {U}  'HP913XA series hard disc',
   111:D        2           {V}  'HP913XB series hard disc',
   112:D        2           {W}  'HP913XC series hard disc',
   113:D        2           {H}  'HP9895 flexible disc',
   114:D        2           {F}  'HP9885 flexible disc',
   115:D        2           {C}  'HP7906 removeable hard disc',
   116:D        2           {D}  'HP7906 fixed hard disc',
   117:D        2           {P}  'HP7920 hard disc',
   118:D        2           {X}  'HP7925 hard disc',
   119:D        2           {B}  'HP98259 bubble memory'];                         {rdq}
   120:D        2     nonabortive_ioresult_set = [ord(inoerror), ord(zbadblock), ord(znoblock),
   121:D        2                                 ord(zuninitialized), ord(inodirectory)];
   122:S
   123:D        2   var
   124:D  -256  2     response: string255;
   125:D  -274  2     fvid: vid;
   126:D  -396  2     ftitle: fid;
   127:D  -400  2     fsegs: integer;
   128:D  -402  2     fkind: filekind;
   129:D  -403  2     scantitle_ok: boolean;
   130:D  -408  2     saved_ioresult: integer;
   131:D  -664  2     devicename: string255;
   132:S
   133:C        2   begin {request_vol}
   134:S
   135:C        2     repeat
   136:C        3       write('Volume ID? ');
   137:C        3       readln(response);
***WARNING: (line  138): STRPOS does not conform to HP standard, see $SWITCH_STRPOS$
   138:C        3       if (strlen(response)=0) or (strpos(#27,response)<>0) then escape(-1);
   139:C        3       scantitle_ok := scantitle(response, fvid, ftitle, fsegs, fkind);
   140:C        3       if scantitle_ok and (ftitle='') then
   141:C        4         begin
   142:C        4           lun := findvolume(fvid, true);
   143:C        4           saved_ioresult := ioresult;          {for decoding below}
   144:C        4           getioerrmsg(response, ord(inounit));  {for the case lun=0}
   145:C        4         end {then}
   146:C        4       else
   147:C        4         begin
   148:C        4           lun := 0;
   149:C        4           response := 'illegal syntax';
   150:C        4         end; {else}
   151:C        3       if lun=0 then
   152:C        4         writeln('   ',response);
   153:C        3     until lun<>0;
   154:S
   155:C        2     extended_features_mode := fsegs<>0;
```



```
   156:C        2     if extended_features_mode then
   157:C        3       begin
   158:C        3         writeln;
   159:C        3         writeln   ('*** Warning ***');
   160:C        3         writeln   ('  You have invoked an "extended features mode",');
   161:C        3         writeln   ('  which for certain devices requires you to have');
   162:C        3         writeln   ('  extensive knowledge about the device''s command');
   163:C        3         if not yes('  set.  Are you SURE you want to proceed?') then
   164:C        4           escape(-1);
   165:C        3       end; {if}
   166:S
   167:C        2     uep := addr(unitable^[lun]);
   168:C        2     with uep^ do
   169:C        3       begin
   170:C        3         case letter of
   171:C        4           'M':                device := M;
   172:C        4           'N':                device := N;
   173:C        4           'U':                device := U;
   174:C        4           'V':                device := V;
   175:C        4           'W':                device := W;
   176:C        4           'H':                device := H;
   177:C        4           'F':                device := F;
   178:C        4           'C':  if dv=0 then device := C
   179:C        5                             else device := D;
   180:C        4           'P':                device := P;
   181:C        4           'X':                device := X;
   182:C        4           'B':                device := B;                     {rdq}
   183:C        4           'Q':                device := Q;
   184:C        4           otherwise
   185:C        4             begin
   186:C        4               writeln('Mediainit does not support this device');
   187:C        4               escape(-1);
   188:C        4             end;
   189:C        4         end; {case}
   190:S
   191:C        3         if device=Q
   192:C        4           then devicename := Qdevicename(uep, saved_ioresult)  {may do I/O, alter saved_ioresult}
   193:C        4           else devicename := dev_name[device];
   194:C        3         writeln;
   195:C        3         write('Device: ',devicename);
   196:S
   197:C        3         if device<>M then {it has a select code & possibly a bus address}
   198:C        4           if (device=F) or (device=B) {it has a select code only}       {rdq}
   199:C        5             then write(', ',sc:1)
   200:C        5             else write(', ',100*sc+ba:1);
   201:C        3         if device<>B then write(', ',du:1);                      {rdq}
   202:C        3         if device in [C,D,Q] then write(', ',dv:1);
   203:C        3         writeln;
   204:C        3       end; {with uep^}
   205:S
   206:C        2     write('Logical unit #',lun:1,' - ');
   207:C        2     if fvid[1]<>'#' then writeln('''',fvid,':''')               {rdq}
   208:C        3     else                                                        {rdq}
   209:C        3       if (saved_ioresult in nonabortive_ioresult_set) or        {rdq}
   210:C        4          ((device=B) and (saved_ioresult<>ord(znodevice)) and   {rdq}
   211:C        4           (saved_ioresult<>ord(zbadhardware)))                  {rdq}
   212:C        4       then writeln('<no directory>')                            {rdq}
   213:C        4       else
   214:C        4         fatal_ioresult('<Mediainit aborted>', saved_ioresult);
   215:S
```

```
   216:C        2   end; {request_vol}
```



```
   217:D   -10  1 $page$
   218:S
   219:D        1 procedure other_volume_check;
   220:D        2   var
   221:D    -1  2     warning_issued: boolean;
   222:D    -4  2     flun: unitnum;
   223:D    -6  2     line: shortint;
   224:D   -18  2     CRTline, CRTcolumn, stopline: integer;
   225:D   -36  2     fvid: vid;
   226:D   -36  2   const
   227:D   -36  2     maxlines = 10;
   228:D   -36  2     fieldwidth = 16;
   229:C        2   begin {other_volume_check}
   230:S
   231:C        2     warning_issued := false;
   232:S
   233:C        2     for flun := 1 to maxunit do
   234:C        3       if on_same_medium(flun) then
   235:C        4         with unitable^[flun] do
   236:C        5           begin
   237:C        5             if flun<>lun then
   238:C        6               begin
   239:C        6                 if not warning_issued then
   240:C        7                   begin
   241:C        7                     writeln('WARNING: the initialization will also destroy:');
   242:C        7                     warning_issued := true;
   243:C        7                     line := 1;
   244:C        7                   end; {if}
   245:S
   246:C        6                 fgetxy(output,CRTcolumn, CRTline);
   247:C        6                 write(' #',flun:1);
   248:C        6                 fgotoxy(output,CRTcolumn+5, CRTline);
   249:C        6                 call(dam, fvid, flun, getvolumename);
   250:C        6                 if strlen(fvid)>0
   251:C        7                   then write(fvid,':')
   252:C        7                   else write('<no dir>');
   253:C        6                 if CRTcolumn=0
   254:C        7                   then writeln                (scrolls the screen if necessary)
   255:C        7                   else fgotoxy(output,CRTcolumn, CRTline+1);
   256:S
   257:C        6                 line := line+1;
   258:C        6                 if line>maxlines then
   259:C        7                   begin
   260:C        7                     fgetxy(output,CRTcolumn, stopline);
   261:C        7                     fgotoxy(output,CRTcolumn+fieldwidth, stopline-maxlines);
   262:C        7                     line := 1;
   263:C        7                   end; {if}
   264:S
   265:C        6               end; {if flun<>lun}
   266:C        5             umediavalid := false;
   267:C        5           end; {with}
   268:S
   269:C        2     if warning_issued and (CRTcolumn<>0) then
   270:C        3       fgotoxy(output,0, stopline);
   271:S
   272:C        2   end; {other_volume_check}
```

```
273:D   -10  1 $page$
274:S
275:D        1 procedure user_confirmation;
276:C        2   begin {user_confirmation}
277:C        2     writeln;
278:C        2     if not yes('Are you SURE you want to proceed?') then
279:C        3       escape(-1);
280:C        2   end; {user_confirmation}
281:S
282:S
283:D        1 procedure initialize_medium;
284:S
285:D        2   var
286:D    -2  2     interleave: shortint;
287:S
288:D    -6  2   procedure usi(intdata: interleave_data);
289:D    -6  3     var
290:D  -262  3       response: string255;
291:D  -266  3       i: integer;
292:D  -267  3       number_conversion_ok: boolean;
293:C        3     begin {user-specified interleave}
294:C        3       with intdata do
295:C        4         if (def>=min) and (def<=max) and (min<max) then
296:C        5           begin
297:C        5             writeln;
298:C        5             repeat
299:C        6               write('Interleave factor? ');
300:C        6               if max<>255 then
301:C        7                 write('[', min:1, '..', max:1, '] ');
302:C        6               write('(defaults to ', def:1, ') ');
303:C        6               readln(response);
304:C        6               number_conversion_ok := strlen(response)=0;
305:C        6               if number_conversion_ok then
306:C        7                 interleave := def
307:C        7               else
308:C        7                 try
309:C        8                   strread(response, 1, i, interleave);
310:C        8                   number_conversion_ok := i=strlen(response)+1;
311:C        8                 recover
312:C        8                   if not (-escapecode in [8, 10]) then escape(escapecode);
313:C        6             until number_conversion_ok and (interleave>=min) and (interleave<=max);
314:C        5           end {then}
315:C        5         else
316:C        5           interleave := def;
317:C        3     end; {user-specified interleave}
318:S
319:D        2   function fudged_d(dev: sup_dev_type; letter: char): char;
320:C        3     begin {fudged_d}
321:C        3       if dev=D
322:C        4         then fudged_d := 'D'
323:C        4         else fudged_d := letter;
324:C        3     end; {fudged_d}
```



```
325:D    -2  2 $page$
326:S
327:C        2   begin {initialize_medium}
328:C        2         { determine interleave factor }
329:C        2     case device of
330:C        3       M:          usi(Mintdata);
331:C        3       N:          usi(Nintdata(uep));
332:C        3       U,V,W:      interleave := 9; {N/A but passed as a parmameter}
333:C        3       H:          usi(Hintdata);
334:C        3       F:          usi(Fintdata);
335:C        3       Q:          begin
336:C        3                     Qgetinitparms;
337:C        3                     usi(Qintdata(uep));
338:C        3                   end;
339:C        3       otherwise  {not applicable};
340:C        3     end; {case}
341:S
342:C        2     lockup;
343:C        2     try
344:C        3       writeln;
345:C        3       writeln('Medium initialization in progress');
346:C        3       with uep^ do
347:C        4         case device of
348:C        5           M:          Mminitialize(interleave);
349:C        5           H,N,U,V,W:  Hminitialize(uep, interleave);
350:C        5           F:          Fminitialize(isc_table[sc].card_ptr, du, interleave);
351:C        5           C,D,P,X:    Xminitialize(fudged_d(device, letter), sc, ba, du);
352:C        5           Q:          Qminitialize(uep, interleave);
353:C        5           B:          Bminitialize(lun);                          {rdq}
354:C        5         end; {case}
355:C        3       writeln('Medium initialization completed');
356:C        3       escape(0);  {to do the lockdown}
357:C        3     recover
358:C        3       begin
359:C        3         lockdown;
360:C        3         if escapecode=0 then
361:C        4           {do nothing}
362:C        4         else if escapecode=-10 then
363:C        5           fatal_ioresult(#7'Medium initialization aborted:', ioresult)
364:C        5         else
365:C        5           escape(escapecode);
366:C        3       end; {recover}
367:S
368:C        2   end; {initialize_medium}
369:S
370:S
371:D        1 procedure make_initial_directories;
372:D        2
373:D        2   var
374:D    -2  2     flun: unitnum;
375:D   -82  2     dircatentry: catentry;
376:D  -744  2     dam_fib: fib;
377:D  -748  2     j: integer;
378:S
379:D        2   procedure check_ioresult;
380:C        3     begin
381:C        3       if ioresult<>ord(inoerror) then
382:C        4         fatal_ioresult(#7'Volume zeroing aborted:', ioresult);
383:C        3     end;
```

```
   384:D  -748  2 $page$
   385:S
   386:D        2   procedure make_initial_LIF_directory;
   387:S
   388:D        3     type
   389:D        3       bcd = 0..9;
   390:D        3       bcd12 = packed array[1..12] of bcd;
   391:D        3       b15 = 0..32767;
   392:S
   393:D        3       LIF_vol_type =
   394:D        3         packed record
   395:D        3           id: shortint;
   396:D        3           vol_label: packed array[1..6] of char;
   397:D        3           dir_sa: integer;
   398:D        3           oct_10000: shortint;
   399:D        3           dummy: shortint;
   400:D        3           dir_len: integer;
   401:D        3           version: shortint;
   402:D        3           zero: shortint;
   403:D        3           phydata: physical_data;
   404:D        3           vol_stamp: bcd12;
   405:D        3         end;
   406:S
   407:D        3       LIF_dir_entry =
   408:D        3         packed record
   409:D        3           file_name: packed array[1..10] of char;
   410:D        3           file_type: shortint;
   411:D        3           start_add: integer;
   412:D        3           file_len: integer;
   413:D        3           toc: bcd12;
   414:D        3           l_flag: boolean;
   415:D        3           vol_num: b15;
   416:D        3           implem: integer;
   417:D        3         end;
   418:S
   419:D        3       sector_type =
   420:D        3         record case integer of
   421:D        3           1: (LIF_vol_label: LIF_vol_type);
   422:D        3           2: (LIF_directory: array[0..7] of LIF_dir_entry);
   423:D        3         end;
   424:S
   425:D        3     const
   426:D        3       null_bcd12 = bcd12[0,0,0,0,0,0,0,0,0,0,0,0];
   427:S
   428:D        3       P_LIF_vl = {standard LIF volume label}
   429:D        3         LIF_vol_type
   430:D        3           [id: -32768,
   431:D        3            vol_label: 'P9826 ',
   432:D        3            dir_sa: 2,
   433:D        3            oct_10000: 4096,
   434:D        3            dummy: 0,
   435:D        3            dir_len: 2,
   436:D        3            version: 1,
   437:D        3            zero: 0,
   438:D        3            phydata: physical_data[ntps: 0, nspm: 0, nspt: 0],    {filled in later}
   439:D        3            vol_stamp: null_bcd12 ];
   440:S
   441:D        3       P_LIF_leod = {logical end of directory mark}
   442:D        3         LIF_dir_entry
   443:D        3           [file_name: '
```



```
   444:D        3             file_type: -1,
   445:D        3             start_add: 0,
   446:D        3             file_len: 0,
   447:D        3             toc: null_bcd12,
   448:D        3             l_flag: false,
   449:D        3             vol_num: 0,
   450:D        3             implem: 0 ];
   451:S
   452:D        3     var
   453:D        3       buffer: {four sector buffer for LIF volume label and dummy directory}
   454:D        3         record case integer of
   455:D        3           0: (long: array[0..255] of integer);
   456:D        3           1: (sector: array[0..3] of sector_type);
   457:D -1024  3         end;
   458:D -1026  3       i: shortint;
   459:D -1688  3       tm_fib: fib;
   460:S
   461:C        3     begin {make_initial_LIF_directory}
   462:S
   463:C        3       with buffer do
   464:C        4         begin
   465:S
   466:C        4           for i := 0 to 255 do    {zero the entire buffer}
   467:C        5             long[i] := 0;
   468:S
   469:C        4           with sector[0], LIF_vol_label do
   470:C        5             begin
   471:C        5               LIF_vol_label := P_LIF_vl;
   472:C        5               case device of
   473:C        6                 M:                      phydata := Mphydata;
   474:C        6                 H,N,U,V,W,C,D,P,X:      phydata := Hphydata(uep);
   475:C        6                 F:                      phydata := Fphydata;
   476:C        6                 Q:                      phydata := Qphydata(uep);
   477:C        6                 B:                      phydata := Bphydata(lun);         {rdq}
   478:C        6               end; {case}
   479:C        5             end; {with}
   480:S
   481:C        4           sector[2].LIF_directory[0] := P_LIF_leod;
   482:S
   483:C        4         end; {with}
   484:S
   485:C        3       with tm_fib do
   486:C        4         begin
   487:C        4           funit := lun;
   488:C        4           fileid := 0;
   489:C        4           fpeof :=  sizeof(buffer);
   490:C        4         end; {with}
   491:S
   492:C        3       uep^.ureportchange := false;
   493:C        3       call(uep^.tm, addr(tm_fib), writebytes, buffer, sizeof(buffer), 0);
   494:C        3       check_ioresult;
   495:C        3       call(uep^.tm, addr(tm_fib), flush     , buffer, 0, 0); {flush tapebuf}
   496:C        3       check_ioresult;
   497:C        3       call(uep^.tm, addr(tm_fib), clearunit , buffer, 0, 0); {update tape parms}
   498:C        3       check_ioresult;
   499:C        3       uep^.ureportchange := true;
   500:S
   501:C        3     end; {make_initial_LIF_directory}
```

```
   502:D  -748  2 $page$
   503:S
   504:C        2   begin {make_initial_directories}
   505:S
   506:C        2     writeln;
   507:C        2     writeln('Volume zeroing in progress');
   508:S
   509:C        2     if uep^.byteoffset=0 then
   510:C        3       make_initial_LIF_directory;
   511:S
   512:C        2     for flun := maxunit downto 1 do
   513:C        3       with unitable^[flun] do
   514:C        4         if (flun=lun) or (on_same_medium(flun) and (byteoffset<>uep^.byteoffset)) then
   515:C        5           begin
   516:C        5             with dircatentry do
   517:C        6               begin
   518:C        6                 cname := 'V';
   519:C        6                 strwrite(cname, 2, j, flun:1);
   520:C        6                 ureportchange := false;  {for ueovbytes' sake}
   521:C        6                 cpsize := ueovbytes(flun);
   522:C        6                 ureportchange := true;
   523:C        6                 cextral := 0;                {let dam decide # of dir entries}
   524:C        6               end; {with}
   525:C        5             with dam_fib do
   526:C        6               begin
   527:C        6                 funit := flun;
   528:C        6                 fwindow := addr(dircatentry);
   529:C        6                 call(dam, fvid, flun, getvolumename);          {assign fvid}
   530:C        6                 pathid := -1;
   531:C        6                 ftitle := '';
   532:C        6               end; {with}
   533:C        5             call(dam, dam_fib, flun, makedirectory);
   534:C        5             check_ioresult;
   535:C        5           end; {then}
   536:S
   537:C        2     writeln('Volume zeroing completed');
   538:S
   539:C        2   end; {make_initial_directories}
```



```
   540:D   -10  1 $page$
   541:S
   542:C        1 begin {miui}
   543:C        1   try
   544:C        2     initialize_bkgnd;
   545:C        2     write_program_header;
   546:C        2     request_vol;
   547:C        2     other_volume_check;
   548:C        2     user_confirmation;
   549:C        2     initialize_medium;
   550:C        2     make_initial_directories;
   551:C        2   recover
   552:C        2     if escapecode=-20 then
   553:C        3       begin
   554:C        3         writeln;
   555:C        3         writeln('Mediainit stopped by user');
   556:C        3       end {then}
   557:C        3     else
   558:C        3       escape(escapecode);
   559:C        1 end. {miui}
   560:S
   561:S

No errors. 1 warnings.
            ***** Nonstandard language features enabled *****
```

MOUSE

Description

MOUSE provides driver code for the mouse and other HPHIL relative locator input devices.

Requirements

SYSDEVS, and SYSGLOBALS.

Notes

HPHIL must be *executed* either before or after MOUSE executes for the mouse to be functional.

```
     1:S          (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1984.
     4:S          All rights are reserved.  Copying or other
     5:S          reproduction of this program except for archival
     6:S          purposes is prohibited without the prior
     7:S          written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                       RESTRICTED RIGHTS LEGEND
    11:S
    12:S          Use, duplication, or disclosure by the Government
    13:S          is subject to restrictions as set forth in
    14:S          paragraph (b) (3) (B) of the Rights in Technical
    15:S          Data and Computer Software clause in
    16:S          DAR 7-104.9(a).
    17:S
    18:S          HEWLETT-PACKARD COMPANY
    19:D        0 Fort Collins, Colorado                        *)
    20:S
    21:D        0 $sysprog$
    22:D        0 $stackcheck off$ $heap_dispose off$
    23:D        0 $iocheck off$ $range off$ $ovflcheck off$
    24:D        0 $debug off$
    25:S
    26:D        0 program mouseinit;
    27:S
    28:D        1 module mouse; { mouse module }
    29:D        1 import sysdevs,sysglobals;
    30:S
    31:D        1 export
    32:S
    33:D        1   function initmouse:boolean;
    34:S
    35:D        1 implement
    36:D        1   type
    37:D        1     sint = -32768..32767;
    38:S
    39:D        1     pollblock = packed record
    40:D        1                   case integer of
    41:D        1                   0:(d: packed array[1..7] of byte);
    42:D        1                   1:(twosets  : boolean;
    43:D        1                      kcodes   : 0..3;
    44:D        1                      filler   : 0..7;
    45:D        1                      numaxes  : 0..3;
    46:D        1                      xdata    : sint;
    47:D        1                      ydata    : sint;
    48:D        1                      zdata    : sint)
    49:D        1                 end;
    50:D        1     lkeytype = packed record
    51:D        1                  case integer of
    52:D        1                  0:(b:byte);
    53:D        1                  1:(row:0..15;
    54:D        1                     col:0..7;
    55:D        1                     up : boolean)
    56:D        1                end;
    57:D        1     kmtype = array[0..7] of byte;
    58:D        1   const
    59:D        1     mkeymap = kmtype[3,13,8,0,0,0,0,0]; { ex,cr,bs,.... }
    60:D        1   var
```



```
    61:D    -4  1     driver : loopdvrptr;
    62:D   -12  1     keystate : array[0..7] of boolean;
    63:D   -24  1     cumdata  : array[1..3] of integer;
    64:D   -32  1     gdata    : pollblock;
    65:D   -34  1     cindex   : 0..3;
    66:S
    67:D        1   procedure doreset;
    68:D    -2  2   var i : sint;
    69:C        2   begin
    70:C        2     for i:=0 to 7 do keystate[i]:=true; { all keys up }
    71:C        2     for i:=1 to 7 do gdata.d[i]:=0;
    72:C        2     for i:=1 to 3 do cumdata[i]:=0;
    73:C        2   end;
    74:S
    75:D        1   procedure mdataproc(var statbyte,databyte:byte; var done:boolean);
    76:D        2   var
    77:D    -2  2     k      : lkeytype;
    78:C        2   begin {mdataproc;  mouse data proc }
    79:C        2    WITH LOOPCONTROL^ DO                    {4/9/84 SFB}
    80:C        3     with loopdevices[loopdevice] do
    81:C        4     case devstate of
    82:C        5     1:begin
    83:C        5         gdata.d[1]:=databyte; cindex:=0;
    84:C        5         if gdata.numaxes=0 then devstate:=8 else devstate:=2;
    85:C        5       end;
    86:C        5     2,4,6:
    87:C        5       begin  { co-ord data } { low byte }
    88:C        5         if not descrip.size16 then
    89:C        6           if databyte>127 then gdata.d[devstate]:=255 { sign extend }
    90:C        7                           else gdata.d[devstate]:=0;
    91:C        5         gdata.d[devstate+1]:=databyte;
    92:C        5         devstate:=devstate+1+ord(not descrip.size16);
    93:C        5         if not descrip.size16 then
    94:C        6         begin   { 8 bit mode }
    95:C        6           cindex:=cindex+1;
    96:C        6           if cindex>=gdata.numaxes then devstate:=8;
    97:C        6         end;
    98:C        5       end;
    99:C        5     3,5,7:   { co-ord data } { high byte }
   100:C        5       begin
   101:C        5         gdata.d[devstate-1]:=databyte;
   102:C        5         devstate:=devstate+1;
   103:C        5         cindex:=cindex+1;
   104:C        5         if cindex>=gdata.numaxes then devstate:=8;
   105:C        5       end;
   106:C        5     8:begin { keydata }
   107:C        5         case gdata.kcodes of
   108:C        6         0:;     { not supposed to be any data }
   109:C        6         1:begin         { is all ready ascii }
   110:C        6             statbyte:=0;
   111:C        6             call(kbdisrhook,statbyte,databyte,done);
   112:C        6           end;
   113:C        6         otherwise       { key code data }
   114:C        6           k.b:=databyte;
   115:C        6           keystate[k.col]:=k.up; { record the state }
   116:C        6           if not k.up then
   117:C        7           begin
   118:C        7             statbyte:=0; databyte:=mkeymap[k.col];
   119:C        7             call(kbdisrhook,statbyte,databyte,done);
   120:C        7           end;
```

```
   121:C        6          end;
   122:C        5        end;
   123:C        5      end;
   124:C        2    end; {mdataproc}
   125:S
   126:D        1    procedure mopsproc(op:loopdvrop);
   127:D        2    var
   128:D    -4  2      limit : integer;
   129:D    -8  2      statbyte,databyte : byte;
   130:D    -9  2      done  : boolean;
   131:S
   132:D        2      procedure check(i:integer;d:sint);
   133:C        3      begin
   134:C        3        cumdata[i]:=cumdata[i]+d; done:=false;
   135:C        3        if abs(cumdata[i])>limit then
   136:C        4        begin
   137:C        4          done:=true;
   138:C        4          if (i=1) then
   139:C        5          begin
   140:C        5            statbyte:=hex('FF'); { normal} { x }
   141:C        5            if d>0 then databyte:=255 else databyte:=1;
   142:C        5          end
   143:C        5          else
   144:C        5          begin
   145:C        5            statbyte:=hex('EF'); { shift } { y }
   146:C        5            if d>0 then databyte:=1 else databyte:=255;
   147:C        5          end;
   148:C        4          cumdata[i]:=0;
   149:C        4        end;
   150:C        3      end;  { check }
   151:S
   152:C        2    begin { mopsproc; mouse ops }
   153:C        2     WITH LOOPCONTROL^ DO                    {4/9/84 SFB}
   154:C        3      with loopdevices[loopdevice] do
   155:C        4      case op of
   156:C        5        datastarting: devstate:=1;
   157:C        5        dataended:
   158:C        5          begin
   159:C        5            if descrip.size16 then limit:=descrip.counts div 8
   160:C        6                              else limit:=descrip.counts div 800;
   161:C        5            check(1,gdata.xdata);
   162:C        5            if done then
   163:C        6            begin call(rpgisrhook,statbyte,databyte,done);
   164:C        6                  gdata.ydata:=0;
   165:C        6            end
   166:C        6            else
   167:C        6            begin
   168:C        6              check(2,gdata.ydata);
   169:C        6              if done then
   170:C        7              begin call(rpgisrhook,statbyte,databyte,done);
   171:C        7                    gdata.xdata:=0;
   172:C        7              end;
   173:C        6            end;
   174:C        5          end;
   175:C        5        resetdevice: doreset;
   176:C        5      end;
   177:C        2    end; { mopsproc)
   178:D   -34  1
   179:D        1    function initmouse:boolean;
   180:C        2    begin
```



```
   181:C        2      if (kbdtype=itfkbd) and (driver=nil) then
   182:C        3       begin
   183:C        3        new(driver);
   184:C        3        with driver^ do
   185:C        4        begin     { initialize driver log in record }
   186:C        4          lowid:=96;       { all relative pointers }
   187:C        4          highic:=127;
   188:C        4          daddr:=0;        { any device address }
   189:C        4          opsproc:=mopsproc;      { set procedure vars }
   190:C        4          dataproc:=mdataproc;
   191:C        4          next:=loopdriverlist;   { add to driver list }
   192:C        4          loopdriverlist:=driver;
   193:C        4        end;
   194:C        3        if LOOPCONTROL^.loopisok                  {4/9/84 SFB}
   195:C        4         then CALL(HPHILCMDHOOK,CONFIGUREOP);     {4/9/84 SFB}
   196:C        3        initmouse:=true;
   197:C        3       end
   198:C        3      else initmouse:=false;
   199:C        2    end;  { initloop }
   200:S
   201:C        1  end;    { mouse interface  module }
   202:S
   203:D        1  import mouse, loader;
   204:S
   205:C        1  begin
   206:C        1    if initmouse then markuser;
   207:C        1  end.
   208:S
   209:S
```

No errors. No warnings.
***** Nonstandard language features enabled *****

NONUSKBD1

Description

NONUSKBD1 provides initialization for handling 98203 non-U.S.-language keyboards, and handles all Katakana keyboards.

Requirements

SYSDEVS, MISC, SYSGLOBALS, and ASM.

```
Pascal [Rev 3.0M  6/ 4/84] NONUSKBD1.TEXT          26-Dec-84 21:53:09 Page 1   *** File name: NONUSKBD1 ***

     1:S                                                    (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1984.
     4:S          All rights are reserved.  Copying or other
     5:S          reproduction of this program except for archival
     6:S          purposes is prohibited without the prior
     7:S          written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                       RESTRICTED RIGHTS LEGEND
    11:S
    12:S          Use, duplication, or disclosure by the Government
    13:S          is subject to restrictions as set forth in
    14:S          paragraph (b) (3) (B) of the Rights in Technical
    15:S          Data and Computer Software clause in
    16:S          DAR 7-104.9(a).
    17:S
    18:S          HEWLETT-PACKARD COMPANY
    19:D        0 Fort Collins, Colorado                        *)
    20:S
    21:D        0 $SYSPROG$
    22:D        0 $heap_dispose off$
    23:D        0 $iocheck off$
    24:D        0 $range off$ $ovflcheck off$
    25:D        0 $debug off$
    26:D        0 $stackcheck off$
    27:D        0 program NONUS1INIT;
    28:S
    29:D        1 module NON_US_KBD1;
    30:S
    31:D        1 import sysdevs,misc;
    32:D        1 export
    33:D        1   function initlang:boolean;
    34:S
    35:D        1 implement
    36:S
    37:D        1 procedure katakanatrans;
    38:D        2 label 1;
    39:D        2 { katakana language keycode->character code translations }
    40:S
    41:C        2 begin {katakanatrans}
    42:C        2   if langcom.status=0 then langcom.result:=alpha_key
    43:C        3   else
    44:C        3   if langcom.data>127 then langcom.result:=ignored_key
    45:C        4   else
    46:C        4   with langcom, langtable[0]^, keytable[data] do
    47:C        5   begin
    48:C        5     result := keyclass;
    49:C        5     if (result=ignored_key) and not kbdaltlock then goto 1;
    50:S
    51:C        5     if (data=18) or (data=19)  then        { change to/from  alternate }
    52:C        6       begin result:=ignored_key; kbdaltlock:=(data=18); end
    53:C        6     else
    54:C        6     if control and (kbdtype<>ITFKBD) and
    55:C        7        ((data=96) or (data=97)) then       { change to/from alternate }
    56:C        7       begin result:=ignored_key; kbdaltlock:=(data=96); end
    57:C        7     else
    58:C        7     if kbdaltlock then
    59:C        8     begin       { use alternate language }
    60:C        8       langindex:=1-langindex; extension:=false;
```

```
Pascal [Rev 3.0M  6/ 4/84] NONUSKBD1.TEXT          26-Dec-84 21:53:09 Page 2   *** File name: NONUSKBD1 ***

    61:C        8       call(langtable[langindex]^.semantics);
    62:C        8       langindex:=1-langindex;
    63:C        8     end
    64:C        8     else
    65:C        8     if result<>ignored_key then
    66:C        9     begin { normal processing }
    67:C        9       extension := false;
    68:C        9       shift:= shift and not no_shift;
    69:C        9       key  := keys[shift];
    70:C        9     end;
    71:C        5   end;  { with langcom etc. }
    72:C        2 1:end;    {katakanatrans}
    73:S
    74:D        1 function initlang:boolean;
    75:D        2 type
    76:D        2   t1 = packed array[boolean, 60..98] of char;
    77:D        2   t2 = packed array[FRENCH_KBD..KATAKANA_KBD] of t1;
    78:S
    79:D        2 const
    80:D        2  keylookup = t2[
    81:D        2         {french}
    82:D        2       t1['0..+123-456*789/E()^1234567890''#170#200'&'#197#203',.-',
    83:D        2          '0.[~123`456@789\<[]>!"#$%+/()=?'#171#181'*'#201#179';:_'],
    84:D        2          { german }
    85:D        2       t1['0..+123-456*789/E()^1234567890'#222''''#207'+'#206#204',.-',
    86:D        2          '0.[~123'#179'456@789\<[]>!"#$%&/()=?`'#219'*'#218#216';:_'],
    87:D        2          { swedish/finish }
    88:D        2       t1['0..+123-456*789/E()^1234567890+'#197#212#207#206#204',.-',
    89:D        2          '0.[~123''456@789\<[]>!"#$%&/()=?'#220#208#219#218#216';:_'],
    90:D        2          { spanish }
    91:D        2       t1['0..+123-456*789/E()^1234567890+'#168#179'#'#183'*,.-',
    92:D        2          '0.[+123''456*789\<[]>!"'#185'$%&'#184'()=?/()'#182'@;:_'],
    93:D        2          { katakana }
    94:D        2   t1[
    95:D        2 '0..+123-456*789/E()^'#199#204#177#179#180#181#212#213#214#220#206#205#209#219#218#185#200#217#210,
    96:D        2 '0..+123-456*789/`|\~'#199#204#167#169#170#171#172#173#174#166#176#205#222#223#218#185#164#161#165]
    97:D        2     ];  { KATAKANA TABLE IS NIMITZ BASED AS IN 2.x - MODS LATER IN CODE }
    98:D        2     {end of keylookup }
    99:D        2 type
   100:D        2   k2 = packed array [100..125] of 0..255;
   101:D        2   bytealphabettype = packed array [boolean] of k2;
   102:D        2
   103:D        2 const
   104:D        2   yencode=92;
   105:D        2   kanaalphabet = bytealphabettype
   106:D        2                      [k2[ 215,190,201,216,192,195,178,189,182,221,197,198,
   107:D        2                           193,196,188,202,183,184,207,211,194,187,191,203,
   108:D        2                           186,208 ],
   109:D        2                       k2[ 215,190,201,216,192,195,168,189,182,221,197,198,
   110:D        2                           193,196,188,202,183,184,207,211,175,187,191,203,
   111:D        2                           186,208]];
   112:D        2 var
   113:D    -2  2   i : -32768..32767;
   114:C        2 begin
   115:C        2     if (kbdlang=katakana_kbd) or
   116:C        3        (((kbdtype=largekbd) or (kbdtype=smallkbd)) and
   117:C        3        ((kbdlang>=french_kbd) and (kbdlang<=spanish_kbd))) and
   118:C        3        (langtable[0]<>nil)
   119:C        3   then
   120:C        3   if (langtable[0]^.langcode=us_kbd) then
```

```
   121:C        4   begin
   122:C        4     initlang := true;
   123:C        4     with langtable[0]^ do
   124:C        5     begin
   125:C        5       { extend char keys don't exist on non ITF keyboards }
   126:C        5       { and katakana wants pass thru mode }
   127:C        5       transmode := kpass_extc;
   128:C        5       langcode  := kbdlang;
   129:S
   130:C        5       if kbdlang=katakana_kbd then      { 5/14/84 RQ/SFB}
   131:C        6       begin
   132:C        6         if langtable[1]=nil then new(langtable[1]);
   133:C        6         langtable[1]^:=langtable[0]^;   { copy the us table }
   134:C        6       end;
   135:S
   136:C        5       for i:=60 to 98 do      { MOVED 5/14/84 RQ/SFB}
   137:C        6       begin { special alpha loading }
   138:C        6         keytable[i].keys[false]:=keylookup[kbdlang,false,i];
   139:C        6         keytable[i].keys[true] :=keylookup[kbdlang,true,i];
   140:C        6       end;
   141:C        5       keytable[99].keys[false]:=' '; keytable[99].keys[true]:=' ';
   142:S
   143:C        5       if kbdlang=katakana_kbd then
   144:C        6       begin
   145:C        6         semantics := katakanatrans;
   146:C        6         can_nonadv:= false;
   147:C        6         for i:=100 to 125 do
   148:C        7         begin   { load alpha keys }
   149:C        7           keytable[i].keys[false]:=chr(kanaalphabet[false,i]);
   150:C        7           keytable[i].keys[true] :=chr(kanaalphabet[true,i]);
   151:C        7         end;
   152:C        6         keytable[24].keyclass:=ignored_key;     { ignore capslock key }
   153:S
   154:C        6         if kbdtype=ITFkbd then
   155:C        7         begin { changes to keys 1, 2, 92 and 93 }
   156:C        7          keytable[1].keys[false]:=#219;
   157:C        7          keytable[1].keys[true]:=#176;
   158:C        7          keytable[2].keys[false]:=#209;
   159:C        7          keytable[2].keys[true]:=#163;
   160:C        7          keytable[92].keys[false]:=#222;
   161:C        7          keytable[92].keys[true]:=#222;
   162:C        7          keytable[93].keys[false]:=#223;
   163:C        7          keytable[93].keys[true]:=#162;
   164:C        7         end;
   165:C        6         IF (KBDTYPE = SMALLKBD) OR (KBDTYPE = LARGEKBD) THEN { RQ/SFB 5/11/84 }
   166:C        7          WITH LANGTABLE[1]^ DO
   167:C        8           BEGIN
   168:C        8            KEYTABLE[92].KEYS[TRUE] := '\';
   169:C        8            KEYTABLE[93].KEYS[TRUE] := '|';
   170:C        8           END;
   171:C        6         for i:=0 to 127 do keytable[i].no_capslock:=true;
   172:C        6       end;
   173:S
   174:S
   175:C        5       case kbdlang of
   176:C        6       german_kbd,swedish_kbd:
   177:C        6         begin
   178:C        6           keytable[92].no_capslock:=false;
   179:C        6           keytable[94].no_capslock:=false;
   180:C        6           keytable[95].no_capslock:=false;
```



```
   181:C        6           if kbdlang=german_kbd then
   182:C        7           begin
   183:C        7             keytable[120]:=keytable[109]; {z<=y}
   184:C        7             keytable[109].keys[false]:='z'; keytable[109].keys[true ]:='Z';
   185:C        7           end
   186:C        7           else
   187:C        7           begin       { swedish_kbd }
   188:C        7             keytable[91].no_capslock:=false;
   189:C        7             keytable[93].no_capslock:=false;
   190:C        7           end;
   191:C        6         end;
   192:C        6       spanish_kbd: BEGIN                              { 5/10/84 SFB }
   193:C        6                      keytable[94].no_capslock:=false;
   194:C        6                      KEYTABLE[91].KEYCLASS:=NONA_ALPHA_KEY;  { 5/11/84 SFB }
   195:C        6                    END;
   196:C        6       FRENCH_KBD : KEYTABLE[91].KEYCLASS:=NONADV_KEY;   { 5/11/84 SFB }
   197:C        6       otherwise
   198:C        6       end; { case }
   199:C        5       langindex:=0;
   200:C        5       kbdcapslock:=true;  kbdaltlock:=(kbdlang=katakana_kbd);
   201:C        5     end; { with langtable[0]^ }
   202:S
   203:C        4   end   { if my kind of keyboard }
   204:C        4   else initlang:=false;
   205:C        2 end;    {initlang}
   206:S
   207:C        1 end;    { module nonitflang }
   208:S
   209:D        1 import NON_US_KBD1, loader;
   210:S
   211:C        1 begin
   212:C        1   if initlang then markuser;
   213:C        1 end.
   214:S

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

NONUSKBD2

Description

NONUSKBD2 provides initialization for handling HPHIL non-U.S.-language keyboards.

Requirements

SYSDEVS, MISC, SYSGLOBALS, and ASM.

```
    1:S                                                 (*
    2:S
    3:S           (c) Copyright Hewlett-Packard Company, 1984.
    4:S          All rights are reserved.  Copying or other
    5:S          reproduction of this program except for archival
    6:S          purposes is prohibited without the prior
    7:S          written consent of Hewlett-Packard Company.
    8:S
    9:S
   10:S                       RESTRICTED RIGHTS LEGEND
   11:S
   12:S          Use, duplication, or disclosure by the Government
   13:S          is subject to restrictions as set forth in
   14:S          paragraph (b) (3) (B) of the Rights in Technical
   15:S          Data and Computer Software clause in
   16:S          DAR 7-104.9(a).
   17:S
   18:S          HEWLETT-PACKARD COMPANY
   19:D        0 Fort Collins, Colorado                      *)
   20:S
   21:D        0 $SYSPROG$
   22:D        0 $heap_dispose off$
   23:D        0 $iocheck off$
   24:D        0 $range off$ $ovflcheck off$
   25:D        0 $debug off$
   26:D        0 $stackcheck off$
   27:S
   28:D        0 program NON_US_KBD2INIT;
   29:S
   30:D        1 import sysdevs,misc;
   31:S
   32:D        1 type
   33:D        1   t1 = packed array[boolean, 80..98] of char;
   34:D        1   t2 = packed array[finish_kbd..swedish_kbd] of t1;
   35:D        1   t3 = packed array[boolean, 1..2] of char;
   36:D        1   t4 = packed array[finish_kbd..swedish_kbd] of t3;
   37:D        1
   38:D        1 const
   39:D        1   lowkeys = t4[
   40:D        1                {finish}          t3['<''','>*'],
   41:D        1                {belgian}         t3['$<',';>'],
   42:D        1                {canadian eng}    t3[']@','[#'],
   43:D        1                {canadian fr }    t3[']@','[#'],
   44:D        1                {norwegian}       t3['<@','>*'],
   45:D        1                {danish}          t3['<@','>*'],
   46:D        1                {dutch}           t3['@5','=#190{f}],
   47:D        1                {swiss gr}        t3['=$','3;'],
   48:D        1                {swiss fr}        t3['=$','3;'],
   49:D        1                {spanish_eur}     t3['<5','>3'],
   50:D        1                {spanish_lat}     t3['^9','58'],
   51:D        1                { ABOVE [FALSE,1] CHANGED TO '' 5/8/84 SFB }
   52:D        1                { ABOVE [TRUE ,1] CHANGED TO '5' 5/8/84 SFB }
   53:D        1                { ABOVE [FALSE,2] CHANGED TO '9' 5/10/84 SFB }
   54:D        1                {uk}              t3['<','~>'],
   55:D        1                {italian}         t3['<*','>='],
   56:D        1                {french}          t3['$<',',;>'],
   57:D        1                {german}          t3['<;','>^'],
   58:D        1                {swedish}         t3['<'',','>*']
   59:D        1                ];
   60:D        1    specials =t2[
```



```
   61:D        1                {finish}          t1['1234567890+ETONL,.-',
   62:D        1                                    '!"#$%&/()=?\P[ZX;:_'],
   63:D        1                {belgian}         t1['&E"''(=I!5H)-*`mK;:=',
   64:D        1                                    '12345678903_+*M%./+'],
   65:D        1                { ABOVE [FALSE,92] CHANGED TO "*" 5/8/84 SFB }
   66:D        1                {canadian eng}    t1['1234567890-=*5;),,E',
   67:D        1                                    '!"/$%?&*()_++'#180':'<>\'],
   68:D        1                { ABOVE [FALSE,95] CHANGED TO ")" 5/8/84 SFB }
   69:D        1                { ABOVE [FALSE,92] CHANGED TO '*' 5/9/84 SFB }
   70:D        1                {canadian fr }    t1['1234567890-=*5;),,E',
   71:D        1                                    '!"/$%?&*()_++'#180':'<>\'],
   72:D        1                { ABOVE [FALSE,95] CHANGED TO ")" 5/8/84 SFB }
   73:D        1                { ABOVE [FALSE,92] CHANGED TO '*' 5/9/84 SFB }
   74:D        1                {norwegian}       t1['1234567890+(T+VW,.-',
   75:D        1                                    '!"#$%&/()=?)P^RS;:_'],
   76:D        1                { ABOVE [FALSE,91] CHANGED TO '(' 5/9/84 SFB }
   77:D        1                { ABOVE [TRUE,91] CHANGED TO ')' 5/9/84 SFB }
   78:D        1                {danish}          t1['1234567890+(T+WV,.-',
   79:D        1                                    '!"=$%&/()=?)P^SR;:_'],
   80:D        1                { ABOVE [FALSE,91] CHANGED TO '(' 5/9/84 SFB }
   81:D        1                { ABOVE [TRUE,91] CHANGED TO ')' 5/9/84 SFB }
   82:D        1                {dutch}           t1['1234567890/|+<:(,.-',
   83:D        1                                    '!"#$%&_()''?\^>+};*='],
   84:D        1                { ABOVE [TRUE,89] CHANGED TO "}" 5/8/84 SFB }
   85:D        1                { ABOVE [FALSE,95] CHANGED TO "(" 5/9/84 SFB }
   86:D        1                { ABOVE [TRUE,89] CHANGED TO ")" 5/9/84 SFB }
   87:D        1                {swiss gr}        t1['1234567890!*0+NL,.-',
   88:D        1                                    '+"*5%&/()=?)I(EH;:_'],
   89:D        1                { ABOVE [TRUE,95] CHANGED TO 'H' 5/8/84 SFB }
   90:D        1                { ABOVE [TRUE,93] CHANGED TO '(' 5/9/84 SFB }
   91:D        1                { ABOVE [FALSE,91] CHANGED TO '*' 5/9/84 SFB }
   92:D        1                { ABOVE [TRUE,91] CHANGED TO ')' 5/9/84 SFB }
   93:D        1                {swiss fr}        t1['1234567890!*I+EH,.-',
   94:D        1                                    '+"*5%&/()=?)O(NL;:_'],
   95:D        1                { ABOVE [FALSE,95] CHANGED TO 'H' 5/8/84 SFB }
   96:D        1                { ABOVE [TRUE,93] CHANGED TO '(' 5/9/84 SFB }
   97:D        1                { ABOVE [FALSE,91] CHANGED TO '*' 5/9/84 SFB }
   98:D        1                { ABOVE [TRUE,91] CHANGED TO ')' 5/9/84 SFB }
   99:D        1                {spanish_eur}     t1['1234567890'')@+7(,.-',
  100:D        1                                    '!"9$%&/()=?8##6+;:_'],
  101:D        1                { ABOVE [FALSE,95] CHANGED TO '(' 5/8/84 SFB }
  102:D        1                { ABOVE [FALSE,91] CHANGED TO ')' 5/9/84 SFB }
  103:D        1                {spanish_lat}     t1['1234567890-=('7;,./',
  104:D        1                                    '!@#$%^&*()_++"6:<>?'],
  105:D        1                { ABOVE [FALSE,92] CHANGED TO '(' 5/9/84 SFB }
  106:D        1                { ABOVE [FALSE,93] CHANGED TO '' 5/10/84 SFB }
  107:D        1                { ABOVE [TRUE,93] CHANGED TO '"' 5/10/84 SFB }
  108:D        1                {uk}              t1['1234567890+''[]*\,.-',
  109:D        1                                    '!":$%&^()=?/(){@|;:_'],
  110:D        1                {italian}         t1[';E''{_I*5H)-Y$mK;:J',
  111:D        1                                    '12345678903+=&M%./!'],
  112:D        1                { ABOVE [TRUE,98] CHANGED TO '!' 5/8/84 SFB }
  113:D        1                { ABOVE [FALSE,87] CHANGED TO '*' 5/9/84 SFB }
  114:D        1                {french}          t1['&E"''(=I!5H)-*`mK;:=',
  115:D        1                                    '12345678903_+*M%./+'],
  116:D        1                { ABOVE [FALSE,92] CHANGED TO '*' 5/9/84 SFB }
  117:D        1                {german}          t1['1234567890^''0+NL,.-',
  118:D        1                                    '!"=$%&/()=?^[*ZX;:_'],
  119:D        1                {swedish}         t1['1234567890+ETONL,.-',
  120:D        1                                    '!"#$%&/()=?\P[ZX;:_']
```

```
121:D        1              ];
122:S
123:D        1 procedure doinit;
124:D        2 var
125:D    -2  2   i    : -32768..32767;
126:C        2 begin
127:C        2   if (kbdtype=ITFkbd) and
128:C        3      (kbdlang>=finish_kbd) and (kbdlang<=swedish_kbd) and
129:C        3      (langtable[0]<>nil)
130:C        3   then
131:C        3   if (langtable[0]^.langcode=us_kbd) then
132:C        4   begin
133:C        4     with langtable[0]^ do
134:C        5     begin
135:C        5       { extend char keys are extenstion bit }
136:C        5       transmode:=kshift_extc;
137:C        5       langcode :=kbdlang;
138:C        5       { load special alpha keys }
139:C        5       keytable[1].keys[false]:=lowkeys[kbdlang,false,1];
140:C        5       keytable[1].keys[true]:=lowkeys[kbdlang,true,1];
141:C        5       keytable[2].keys[false]:=lowkeys[kbdlang,false,2];
142:C        5       keytable[2].keys[true]:=lowkeys[kbdlang,true,2];
143:S
144:C        5       for i:=80 to 98 do
145:C        6       with keytable[i] do
146:C        7       begin
147:C        7         keys[false]:=specials[kbdlang,false,i];
148:C        7         keys[true]:=specials[kbdlang,true,i];
149:C        7       end;
150:S
151:C        5       case kbdlang of { language fixups }
152:C        6         finish_kbd,SWEDISH_KBD:          { ADDED SWEDISH 5/10/84 SFB }
153:C        6             for i:=91 to 95 do keytable[i].no_capslock:=false;
154:C        6             { CHANGED BOTTOM INDEX OF ABOVE LOOP TO 91 5/8/84 SFB }
155:C        6           italian_kbd,french_kbd,belgian_kbd:
156:C        6             begin
157:C        6               keytable[119].no_capslock:=true;
158:C        6               keytable[94].no_capslock:=false;
159:C        6               { exchange z-w and m<-,? }
160:C        6               keytable[119].keys[false]:=',';keytable[119].keys[true]:='?';
161:C        6               keytable[105]:=keytable[120]; { z<-w }
162:C        6               keytable[120].keys[false]:='w';keytable[120].keys[true]:='W';
163:C        6               { ABOVE CHANGED TO 'w', 'W' 5/8/84 SFB }
164:C        6               if kbdlang<>italian_kbd then
165:C        7               begin { exchange a-q }
166:C        7                 keytable[104]:=keytable[112]; { q<-a }
167:C        7                 keytable[112].keys[false]:='q';keytable[112].keys[true]:='Q';
168:C        7                 { ABOVE CHANGED TO [112].KEYS[TRUE] 5/8/84 SFB }
169:C        7               end
170:C        7               ELSE
171:C        7                KEYTABLE[87].KEYCLASS:=NONA_ALPHA_KEY;       { 5/9/84 SFB }
172:C        6             end;
173:C        6           cdn_eng_kbd,cdn_fr_kbd:
174:C        6             begin
175:C        6               keytable[93].no_capslock:=false;
176:C        6               keytable[98].no_capslock:=false;
177:C        6               KEYTABLE[95].KEYCLASS:=NONA_ALPHA_KEY;       { 5/9/84 SFB }
178:C        6             end;
179:C        6           norwegian_kbd,german_kbd,danish_kbd:    { REMOVED SWEDISH 5/10/84 SFB }
180:C        6             begin
```



```
181:C        6               keytable[92].no_capslock:=false;
182:C        6               keytable[94].no_capslock:=false;
183:C        6               keytable[95].no_capslock:=false;
184:C        6               if kbdlang=german_kbd then
185:C        7               begin { switch z-y }
186:C        7                 keytable[120]:=keytable[109]; { y<-z }
187:C        7                 keytable[109].keys[false]:='z';keytable[109].keys[true]:='Z';
188:C        7                 { ABOVE CHANGED TO [109].KEYS[TRUE] 5/8/84 SFB }
189:C        7               end;
190:C        6             end;
191:C        6           swiss_gr_kbd,swiss_fr_kbd:
192:C        6             begin { switch z-y }
193:C        6               keytable[120]:=keytable[109]; { y<-z }
194:C        6               keytable[109].keys[false]:='z';keytable[109].keys[true]:='Z';
195:C        6               { ABOVE CHANGED TO [109].KEYS[TRUE] 5/8/84 SFB }
196:C        6               KEYTABLE[93].KEYCLASS:=NONADV_KEY
197:C        6             end;
198:C        6           uk_kbd,dutch_kbd:; { no changes }
199:C        6           spanish_latin_kbd,spanish_eur_kbd: keytable[94].no_capslock:=false;
200:C        6           spanish_kbd,katakana_kbd:; { not implemented here }
201:C        6           otherwise
202:C        6         end;
203:C        5         langindex:=0;
204:C        5         kbdcapslock:=true;  kbdaltlock:=false;
205:C        5         IF KBDLANG IN                          { THESE MODS 5/9/84 SFB }
206:C        6          [NORWEGIAN_KBD,DANISH_KBD,SWISS_GR_KBD,SWISS_FR_KBD] THEN
207:C        6           KEYTABLE[91].KEYCLASS:=NONADV_KEY;
208:C        5         IF KBDLANG IN
209:C        6          [NORWEGIAN_KBD,DANISH_KBD] THEN
210:C        6           KEYTABLE[93].KEYCLASS:=NONA_ALPHA_KEY;
211:C        5         IF KBDLANG IN
212:C        6          [DUTCH_KBD,SPANISH_EUR_KBD] THEN
213:C        6           KEYTABLE[95].KEYCLASS:=NONADV_KEY;
214:C        5         IF KBDLANG = DUTCH_KBD THEN   { 3.01 BUG FIX 9/20/84 SFB }
215:C        6           KEYTABLE[92].KEYCLASS:=NONA_ALPHA_KEY;
216:C        5         IF KBDLANG = FRENCH_KBD THEN  { 3.01 BUG FIX 9/20/84 SFB }
217:C        6           KEYTABLE[92].KEYCLASS:=NONADV_KEY;
218:C        5         IF KBDLANG IN
219:C        6          [BELGIAN_KBD,SPANISH_LATIN_KBD,CDN_ENG_KBD,CDN_FR_KBD] THEN
220:C        6           KEYTABLE[92].KEYCLASS:=NONADV_KEY;
221:C        5         IF KBDLANG=SPANISH_EUR_KBD THEN
222:C        6          KEYTABLE[91].KEYCLASS:=NONA_ALPHA_KEY;
223:C        5       end; { with langtable[0]^ }
224:C        4     end;
225:C        2 end; { doinit }
226:C        1 begin
227:C        1   doinit;       { structure is to avoid global vars }
228:C        1 end.
229:S

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

PRINTER

Description

PRINTER is the transfer method (TM) for the system printer.

Requirements

IODECLARATIONS and whatever IOLIBRARY interface card driver that is applicable.

```
     1:S                                                    (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1984.
     4:S          All rights are reserved.  Copying or other
     5:S          reproduction of this program except for archival
     6:S          purposes is prohibited without the prior
     7:S          written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                       RESTRICTED RIGHTS LEGEND
    11:S
    12:S          Use, duplication, or disclosure by the Government
    13:S          is subject to restrictions as set forth in
    14:S          paragraph (b) (3) (B) of the Rights in Technical
    15:S          Data and Computer Software clause in
    16:S          DAR 7-104.9(a).
    17:S
    18:S          HEWLETT-PACKARD COMPANY
    19:D        0 Fort Collins, Colorado                          *)
    20:S
    21:S
    22:D        0 $modcal$
    23:D        0 $debug off, range off, ovflcheck off$
    24:D        0 $stackcheck off, iocheck off$
    25:S
    26:D        0 $search 'IOLIB:KERNEL','OSFS:SYSDEVS'$
    27:S
```



```
    28:D        0 $page$
    29:D        0 $copyright 'COPYRIGHT (C) 1984 BY HEWLETT-PACKARD COMPANY'$
    30:S
    31:S
    32:D        0 module prtdvr;
    33:S
    34:D        1 import
    35:D        1   sysglobals,
    36:D        1   iodeclarations,
    37:D        1   asm, sysdevs, mini, misc, fs;
    38:S
    39:D        1 export
    40:D        1   procedure prtio (fp: fibp; request: amrequesttype; anyvar buffer: window;
    41:D        2                                                      length, position: integer);
    42:D        1 implement  (prtdvr)
    43:S
    44:D        1 procedure bep;
    45:C        2   begin write(bellchar); end;
    46:S
    47:D        1 procedure prtio(fp: fibp; request: amrequesttype; anyvar buffer: window;
    48:D        2                                                   length, position: integer);
    49:S
    50:D        2 const
    51:D        2   uclr_timeout_const = 25;      {HPIB commands during unitclear}
    52:S
    53:D        2   repeating_timeout  = 333;     {timeout constant after initial timeout}
    54:D        2   timeouts_per_beep  = 40;      {beep period in repeating timeout units}
    55:S
    56:D        2   SDC       = 4;          {selective device clear}
    57:D        2   LAGbase   = 32;         {listen address group base}
    58:D        2   TAGbase   = 64;         {talk address group base}
    59:S
    60:D        2   linefeed = chr(10);    {ASCII linefeed}
    61:D        2   formfeed = chr(12);    {ASCII formfeed}
    62:D        2   return   = chr(13);    {ASCII carriage return}
    63:S
    64:D        2 var
    65:S
    66:D    -2  2   select_code: type_isc;
    67:D    -6  2   sc_table_entry_ptr: ^isc_table_type;
    68:D   -10  2   previous_char_ptr: charptr;
    69:D   -12  2   bus_address: byte;
    70:S
    71:D   -13  2   channel_is_setup: boolean;
    72:D   -14  2   writing_previous_char: boolean;
    73:D   -15  2   previously_timed_out: boolean;
    74:D   -16  2   timeout_blanked: boolean;
    75:D   -20  2   user_spec_timeout: integer;
    76:D   -24  2   current_timeout: integer;
    77:D   -26  2   timeout_counter: shortint;
    78:D   -70  2   saved_line     : string[42];          { 3.0 bug fix -- 4/12/84 }
    79:D   -71  2   line_needs_restoring : boolean;      { 4/12/84 }
    80:D   -76  2   buf: charptr;
    81:D   -77  2   saved_echo: boolean;                 { 5/9/84 }
    82:S
```

```
    83:D   -77  2 $page$
    84:S
    85:D        2 procedure reset_card_and_confirm_timeout;
    86:D        3   var
    87:D    -2  3     saved_escapecode: shortint;
    88:D    -6  3     saved_ioe_sc: integer;
    89:D   -10  3     saved_ioe_result: integer;
    90:C        3   begin {reset_card_and_confirm_timeout}
    91:C        3     saved_escapecode := escapecode;
    92:C        3     saved_ioe_sc := ioe_isc;
    93:C        3     saved_ioe_result := ioe_result;
    94:C        3     try
    95:C        4       with sc_table_entry_ptr^ do
    96:C        5         call(io_drv_ptr^.iod_init, io_tmp_ptr);
    97:C        4     recover
    98:C        4       if (escapecode<>ioescapecode) or (ioe_isc<>select_code) then
    99:C        5         escape(escapecode);
   100:C        3     ioe_isc := saved_ioe_sc;
   101:C        3     ioe_result := saved_ioe_result;
   102:C        3     if (saved_escapecode<>ioescapecode) or (ioe_isc<>select_code) then
   103:C        4       escape(saved_escapecode);
   104:C        3     if ioe_result<>ioe_timeout then
   105:C        4       ioresc(znodevice);
   106:C        3   end; {reset_card_and_confirm_timeout}
   107:S
   108:D        2 procedure clear_unit;
   109:D        3   procedure HPIBsdc;
   110:C        4     begin {HPIBsdc}
   111:C        4       with sc_table_entry_ptr^, io_drv_ptr^, io_tmp_ptr^ do
   112:C        5         begin
   113:C        5           call(iod_send, io_tmp_ptr, '?');
   114:C        5           timeout := uclr_timeout_const;
   115:C        5           call(iod_send, io_tmp_ptr, chr(LAGbase+bus_address));
   116:C        5           call(iod_send, io_tmp_ptr, chr(SDC));
   117:C        5         end; {with}
   118:C        4     end; {HPIBsdc}
   119:C        3   begin {clear_unit}
   120:C        3     with sc_table_entry_ptr^ do
   121:C        4       if card_type=hpib_card then
   122:C        5         try
   123:C        6           HPIBsdc;          {first attempt}
   124:C        6         recover
   125:C        6           begin
   126:C        6             reset_card_and_confirm_timeout;
   127:C        6             try
   128:C        7               HPIBsdc;        {second attempt}
   129:C        7             recover
   130:C        7               begin
   131:C        7                 reset_card_and_confirm_timeout;
   132:C        7                 ioresc(ztimeout);
   133:C        7               end; {recover}
   134:C        6           end {recover}
   135:C        6       else
   136:C        5         try
   137:C        6           call(io_drv_ptr^.iod_init, io_tmp_ptr);
   138:C        6         recover
   139:C        6           if (escapecode<>ioescapecode) or (ioe_isc<>select_code) then escape(escapecode)
   140:C        7           else ioresc(znodevice);
   141:C        3   end; {clear_unit}
   142:S
```



```
   143:D   -77  2 $page$
   144:S
   145:D        2 procedure wrtchar(character: char; last_char: boolean);
   146:S
   147:D        3   var
   148:D    -1  3     hs_successfully_initiated: boolean;
   149:D    -2  3     previous_hs_completed    : boolean;
   150:S
   151:S
   152:D        3   procedure restore_line;
   153:D    -1  4   var dummyc:char;
   154:C        4   begin
   155:C        4     if line_needs_restoring then { 4/12/84 }
   156:C        5     begin
   157:C        5       keybuffer^.echo:=saved_echo;
   158:C        5       keybufops(kdisplay,dummyc);
   159:C        5       line_needs_restoring:=false;
   160:C        5     end;
   161:C        4   end;
   162:S
```

```
163:D    -2  3 $page$
164:S
165:D        3   procedure inform_operator;
166:D   -44  4   var lmstr : string[42];                    { 3.0 bug fix -- 4/12/84 }
167:C        4     begin  {inform_operator}
168:S
169:C        4       if not previously_timed_out then
170:C        5       begin
171:C        5         timeout_blanked := true;
172:C        5         timeout_counter := 0;
173:C        5       end;
174:C        4       if not line_needs_restoring then
175:C        5       begin
176:C        5         saved_line := '* Printer timeout: fix or ';
177:C        5         if intlevel=0 then saved_line:=saved_line+'<stop> aborts *'
178:C        6                       else saved_line:=saved_line+'wait auto-abort*' ;
179:C        5                                              { 3.0 bug fix -- 4/12/84 }
180:C        5         line_needs_restoring := true;
181:C        5         if menustate=m_none then saved_echo:=keybuffer^.echo
182:C        6                             else saved_echo:=true;
183:C        5         menustate := m_none; { 4/12/84 }
184:C        5         keybuffer^.echo :=false;
185:C        5       end;
186:C        4       if timeout_blanked then lmstr:= saved_line
187:C        5                          else lmstr:= ' ';
188:C        4       CALL(CRTLLHOOK,CLLDISPLAY,LMSTR,' ');
189:C        4       timeout_blanked:= (timeout_counter mod 4)<>0;
190:S
191:C        4       if timeout_counter<=1 then bep;
192:C        4       timeout_counter := timeout_counter+1;
193:C        4       if timeout_counter>=timeouts_per_beep then
194:C        5         if intlevel=0 then timeout_counter := 0
195:C        6         else
196:C        6           begin
197:C        6             bep;
198:C        6             restore_line;
199:C        6             ioresc(ztimeout);
200:C        6           end; {else}
201:S
202:C        4     end;  {inform_operator}
203:S
```



```
204:D    -2  3 $page$
205:S
206:C        3   begin {wrtchar}
207:C        3     try
208:C        4       with sc_table_entry_ptr^, io_drv_ptr^, io_tmp_ptr^  do
209:C        5         repeat
210:C        6           try
211:C        7             previous_hs_completed := false;
212:C        7             if not channel_is_setup then
213:C        8               begin
214:C        8                 case card_type of
215:C        9                   hpib_card:
216:C        9                     begin
217:C        9                       call(iod_send, io_tmp_ptr, '?');
218:C        9                       previous_hs_completed := true;
219:C        9                       timeout := current_timeout;
220:C        9                       call(iod_send, io_tmp_ptr, chr(TAGbase+addressed));
221:C        9                       call(iod_send, io_tmp_ptr, chr(LAGbase+bus_address));
222:C        9                     end; {hpib_card}
223:C        9                   serial_card:
224:C        9                     if card_id=hp98626 then  {always set full duplex modem HS}
225:C       10                       call(iod_wtc, io_tmp_ptr, 13, 1);
226:C        9                   otherwise
227:C        9                     {do nothing};
228:C        9                 end; {case}
229:C        8                 channel_is_setup := true;
230:C        8               end; {if}
231:C        7             call(iod_wtb, io_tmp_ptr, character);
232:C        7             previous_char_ptr^ := character;
233:C        7             timeout := current_timeout;
234:C        7             if last_char then
235:C        8               if card_type=hpib_card then
236:C        9                 call(iod_send, io_tmp_ptr, '?');
237:C        7             if previously_timed_out then
238:C        8               if not writing_previous_char then
239:C        9                 begin
240:C        9                   restore_line;
241:C        9                   current_timeout := user_spec_timeout;
242:C        9                   previously_timed_out := false;
243:C        9                 end; {if}
244:C        7             hs_successfully_initiated := true;
245:C        7           recover
246:C        7             begin
247:C        7               reset_card_and_confirm_timeout;
248:C        7               channel_is_setup := false;
249:C        7               inform_operator;
250:C        7               previously_timed_out := true;
251:C        7               current_timeout := repeating_timeout;
252:C        7               if not (writing_previous_char or previous_hs_completed) then
253:C        8                 begin
254:C        8                   writing_previous_char := true;
255:C        8                   wrtchar(previous_char_ptr^, false);
256:C        8                   writing_previous_char := false;
257:C        8                 end; {if}
258:C        7               hs_successfully_initiated := false;
259:C        7             end; {recover}
260:C        6         until hs_successfully_initiated;
261:C        4     recover
262:C        4       begin
263:C        4         restore_line;   { 4/12/84 }
```

```
   264:C        4         escape(escapecode);
   265:C        4       end; {recover}
   266:C        3   end; {wrtchar}
```



```
   267:D   -77  2 $page$
   268:S
   269:C        2   begin  {prtio}
   270:C        2     ioresult := ord(ioerror);                                  { scs 1/17/83 }
   271:C        2     with unitable^[fp^.funit] do
   272:C        3       begin
   273:C        3         select_code := sc;
   274:C        3         sc_table_entry_ptr := addr(isc_table[select_code]);
   275:C        3         bus_address := ba;
   276:C        3         previous_char_ptr := addr(dvrtemp);
   277:C        3         user_spec_timeout := devid;  {user-specified in CTABLE}
   278:C        3       end; {with}
   279:S
   280:C        2     buf := addr(buffer);
   281:C        2     channel_is_setup := false;
   282:C        2     current_timeout := user_spec_timeout;
   283:C        2     previously_timed_out  := false;
   284:C        2     writing_previous_char := false;
   285:C        2     line_needs_restoring  := false; { 4/12/84 }
   286:S
   287:C        2     try
   288:C        3       with sc_table_entry_ptr^, io_tmp_ptr^  do
   289:C        4         begin
   290:C        4           if card_type=no_card then ioresc(znodevice);
   291:C        4           while (in_bufptr<>nil) or (out_bufptr<>nil) do {nothing};
   292:C        4         end; {with}
   293:S
   294:C        3       case request of
   295:C        4         flush:
   296:C        4           {do nothing};
   297:C        4         clearunit:
   298:C        4           clear_unit;
   299:C        4         writeeol:
   300:C        4           begin
   301:C        4             wrtchar(return, false);
   302:C        4             wrtchar(linefeed, true)
   303:C        4           end;
   304:C        4         writebytes:
   305:C        4           while length>0 do
   306:C        5             begin
   307:C        5               wrtchar(buf^, length=1);
   308:C        5               buf := addr(buf^, 1);
   309:C        5               length := length-1;
   310:C        5             end;
   311:C        4         otherwise
   312:C        4           ioresc(zbadmode);
   313:C        4       end; {case}
   314:C        3     recover
   315:C        3       if (escapecode=-20) and previously_timed_out then
   316:C        4         ioresult := ord(ztimeout)
   317:C        4       else if escapecode<>-10 then
   318:C        5         escape(escapecode);
   319:S
   320:C        2   end; {prtio}
   321:S
   322:C        1 end. {prtdvr}
   323:S
```

No errors. No warnings.
***** Nonstandard language features enablec *****

Description

RS_DRV contains low-level I/O drivers.

Usage

RS_DRV is used for the 98626 interface (and the Model 216/217's built-in RS-232 interface).

Requirements

RS and the I/O library kernel (IODECLARATIONS, etc.)

```
     1:S                                                    (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1984.
     4:S          All rights are reserved.  Copying or other
     5:S          reproduction of this program except for archival
     6:S          purposes is prohibited without the prior
     7:S          written consent of Hewlett-Packard Company.
     8:S
     9:S                        RESTRICTED RIGHTS LEGEND
    10:S
    11:S          Use, duplication, or disclosure by the Government
    12:S          is subject to restrictions as set forth in
    13:S          paragraph (b) (3) (B) of the Rights in Technical
    14:S          Data and Computer Software clause in
    15:S          DAR 7-104.9(a).
    16:S
    17:S          HEWLETT-PACKARD COMPANY
    18:S          Fort Collins, Colorado                          *)
    19:D        0
    20:S
    21:S
    22:D        0 $SEARCH 'IOLIB:KERNEL.CODE'$
    23:S
    24:D        0 $COPYRIGHT 'COPYRIGHT (C) 1984 BY HEWLETT-PACKARD COMPANY'$
    25:D        0 $MODCAL ON$
    26:D        0 $PARTIAL_EVAL ON$
    27:D        0 $STACKCHECK ON$
    28:D        0 $RANGE OFF$
    29:D        0 $DEBUG OFF$
    30:D        0 $OVFLCHECK OFF$
```



```
    31:D        0 $PAGE$
    32:S
    33:D        0 (*****************************************************************************)
    34:D        0 (*                                                                           *)
    35:D        0 (*       NOT RELEASED          VERSION 3.0                                   *)
    36:D        0 (*                                                                           *)
    37:D        0 (*****************************************************************************)
    38:D        0 (*                                                                           *)
    39:D        0 (*                                                                           *)
    40:D        0 (*       IOLIB             RS_DRIVERS                                        *)
    41:D        0 (*                                                                           *)
    42:D        0 (*                                                                           *)
    43:D        0 (*****************************************************************************)
    44:D        0 (*                                                                           *)
    45:D        0 (*                                                                           *)
    46:D        0 (*       library       -   IOLIB                                           *)
    47:D        0 (*       name          -   RS_DRIVERS                                      *)
    48:D        0 (*       module(s)     -   init_rs                                         *)
    49:D        0 (*                     -   rs                                              *)
    50:D        0 (*                                                                         *)
    51:D        0 (*       date          -   6 August 1982                                   *)
    52:D        0 (*       update        -   5 March  1984                                   *)
    53:D        0 (*       release       -   ??????????????????                              *)
    54:D        0 (*                                                                         *)
    55:D        0 (*       source        -   IOLIB:RS_DRV.TEXT                               *)
    56:D        0 (*       object        -   IOLIB:RS_DRV.CODE                               *)
    57:D        0 (*                                                                         *)
    58:D        0 (*****************************************************************************)
```

```
    59:D        0 $PAGE$
    60:D        0 (*************************************************************************)
    61:D        0 (*                                                                       *)
    62:D        0 (*                                                                       *)
    63:D        0 (*      BUG FIX HISTORY                                                  *)
    64:D        0 (*                                                                       *)
    65:D        0 (*   jws        ---------        io_init_rs    Find 98644 cards         *)
    66:D        0 (*              3/5/84                                                   *)
    67:D        0 (*                                                                       *)
    68:D        0 (*                                                                       *)
    69:D        0 (*                                                                       *)
    70:D        0 (*                                                                       *)
    71:D        0 (*                                                                       *)
    72:D        0 (*                                                                       *)
    73:D        0 (*                                                                       *)
    74:D        0 (*                                                                       *)
    75:D        0 (*                                                                       *)
    76:D        0 (*                                                                       *)
    77:D        0 (*                                                                       *)
    78:D        0 (*                                                                       *)
    79:D        0 (*************************************************************************)
```



```
    80:D        0 $PAGE$
    81:D        0 (*************************************************************************)
    82:D        0 (*                                                                       *)
    83:D        0 (*                                                                       *)
    84:D        0 (*      This is the source code for an external procedures library       *)
    85:D        0 (*      to be used for general purpose interfacing on the HP 9826.       *)
    86:D        0 (*                                                                       *)
    87:D        0 (*      The library consists of 3 primary sets of modules -              *)
    88:D        0 (*                                                                       *)
    89:D        0 (*              1.      KERNEL modules                                   *)
    90:D        0 (*              2.      driver modules                                   *)
    91:D        0 (*              3.      IOLIB  modules                                   *)
    92:D        0 (*                                                                       *)
    93:D        0 (*      The KERNEL modules consist of the following modules -            *)
    94:D        0 (*                                                                       *)
    95:D        0 (*              1.      iodeclarations  ( contains static r/w space )    *)
    96:D        0 (*              2.      iocomasm                                         *)
    97:D        0 (*              3.      general_0       ( initialization & low level     *)
    98:D        0 (*                                        routines like ioread/iowrite) *)
    99:D        0 (*      The KERNEL modules also have an executable program segement      *)
   100:D        0 (*      that gets executed at the time it is loaded.  This program       *)
   101:D        0 (*      initializes the static read/write memory.  This program also     *)
   102:D        0 (*      allocates the temporary storage for any card that exists -       *)
   103:D        0 (*      independent of whether there is or is not a driver for it.       *)
   104:D        0 (*                                                                       *)
   105:D        0 (*      The driver modules consist of the actual assembly or PASCAL      *)
   106:D        0 (*      routines that deal with a specific interface card.  There is     *)
   107:D        0 (*      also an executable program segment for each driver module.       *)
   108:D        0 (*      This program searches the select code table in the static r/w    *)
   109:D        0 (*      initialized by the KERNEL general_0 module for all select codes *)
   110:D        0 (*      that have the right interface card ( HPIB drivers will search    *)
   111:D        0 (*      for the 98624 interface ).  This program will then set up the    *)
   112:D        0 (*      driver tables to point to the correct drivers.                   *)
   113:D        0 (*                                                                       *)
   114:D        0 (*      The rest of the IOLIB modules are high-level modules that are    *)
   115:D        0 (*      used by an end user in his/her application program.              *)
   116:D        0 (*                                                                       *)
   117:D        0 (*      The KERNEL and some set of driver modules will exist in the      *)
   118:D        0 (*           INITLIB file as object code ( not EXPORT text ).  The       *)
   119:D        0 (*      export text will reside on the IO file.  The rest                *)
   120:D        0 (*      of the library will reside on the IO file.                       *)
   121:D        0 (*                                                                       *)
   122:D        0 (*************************************************************************)
```

```
123:D       0 $PAGE$
124:D       0 (*************************************************************************)
125:D       0 (*                                                                       *)
126:D       0 (*                                                                       *)
127:D       0 (*      REFERENCES :                                                     *)
128:D       0 (*                                                                       *)
129:D       0 (*                                                                       *)
130:D       0 (*      1.  9826 I/O Designers Guide            ( ----------- )         *)
131:D       0 (*                                                                       *)
132:D       0 (*      2.  68000 Manual                        ( Motorola )            *)
133:D       0 (*                                                                       *)
134:D       0 (*      3.  Pascal Language System Users Manual ( ---------- )          *)
135:D       0 (*                                                                       *)
136:D       0 (*      4.  Pascal Procdure Library Users Manual( ---------- )          *)
137:D       0 (*                                                                       *)
138:D       0 (*      5.  9826 card documentation             ( ----------- )         *)
139:D       0 (*                                                                       *)
140:D       0 (*      6.  Pascal I/O Library IRS              ( ------------ )        *)
141:D       0 (*                                                                       *)
142:D       0 (*      7.  98626A ERS                          ( ----------- )         *)
143:D       0 (*                                                                       *)
144:D       0 (*                                                                       *)
145:D       0 (*************************************************************************)
146:S
147:D       0 PROGRAM rs_initialize ( INPUT , OUTPUT );
```



```
148:D       1 $PAGE$
149:D       1 (*************************************************************************)
150:D       1 (*                                                                       *)
151:D       1 (*                                                                       *)
152:D       1 (*      RS-232 CARD DRIVERS                                              *)
153:D       1 (*                                                                       *)
154:D       1 (*                                                                       *)
155:D       1 (*************************************************************************)
156:D       1 EXTERNAL MODULE rs;
157:S                 { update  2/14/83
158:S
159:S                   purpose This module is a declaration of the importation text for
160:S                           the external drivers.
161:S
162:S                   note    The assembly language code that is imported needs to be
163:S                           called 'rs'.  The routines need to be called
164:S                           'rs_@@@@@@'
165:D       1           }
166:S
167:D       1 IMPORT  sysglobals, iodeclarations ;
168:S
169:D       1 EXPORT
170:S
171:S
172:D       1    PROCEDURE rs_init   ( temp : ANYPTR );
173:D       1    PROCEDURE rs_isr    ( temp : PISRIB );
174:D       1    PROCEDURE rs_rdb    ( temp : ANYPTR ;  VAR x : CHAR);
175:D       1    PROCEDURE rs_wtb    ( temp : ANYPTR ;  val   : CHAR);
176:D       1    PROCEDURE rs_rdw    ( temp : ANYPTR ;  VAR x : io_word);
177:D       1    PROCEDURE rs_wtw    ( temp : ANYPTR ;  val   : io_word);
178:D       1    PROCEDURE rs_rds    ( temp : ANYPTR ;  reg   : io_word;
179:D       2                                            VAR x : io_word);
180:D       1    PROCEDURE rs_wtc    ( temp : ANYPTR ;  reg   : io_word;
181:D       2                                            val   : io_word );
182:D       1    PROCEDURE rs_tfr    ( temp : ANYPTR ;  bcb   : ANYPTR );
183:S
184:S
185:S
186:S
187:D       1 END; { of rs }
```

```
   188:D        1 $PAGE$
   189:D        1 MODULE init_rs;
   190:S                  { update  3/5/84 jws
   191:S
   192:S                    purpose This module initializes the RS 232 drivers.
   193:S
   194:D        1         }
   195:S
   196:S
   197:D        1 IMPORT    iodeclarations ;
   198:S
   199:D        1 EXPORT
   200:S
   201:S
   202:D        1   VAR
   203:D  -120  1     rs_drivers    : drv_table_type;
   204:S
   205:D  -120  1   PROCEDURE io_init_rs;
   206:S
   207:S
   208:S
   209:D  -120  1 IMPLEMENT
   210:S
   211:S
   212:D  -120  1   IMPORT  sysglobals ,
   213:D  -120  1           isr ,
   214:D  -120  1           general_0 ,
   215:D  -120  1           rs ;
   216:S
   217:D        1   PROCEDURE io_init_rs;
   218:D        2   VAR
   219:D    -2  2     io_isc        : type_isc;
   220:D    -4  2     dummyword     : io_word;
   221:D    -6  2     io_lvl        : io_byte;
   222:C        2   BEGIN
   223:S
   224:C        2     io_revid := io_revid + ' R3.0';    {vers number changed JS 8/3/83 }
   225:S
   226:C        2     { set up the driver tables }
   227:S
   228:C        2     WITH rs_drivers DO BEGIN
   229:C        3       rs_drivers := dummy_drivers ;
   230:C        3       iod_init  := rs_init;
   231:C        3       iod_isr   := rs_isr;
   232:C        3       iod_rdb   := rs_rdb;
   233:C        3       iod_wtb   := rs_wtb;
   234:C        3       iod_rdw   := rs_rdw;
   235:C        3       iod_wtw   := rs_wtw;
   236:C        3       iod_rds   := rs_rds;
   237:C        3       iod_wtc   := rs_wtc;
   238:C        3       iod_tfr   := rs_tfr;
   239:C        3     END; { of WITH }
   240:S
   241:S
   242:C        2     { set up drivers for the interfaces }
   243:S
   244:C        2     FOR io_isc:=iominisc TO iomaxisc DO
   245:C        3       WITH isc_table[io_isc] DO BEGIN
   246:S
   247:C        4         IF (card_id = hp98626) or (card_id=hp98644)       { jws 3/5/84 }
```



```
   248:C        5           THEN BEGIN
   249:S
   250:C        5             io_drv_ptr:=ADDR(rs_drivers);
   251:S
   252:C        5             { if the card exists then link in an ISR for it }
   253:C        5             { ??? - what happens if an ISR fires during init }
   254:S
   255:C        5             io_lvl:=((ioread_byte(io_isc,3) DIV 16) MOD 4)+3;
   256:C        5             IF io_tmp_ptr^.myisrib.INTREGADDR <> NIL
   257:C        6               THEN BEGIN
   258:C        6                 { if isr exists then unlink it }
   259:C        6                 ISRUNLINK(io_lvl ,                        { level         }
   260:C        6                       ADDR(io_tmp_ptr^.myisrib));         { isrib info    }
   261:C        6               END; { of IF }
   262:S
   263:C        5             PERMISRLINK(io_drv_ptr^.iod_isr,              { isr           }
   264:C        5                   ANYPTR(INTEGER(card_ptr)+3),            { card address  }
   265:C        5                   192,                                    { intr. mask    }
   266:C        5                   192,                                    { intr. value   }
   267:C        5                   io_lvl,                                 { level         }
   268:C        5                   ADDR(io_tmp_ptr^.myisrib));             { isrib info    }
   269:C        5           END; { of IF card_type = hp98626 }
   270:S
   271:C        4     END; { of FOR io_isc WITH isc_table[io_isc] BEGIN }
   272:S
   273:S
   274:C        2     { call the actual driver initialization }
   275:S
   276:C        2     FOR io_isc:=iominisc TO iomaxisc DO
   277:C        3       WITH isc_table[io_isc] DO
   278:C        4       IF (card_id = hp98626) OR (card_id = hp98644)         { jws  3/5/84 }
   279:C        5         THEN BEGIN
   280:S
   281:C        5           CALL( io_drv_ptr^.iod_init, io_tmp_ptr );
   282:S
   283:C        5         END; { of WITH IF }
   284:S
   285:S
   286:C        2   END; { of io_init_rs }
   287:S
   288:C        1 END; { of MODULE init_rs }
```

```
   289:D        1 $PAGE$
   290:S
   291:D        1 IMPORT    init_rs ,
   292:D        1           LOADER ;
   293:S
   294:S
   295:C        1 BEGIN
   296:C        1   io_init_rs;
   297:C        1   MARKUSER;
   298:C        1 END. ( of rs_initialize )

No errors. No warnings.
             ***** Nonstandard language features enabled *****
```

SEGMENTER

Description

SEGMENTER provides for the loading, calling, and unloading of code segments.

Requirements

LOADER, SYSGLOBALS, ASM, LDR, MISC, and FS.

```
     1:D        0 $modcal$
     2:S
     3:D        0 {  SEGMENTER module --                                          }
     4:D        0 {                                                               }
     5:D        0 {                                                               }
     6:D        0 {                                                               }
     7:D        0 {                                                               }
     8:D        0 {                                                               }
     9:D        0 {                                                               }
    10:D        0 {                                                               }
    11:D        0 {                                                               }
    12:D        0 {                                                               }
    13:D        0 {                                                               }
    14:D        0 {                                                               }
    15:S
    16:D        0 module segmenter;
    17:S
    18:D        1 import loader, ldr, sysglobals, MISC;                 { JWS 5/24/83 }
    19:S
    20:D        1 export
    21:D        1   type segment_proc = procedure;
    22:D        1        proc_name    = string[120];
    23:S
    24:D        1   procedure init_segmenter(anyvar lowcode,   highcode,
    25:D        2                                  lowglobal, highglobal: byte);
    26:S
    27:D        1   procedure load_segment     (filename: fid);
    28:D        1   procedure load_heap_segment(filename: fid);
    29:D        1   procedure unload_segment;
    30:D        1   procedure unload_all;
    31:S
    32:D        1   procedure call_segment     (filename: fid);
    33:D        1   procedure call_segment_proc(filename: fid; symbol: proc_name);
    34:S
    35:D        1   function find_proc(symbol: proc_name): segment_proc;
    36:D        1   function exists_proc(p: segment_proc): boolean;
    37:S
    38:D        1   procedure segment_space(var code, global: integer);
    39:S
    40:D        1 implement
    41:S
    42:D        1 external module asm;
    43:D        1 export
    44:D        1   function  allocate(size: integer): anyptr;
    45:D        1   procedure newbytes(var p: anyptr; size: integer);
    46:D        1 end;
    47:S
    48:D        1 import asm;
    49:S
    50:D        1 type  proc = procedure;
    51:S
    52:D        1       trick_proc = record case boolean of
    53:D        1              true:  (p: proc);
    54:D        1              false: (ep, sl: integer);
    55:D        1              end;
    56:D        1
    57:D        1       state_ptr = ^state_rec;
    58:D        1       name_ptr  = ^proc_name;
    59:S
    60:D        1       state_rec = record
```



```
    61:D        1             savelist:  state_ptr;
    62:D        1             restore_heap: boolean;
    63:D        1             saveheap:  ^integer;
    64:D        1             saveglob,
    65:D        1             savecod:   integer;
    66:D        1             saveentry,
    67:D        1             savedef:   moddescptr;
    68:D        1             savefiles: anyptr;
    69:D        1             end;
    70:S
    71:D    -8  1 var highglob, lowglob:     integer;
    72:D   -16  1     lowcod,   highcod:     integer;
    73:D   -20  1     segment_list:          state_ptr;
    74:S
    75:D        1 procedure init_segmenter(anyvar lowcode,   highcode,
    76:D        2                                 lowglobal, highglobal: byte);
    77:C        2 begin
    78:C        2  lowcod := integer(addr(lowcode));
    79:C        2  highcod := integer(addr(highcode));
    80:C        2  if highcod < lowcod then begin
    81:C        3                           lowcod := integer(addr(highcode));
    82:C        3                           highcod := integer(addr(lowcode));
    83:C        3                           end;
    84:C        2  lowglob := integer(addr(lowglobal));
    85:C        2  highglob := integer(addr(highglobal));
    86:C        2  if highglob < lowglob then begin
    87:C        3                           lowglob := integer(addr(highglobal));
    88:C        3                           highglob := integer(addr(lowglobal));
    89:C        3                           end;
    90:C        2  if odd(lowcod)   then lowcod   := lowcod + 1;
    91:C        2  if odd(lowglob)  then lowglob  := lowglob + 1;
    92:C        2  if odd(highcod)  then highcod  := highcod - 1;
    93:C        2  if odd(highglob) then highglob := highglob - 1;
    94:C        2 end;
    95:S
    96:D        1 procedure segment_space(var code, global: integer);
    97:C        2 begin
    98:C        2   code   := highcod  - lowcod;
    99:C        2   global := highglob - lowglob;
   100:C        2 end;
   101:S
   102:C        1 procedure no_proc; begin escape(120); end;
   103:S
   104:D        1 function exists_proc(p: segment_proc): boolean;
   105:C        2 begin
   106:C        2   exists_proc := not(p = no_proc);
   107:C        2 end;
   108:S
   109:D  -122  1 function find_proc(symbol: proc_name): segment_proc;
   110:D  -122  2 var
   111:D  -126  2   modp: moddescptr;
   112:D  -134  2   ptr, valueptr: addrec;
   113:D  -135  2   found: boolean;
   114:D  -144  2   proc_rec: trick_proc;
   115:C        2 begin
   116:C        2 find_proc := no_proc;
   117:C        2 found := false;
   118:C        2 modp := sysdefs;
   119:C        2 while (modp<>nil) and not found do
   120:C        3   with modp^ do
```

```
   121:C        4     begin
   122:C        4       ptr := defaddr;
   123:C        4       while (ptr.a<defaddr.a+defsize) and not found do
   124:C        5         begin
   125:C        5           found := ptr.syp^=symbol;
   126:C        5           ptr.a := ptr.a+strlen(ptr.syp^)+1;
   127:C        5           ptr.a := ptr.a+ord(odd(ptr.a));
   128:C        5           valueptr.a := ptr.a+2;
   129:C        5           if found then with proc_rec do
   130:C        7             begin
   131:C        7             sl := 0;
   132:C        7             ep := valueptr.vep^.value;
   133:C        7             find_proc := p;
   134:C        7             end;
   135:C        5           ptr.a := ptr.a+ptr.gvp^.short;
   136:C        5         end; {while}
   137:C        4       modp := link;
   138:C        4     end; {with modp^}
   139:C        2 end;
   140:S
   141:D        1 procedure save_state(var state: state_rec; heapoff, codeoff: integer);
   142:C        2 begin with state do
   143:C        3   begin
   144:C        3         savelist   := segment_list;      segment_list := addr(state);
   145:C        3   mark(saveheap);                        restore_heap := true;
   146:C        3         saveheap   := addr(saveheap^, heapoff);
   147:C        3         saveglob   := highglob;
   148:C        3         savecod    := lowcod + codeoff;
   149:C        3         savedef    := sysdefs;
   150:C        3         saveentry  := entrypoint;
   151:C        3         savefiles  := openfileptr;       openfileptr := anyptr(-1);
   152:C        3   end;
   153:C        2 end;
   154:S
   155:D        1 procedure unload_segment;
   156:C        2 begin
   157:C        2 if segment_list = nil then escape(121) else
   158:C        3 with segment_list^ do
   159:C        4   begin
   160:C        4                 segment_list := savelist;
   161:C        4   if restore_heap then release(saveheap);
   162:C        4                 highglob     := saveglob;
   163:C        4                 lowcod       := savecod;
   164:C        4                 sysdefs      := savedef;
   165:C        4                 entrypoint   := saveentry;
   166:C        4                 openfileptr  := savefiles;
   167:C        4   end;
   168:C        2 end;
   169:S
   170:D        1 procedure unload_all;
   171:C        2 begin
   172:C        2 while segment_list <> nil do unload_segment;
   173:C        2 end;
   174:S
   175:D        1 procedure load_seg(var filename: fid; p: proc);
   176:D    -4  2 var space: integer;
   177:D    -8  2     modnum:    integer;
   178:D   -12  2     highheap0: addrec;
   179:D   -16  2     ESCTEMP: INTEGER;                { JWS 5/24/83 }
   180:D   -20  2     IOTEMP:  INTEGER;                { JWS 5/24/83 }
```



```
   181:S
   182:S
   183:C        2 begin
   184:C        2 LOADFIB.PHP:=NIL;                    { JWS 5/24/83 }
   185:C        2 try
   186:C        3   space := memavail - 3000;              {guess as to required stack space}
   187:C        3   if space <= 0 then escape(-2);
   188:C        3   mark(lowheap.p);
   189:C        3   highheap.a := lowheap.a + space; release(highheap.p);
   190:C        3   highheap0 := highheap;
   191:C        3   newmods := sysdefs;    endmod := sysdefs;
   192:S
   193:C        3   openlinkfile(filename);
   194:C        3   if fdirectory = nil then escape(-10);
   195:C        3   for modnum := 1 to fdirectory^[0].dnumfiles do
   196:C        4                         begin loadinfo(modnum, true, false); checkrev; end;
   197:C        3   allresolved := true; matchfile;
   198:S
   199:C        3   if not allresolved then escape(119);
   200:S
   201:C        3   highheap := highheap0;
   202:S
   203:C        3   countcode;
   204:S
   205:C        3   startglobal := highglob - a5;
   206:C        3   highglob := highglob - totalglobal;
   207:C        3   if highglob < lowglob then escape(117);
   208:C        3   zeromem(anyptr(highglob), totalglobal);
   209:C        3   call(p);
   210:C        3 recover begin
   211:C        3         LOCKUP;                      { JWS 5/24/83 }
   212:C        3         ESCTEMP:=ESCAPECODE;         { JWS 5/24/83 }
   213:C        3         IOTEMP:=IORESULT;            { JWS 5/24/83 }
   214:C        3         CLOSEFILES;                  { JWS 5/24/83 }
   215:C        3         IORESULT:=IOTEMP;            { JWS 5/24/83 }
   216:C        3         unload_segment;
   217:C        3         LOCKDOWN;                    { JWS 5/24/83 }
   218:C        3         escape(ESCTEMP);             { JWS 5/24/83 }
   219:C        3         end;
   220:C        2 end;
   221:S
   222:D        1 procedure release_heap;
   223:C        2 begin with segment_list^ do
   224:C        3   begin
   225:C        3   release(saveheap);
   226:C        3   restore_heap := false;
   227:C        3   end;
   228:C        2 end;
   229:S
   230:D        1 procedure local(var filename: fid; p: proc);
   231:D   -30  2 var state: state_rec;
   232:S
   233:D        2   procedure loadproc;
   234:C        3   begin
   235:C        3   highheap.a := highheap.a - totalreloc;
   236:C        3   if highheap.a < lowheap.a then escape(-2);
   237:C        3   release(highheap.p);
   238:C        3   startreloc := integer(allocate(totalreloc));
   239:C        3   loadtext(false);
   240:S
```

```
   241:C         3   highheap.a := highheap.a - totaldefs;
   242:C         3   if highheap.a < lowheap.a then escape(-2);
   243:C         3   release(highheap.p);
   244:C         3   movedefs(integer(allocate(totaldefs)));
   245:C         3   release_heap;
   246:C         3   call(p);
   247:C         3   end;
   248:S
   249:C         2 begin
   250:C         2 save_state(state, 0 , 0);
   251:C         2 load_seg(filename, loadproc);
   252:C         2 unload_segment;
   253:C         2 end;
   254:S
   255:D  -122   1 procedure call_segment     (filename: fid);
   256:S
   257:D         2  procedure callit;
   258:D    -8   3  var proc_rec: trick_proc;
   259:D   -12   3      modptr: moddescptr;
   260:C         3  begin
   261:C         3  modptr := entrypoint;
   262:C         3  while modptr<>nil do with modptr^ do
   263:C         5    begin
   264:C         5    if startaddr <> 0 then with proc_rec do
   265:C         7      begin
   266:C         7      sl := 0;
   267:C         7      ep := startaddr;
   268:C         7      call(p);
   269:C         7      end;
   270:C         5    if lastmodule then modptr := nil else modptr := link;
   271:C         5    end;
   272:C         3  end;
   273:S
   274:C         2 begin
   275:C         2  local(filename, callit);
   276:C         2 end;
   277:S
   278:D  -244   1 procedure call_segment_proc(filename: fid; symbol: proc_name);
   279:D  -252   2 var p: segment_proc;
   280:D  -252   2
   281:C         2  procedure callit; begin call(find_proc(symbol)) end;
   282:S
   283:C         2 begin
   284:C         2 p := find_proc(symbol);
   285:C         2 if exists_proc(p) then call(p)
   286:C         3                   else local(filename, callit);
   287:C         2 end;
   288:S
   289:D  -122   1 procedure load_segment     (filename: fid);
   290:D  -126   2 var state:  state_ptr;
   291:D  -126   2
   292:D         2   function code_space(size: integer): anyptr;
   293:C         3   begin
   294:C         3    if lowcod + size > highcod then escape(122)
   295:C         4    else
   296:C         4     begin
   297:C         4     code_space := anyptr(lowcod);
   298:C         4     lowcod := lowcod + size;
   299:C         4     end;
   300:C         3   end;
```



```
   301:S
   302:D         2   procedure loadproc;
   303:C         3   begin
   304:C         3   startreloc := integer(code_space(totalreloc));
   305:C         3   loadtext(false);
   306:C         3   movedefs(integer(code_space(totaldefs)));
   307:C         3   release_heap;
   308:C         3   end;
   309:S
   310:C         2 begin
   311:C         2 state := code_space(sizeof(state_rec));
   312:C         2 save_state(state^, 0, -sizeof(state_rec));
   313:C         2 load_seg(filename, loadproc);
   314:C         2 end;
   315:S
   316:D  -122   1 procedure load_heap_segment(filename: fid);
   317:D  -126   2 var state:  state_ptr;
   318:S
   319:D         2   procedure loadproc;
   320:C         3   begin
   321:C         3   loadtext(true);
   322:C         3   movedefs(startreloc+totalreloc);
   323:C         3   release(anyptr(startdefs+totaldefs));
   324:C         3   end;
   325:S
   326:C         2 begin
   327:C         2 newbytes(state, sizeof(state_rec));
   328:C         2 save_state(state^, -sizeof(state_rec), 0);
   329:C         2 mark(lowheap.p); startreloc := lowheap.a;
   330:C         2 load_seg(filename, loadproc);
   331:C         2 end;
   332:S
   333:C         1 end.

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

Description

SRMAM is the access method (transfer method) used to read and write files on the SRM.

Requirements

SRM_DRV and the I/O library drivers for the 98629 interface card (DATA_COMM).

```
    1:S                                                         (*
    2:S
    3:S           (c) Copyright Hewlett-Packard Company, 1983.
    4:S           All rights are reserved.  Copying or other
    5:S           reproduction of this program except for archival
    6:S           purposes is prohibited without the prior
    7:S           written consent of Hewlett-Packard Company.
    8:S
    9:S
   10:S                       RESTRICTED RIGHTS LEGEND
   11:S
   12:S           Use, duplication, or disclosure by the Government
   13:S           is subject to restrictions as set forth in
   14:S           paragraph (b) (3) (B) of the Rights in Technical
   15:S           Data and Computer Software clause in
   16:S           DAR 7-104.9(a).
   17:S
   18:S           HEWLETT-PACKARD COMPANY
   19:D         0 Fort Collins, Colorado                        *)
   20:S
   21:S
   22:D         0 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'$
   23:D         0 $range off$
   24:D         0 $debug off$
   25:D         0 $nodcal$
   26:S
   27:D         0 module srmammodule;
   28:S
   29:D         1 $search 'SRM_DRV',
   30:D         1         'IOLIB:KERNEL'$
   31:D         1 import
   32:D         1   sysglobals,
   33:D         1   misc,
   34:D         1   iodeclarations,
   35:D         1   srm;
   36:S
   37:D         1 export
   38:S
   39:D         1 procedure srmam(fp      : fibp;
   40:D         2                 request : amrequesttype;
   41:D         2           anyvar buffer : window;
   42:D         2                 bufsize : integer;
   43:D         2                 position: integer);
   44:S
   45:D         1 implement
   46:D         1 type
   47:D         1   pointer       = ^char;
   48:S
   49:D         1 (*************************************************************************)
   50:D         1 procedure srm_reac(anyvar f      : fib;
   51:D         2                    startoffset  : integer;
   52:D         2                    bytecount    : integer;
   53:D         2                    ramaddress   : charptr);
   54:D         2 var
   55:D    -4   2   sent  : integer;
   56:D    -8   2   access: integer;
   57:D   -12   2   offset: integer;
   58:C         2 begin
   59:C         2   if bytecount > 0 then
   60:C         3     with f, packet_ptr.rhead^ (, packet_ptr.rread^) do
```



```
   61:C         4       begin
   62:C         4         access := random_access;
   63:C         4         offset := startoffset;
   64:C         4         sent   := 0;
   65:C         4         if bytecount > 512 then
   66:C         5           begin
   67:C         5             sendreadpack(funit,fileid,access,512,offset,ramaddress);
   68:C         5             sent        := 512;
   69:C         5             ramaddress  := addr(ramaddress^,512);
   70:C         5             access      := sequential_access;
   71:C         5             offset      := 0;
   72:S
   73:C         5             while  (bytecount - sent > 512) and (ioresult = ord(inoerror)) do
   74:C         6               begin
   75:C         6                 sendreadpack(funit,fileid,access,512,offset,ramaddress);
   76:C         6                 sent           := sent + 512;
   77:C         6                 ramaddress     := addr(ramaddress^,512);
   78:C         6                 packetin(funit,req_read);
   79:C         6               end;
   80:C         5           end;
   81:S
   82:C         4         if ioresult = ord(inoerror) then
   83:C         5           begin
   84:C         5             sendreadpack(funit,fileid,access,bytecount-sent,offset,ramaddress);
   85:C         5             packetin(funit,req_read);
   86:C         5           end;
   87:S
   88:C         4         if (bytecount > 512) and (ioresult = ord(inoerror)) then
   89:C         5           packetin(funit,req_read);
   90:C         4
   91:C         4         if ioresult <> ord(inoerror) then
   92:C         5           resetcard(funit);
   93:C         4       end;
   94:C         2 end;
   95:S
   96:D         1 (*************************************************************************)
   97:D         1 procedure srm_write(anyvar f     : fib;
   98:D         2                     startoffset  : integer;
   99:D         2                     bytecount    : integer;
  100:D         2                     ramaddress   : charptr);
  101:D         2 var
  102:D    -4   2   sent  : integer;
  103:D    -8   2   access: integer;
  104:D   -12   2   offset: integer;
  105:C         2 begin
  106:C         2   if bytecount > 0 then
  107:C         3     with f, packet_ptr.rhead^ , packet_ptr.rwrite^ do
  108:C         4       begin
  109:C         4         access := random_access;
  110:C         4         offset := startoffset;
  111:C         4         sent   := 0;
  112:C         4         if bytecount > 512 then
  113:C         5           begin
  114:C         5             sendwritepack(funit,fileid,access,512,offset,ramaddress);
  115:C         5             sent        := 512;
  116:C         5             ramaddress  := addr(ramaddress^,512);
  117:C         5             access      := sequential_access;
  118:C         5             offset      := 0;
  119:S
  120:C         5             while (bytecount - sent > 512) and (ioresult = ord(inoerror)) do
```

```
   121:C        6                begin
   122:C        6                  sendwritepack(funit,fileid,access,512,offset,ramaddress);
   123:C        6                  sent           := sent + 512;
   124:C        6                  ramaddress     := addr(ramaddress^,512);
   125:C        6                  packetin(funit,req_write);
   126:C        6                end;
   127:C        5              end;
   128:S
   129:C        4            if ioresult = ord(inoerror) then
   130:C        5              begin
   131:C        5                sendwritepack(funit,fileid,access,bytecount-sent,offset,ramaddress);
   132:C        5                packetin(funit,req_write);
   133:C        5              end;
   134:S
   135:C        4            if (bytecount > 512) and (ioresult = ord(inoerror)) then
   136:C        5              packetin(funit,req_write);
   137:S
   138:C        4            if ioresult <> ord(inoerror) then
   139:C        5              resetcard(funit);
   140:C        4          end;
   141:C        2 end;
   142:S
   143:D        1 (*************************************************************************)
   144:D        1 procedure srm_clearunit(anyvar f    : fib);
   145:D        2 var
   146:D    -2  2   u             : unitnum;
   147:D    -3  2   keepworkdirs  : boolean;
   148:C        2 begin
   149:C        2   with f, unitable^[funit], isc_table[sc] do
   150:C        3     if (card_id <> hp98629) then  {this is not a theodore card}
   151:C        4       ioresult := ord(znodevice)
   152:C        4     else
   153:C        4       begin
   154:C        4         resetcard(funit);
   155:C        4         areyoualivepack(funit);
   156:C        4         if ioresult = ord(inoerror) then
   157:C        5           volpack(funit);
   158:C        4         if ioresult <> ord(inoerror) then
   159:C        5           ioresult := ord(znodevice)
   160:C        5         else
   161:C        5           begin
   162:C        5             keepworkdirs := false;
   163:C        5             if strlen(uvid) > 0 then
   164:C        6               keepworkdirs := true
   165:C        6             else
   166:C        6               for u := 1 to maxunit do
   167:C        7                 if sc = unitable^[u].sc then
   168:C        8                   if ba = unitable^[u].ba then
   169:C        9                     if strlen(unitable^[u].uvid) > 0 then
   170:C       10                       keepworkdirs := true;
   171:C        5             gangcleanpack(funit, keepworkdirs);
   172:C        5           end;
   173:C        4       end;
   174:C        2 end;
   175:S
   176:D        1 (*************************************************************************)
   177:D        1 procedure srmam(fp      : fibp;
   178:D        2                 request : amrequesttype;
   179:D        2           anyvar buffer  : window;
   180:D        2                 bufsize : integer;
```



```
   181:D        2                 position: integer);
   182:C        2 begin
   183:C        2   ioresult  := ord(inoerror);
   184:C        2   srmsavesc := 0;
   185:C        2   lockup;
   186:C        2   try
   187:C        3     with fp^ do
   188:C        4       if (request <> clearunit) and (unitable^[funit].offline) then
   189:C        5         ioresult        := ord(znodevice)
   190:C        5       else
   191:C        5         case request of
   192:C        6           readbytes   :begin
   193:C        6                          if flockable and (not flocked) then
   194:C        7                            ioresult := ord(ifileunlocked)
   195:C        7                          else
   196:C        7                            srm_read(fp^,position,bufsize,addr(buffer));
   197:C        6                        end;
   198:S
   199:C        6           writebytes  :begin
   200:C        6                          if flockable and (not flocked) then
   201:C        7                            ioresult := ord(ifileunlocked)
   202:C        7                          else
   203:C        7                            srm_write(fp^,position,bufsize,addr(buffer));
   204:C        6                        end;
   205:S
   206:C        6           clearunit   :begin
   207:C        6                          srm_clearunit(fp^);
   208:C        6                        end;
   209:S
   210:C        6           flush       : {do nothing, but no error};
   211:S
   212:C        6           otherwise   ioresult := ord(ibadrequest);
   213:C        6         end;
   214:C        3     if ioresult = ord(isrmcatchall) then
   215:C        4       if srmsavesc <> 0 then
   216:C        5         escape(srmsavesc);
   217:C        3   recover
   218:C        3     begin
   219:C        3       if escapecode = ioescapecode then
   220:C        4         ioresult  := ord(isrmcatchall)
   221:C        4       else
   222:C        4         begin
   223:C        4           lockdown;
   224:C        4           escape(escapecode);
   225:C        4         end;
   226:C        3     end;
   227:C        2   lockdown;
   228:C        2 end; {srmammodule}
   229:S
   230:C        1 end.

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

SRMDAM

Description

SRMDAM is the Directory Access Method for the SRM.

Requirements

SRM_DRV and the I/O library drivers for the 98629 interface card (DATA_COMM).

Notes

The DAM operates by making requests to the SRM to which it is connected. It does not understand the layout of an SRM disc.

```
    1:S                                                     (*
    2:S
    3:S          (c) Copyright Hewlett-Packard Company, 1983.
    4:S         All rights are reserved.  Copying or other
    5:S         reproduction of this program except for archival
    6:S         purposes is prohibited without the prior
    7:S         written consent of Hewlett-Packard Company.
    8:S
    9:S
   10:S                        RESTRICTED RIGHTS LEGEND
   11:S
   12:S         Use, duplication, or disclosure by the Government
   13:S         is subject to restrictions as set forth in
   14:S         paragraph (b) (3) (B) of the Rights in Technical
   15:S         Data and Computer Software clause in
   16:S         DAR 7-104.9(a).
   17:S
   18:S         HEWLETT-PACKARD COMPANY
   19:D       0 Fort Collins, Colorado                        *)
   20:S
   21:S
   22:D       0 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'$
   23:D       0 $range off$
   24:D       0 $debug off$
   25:D       0 $modcal$
   26:D       0 program init_srm;
   27:S
   28:D       1 module srmdammodule;
   29:S
   30:D       1 $search 'SRM_DRV',
   31:D       1         'IOLIB:KERNEL'$
   32:D       1 import
   33:D       1   sysglobals,
   34:D       1   misc,
   35:D       1   bootdammodule,
   36:D       1   iodeclarations,
   37:D       1   srm;
   38:S
   39:D       1 export
   40:S
   41:D       1 procedure srmdaminit;
   42:S
   43:D       1 procedure srmdam(anyvar f       : fib;
   44:D       2                  unum    : unitnum;
   45:D       2                  request : damrequesttype);
   46:S
   47:D       1 implement
   48:S
   49:D       1 type
   50:D       1   passarray     = array[1..8] of passentry;
   51:D       1 const
   52:D       1   extentsize    = 8*512;   {arbitrary choice -- multiple of common block sizes}
   53:D       1   constpassarray= passarray[
   54:D       1                             passentry[pbits:hex('80000000'),pword:'MANAGER'   ],
   55:D       1                             passentry[pbits:hex('40000000'),pword:'READ'      ],
   56:D       1                             passentry[pbits:hex('20000000'),pword:'WRITE'     ],
   57:D       1                             passentry[pbits:hex('10000000'),pword:'SEARCH'    ],
   58:D       1                             passentry[pbits:hex('08000000'),pword:'PURGELINK' ],
   59:D       1                             passentry[pbits:hex('04000000'),pword:'CREATELINK'],
   60:D       1                             passentry[pbits:hex('FFFFFFFF'),pword:'ALL'       ],
```



```
   61:D       1                             passentry[pbits:hex('00000000'),pword:'NONE'      ]
   62:D       1                             ];
   63:D       1   allcapabilities = access_capabilities[32 of true];
   64:D       1   nocapabilities  = access_capabilities[32 of false];
   65:D       1   temp_file_pass  = '>TEMP_FILE_PASS<';          {password on temporary files}
   66:D       1   BDATTYPE       = -5791;
   67:D       1   BASICBINTYPE   = -5775;
   68:D       1   BASICPROGTYPE  = -5808;
   69:D       1   SYSTMTYPE      = -5822;
   70:D       1   DATATYPE       = -5622;
   71:D       1   CODETYPE       = -5582;
   72:D       1   TEXTTYPE       = -5570;
   73:S
   74:D       1 var
   75:D    -4 1   passwordarrayptr      : ^passarray;
   76:D    -6 1   tempcounter           : shortint;
   77:S
   78:D    -6 1 (*************************************************************************)
   79:D       1 procedure setioresult(result    : integer);
   80:C       2 begin
   81:C       2   if ioresult = ord(inoerror) then
   82:C       3     ioresult := result;
   83:C       2 end;
   84:S
   85:D    -6 1 (*************************************************************************)
   86:D       1 function mapfkind(ftype : gang_file_codes) : filekind;
   87:D       2 var
   88:D    -2 2   fk: filekind;
   89:C       2 begin
   90:C       2   mapfkind      := datafile;
   91:C       2   for fk := lastfkind downto untypedfile do
   92:C       3     if efttable^[fk] = ftype.si2 then
   93:C       4       mapfkind := fk;
   94:C       2 end;
   95:S
   96:D    -6 1 (*************************************************************************)
   97:D       1 procedure paoc16tostr(anyvar paoc       : name_type;
   98:D       2                       anyvar strng      : string255);
   99:D       2 var
  100:D    -2 2   i     : shortint;
  101:C       2 begin
  102:C       2   i     := sizeof(paoc);
  103:C       2   while (paoc[i] = ' ') and (i > 0) do
  104:C       3     i   := i - 1;
  105:C       2   setstrlen(strng,0);
  106:C       2   strmove(i,paoc,1,strng,1);
  107:C       2 end;
  108:S
  109:D    -6 1 (*************************************************************************)
  110:D       1 procedure strtopaoc16(anyvar strng      : string255;
  111:D       2                       anyvar paoc       : name_type);
  112:C       2 begin
  113:C       2   paoc  := ' ';
  114:C       2   if strlen(strng) < 17 then
  115:C       3     strmove(strlen(strng),strng,1,paoc,1);
  116:C       2 end;
  117:S
  118:D    -6 1 (*************************************************************************)
  119:D       1 procedure setup_fns(var f       : fib;
  120:D       2                     anyvar fns  : file_name_set);
```

```
   121:C        2 begin
   122:C        2   with f, fns do
   123:C        3     if strlen(ftid) = 0 then
   124:C        4       setioresult(ord(ibadtitle))
   125:C        4     else
   126:C        4       begin
   127:C        4         strtopaoc16(ftid,file_name);
   128:C        4         strtopaoc16(ffpw,password)
   129:C        4       end;
   130:C        2 end;
   131:S
   132:D    -6  1 (*************************************************************************)
   133:D        1 procedure setup_fns3(var f      : fib;
   134:D        2                      anyvar nsa    : name_set_array_three);
   135:D        2 var
   136:D    -4  2   n             : integer;
   137:D    -8  2   tempioresult  : integer;
   138:D   -26  2   tempstr       : string[16];
   139:C        2 begin
   140:C        2   with f do
   141:C        3     if (not fisnew) or fnosrmtemp then
   142:C        4       setup_fns(f,nsa)
   143:C        4     else
   144:C        4       if (strlen(ftid) = 0) and (not faronymous) then
   145:C        5         setioresult(ord(ibadtitle))
   146:C        5       else
   147:C        5         begin
   148:C        5           with nsa[1] do
   149:C        6             begin
   150:C        6               password  := ' ';
   151:C        6               file_name := 'WORKSTATIONS';
   152:C        6             end;
   153:C        5           with nsa[2] do
   154:C        6             begin
   155:C        6               password  := ' ';
   156:C        6               file_name := 'TEMP_FILES';
   157:C        6             end;
   158:C        5           with nsa[3] do
   159:C        6             begin
   160:C        6               password          := temp_file_pass;
   161:C        6               setstrlen(tempstr,0);
   162:C        6               tempioresult      := ioresult;
   163:C        6               strwrite(tempstr,1,n,srmnode(unitable^[funit].sc),'_',fanonctr:1);
   164:C        6               ioresult          := tempioresult;
   165:C        6               strtopaoc16(tempstr,file_name);
   166:C        6             end;
   167:C        5         end;
   168:C        2 end;
   169:S
   170:D    -6  1 (*************************************************************************)
   171:D        1 procedure check_protectcode_set_array(    nps   : integer;
   172:D        2                                       var psa   : protectcode_set_array);
   173:S          {
   174:S          9-May-1983 RAM
   175:S          This routine has been added to check for right angle brackets ('>') in
   176:S          passwords.  If any are found, ioresult is set to ord(ibadpass).  This
   177:S          is because the parsing routines normally used with file opens terminate
   178:S          passwords at the first '>', therefore it is not possible to use them in
   179:S          passwords in normal operation.  If they are really desired, they can still be
   180:S          created by calling the lower level packet routines directly.
```



```
   181:S          Note that temporary files still have an "illegal" password.
   182:S          This routine is called from srm_create_file and from srm_set_pass.
   183:D        2 }
   184:D    -4  2 var     n       : integer;
   185:D    -8  2         i       : integer;
   186:C        2 begin
   187:C        2   for n := 1 to nps do
   188:C        3     for i := 1 to name_type_len do
   189:C        4       if psa[n].password[i] = '>' then
   190:C        5         setioresult(ord(ibadpass));
   191:C        2 end;
   192:S
   193:D    -6  1 (*************************************************************************)
   194:D        1 procedure parseoptparm(    foptstring   : string255ptr;
   195:D        2                        var sharemode    : integer;
   196:D        2                        var lockable     : boolean;
   197:D        2                        var nps          : integer;
   198:D        2                        var psa          : protectcode_set_array;
   199:D        2                            modeonly     : boolean);
   200:D        2 type
   201:D        2   tokentype     = (none,mode,pass,cap);
   202:D        2   statetype     = (needmodeorpass,needpass,needcap);
   203:D        2   acstrarrtype  = array [ac_manager .. ac_createlink] of string[10];
   204:D        2 const
   205:D        2   tokenlen      = 16;
   206:D        2   acstrarray    = acstrarrtype['MANAGER','READ','WRITE',
   207:D        2                               'SEARCH','PURGELINK','CREATELINK'];
   208:D        2 var
   209:D    -2  2   typeoftoken   : tokentype;
   210:D    -4  2   state         : statetype;
   211:D    -8  2   sindx         : integer;
   212:D    -9  2   delim         : char;
   213:D   -12  2   ac            : ac_manager..ac_createlink;
   214:D   -30  2   token         : string[16];
   215:D   -31  2   ok            : boolean;
   216:S
   217:D        2 procedure getuntildelim(del1    : char;
   218:D        3                         del2    : char);
   219:D        3 var
   220:D    -4  3   startindx     : integer;
   221:C        3 begin
   222:C        3   delim         := chr(0);
   223:C        3   startindx     := sindx;
   224:C        3   while (sindx <= strlen(foptstring^)) and (delim = chr(0)) do
   225:C        4     if  (foptstring^[sindx] <> del1)
   226:C        5     and (foptstring^[sindx] <> del2) then
   227:C        5       sindx     := sindx + 1
   228:C        5     else
   229:C        5       delim     := foptstring^[sindx];
   230:C        3   if (sindx - startindx) <= tokenlen then
   231:C        4     token := str(foptstring^,startindx,sindx - startindx)
   232:C        4   else
   233:C        4     setioresult(ord(ibadvalue));
   234:C        3   if sindx <= strlen(foptstring^) then
   235:C        4     sindx     := sindx + 1;
   236:C        3 end;
   237:S
   238:C        2 begin   {parseoptparm}
   239:C        2   sharemode     := exclusive_share_code;
   240:C        2   lockable      := false;
```

```
241:C        2   state          := needmodeorpass;
242:C        2   nps            := 0;
243:C        2   sindx          := 1;
244:C        2   if foptstring <> nil then
245:C        3     while (sindx <= strlen(foptstring^)) and (ioresult = ord(inoerror)) do
246:C        4       begin
247:C        4         case state of
248:C        5           needmodeorpass:
249:C        5                          begin
250:C        5                            getuntildelim(',', ':');
251:C        5                            if delim = ':' then
252:C        6                              begin
253:C        6                                typeoftoken     := pass;
254:C        6                                state           := needcap;
255:C        6                              end
256:C        6                            else
257:C        6                              begin
258:C        6                                typeoftoken     := mode;
259:C        6                                state           := needpass;
260:C        6                              end;
261:C        5                            if modeonly then
262:C        6                              sindx := strlen(foptstring^) + 1;
263:C        5                          end;
264:C        5           needpass     : begin
265:C        5                            getuntildelim(':', ':');
266:C        5                            if delim = ':' then
267:C        6                              begin
268:C        6                                typeoftoken     := pass;
269:C        6                                state           := needcap;
270:C        6                              end
271:C        6                            else
272:C        6                              setioresult(ord(ibadvalue));
273:C        5                          end;
274:C        5           needcap      : begin
275:C        5                            getuntildelim(',', ';');
276:C        5                            typeoftoken       := cap;
277:C        5                            if delim = ',' then
278:C        6                              state     := needcap
279:C        6                            else if delim = ';' then
280:C        7                              state     := needpass;
281:C        5                          end;
282:C        5         end;    {case}
283:C        4         if ioresult = ord(inoerror) then
284:C        5           case typeoftoken of
285:C        6             mode  : begin
286:C        6                       upc(token);
287:C        6                       if token = 'EXCLUSIVE' then
288:C        7                         sharemode := exclusive_share_code
289:C        7                       else if token = 'SHARED' then
290:C        8                         sharemode := shared_share_code
291:C        8                       else if token = 'LOCKABLE' then
292:C        9                         begin
293:C        9                           sharemode := shared_share_code;
294:C        9                           lockable  := true;
295:C        9                         end
296:C        9                       else
297:C        9                       setioresult(ord(ibadvalue));
298:C        6                     end;
299:C        6             pass  : begin
300:C        6                       nps := nps + 1;
```



```
301:C        6                       with psa[nps] do
302:C        7                         begin
303:C        7                           strtopaoc16(token,password);
304:C        7                           capabilities := nocapabilities;
305:C        7                         end;
306:C        6                     end;
307:C        6             cap   : begin
308:C        6                       upc(token);
309:C        6                       ok := false;
310:C        6                       with psa[nps] do
311:C        7                         if token = 'ALL' then
312:C        8                           begin
313:C        8                             capabilities := allcapabilities;
314:C        8                             ok := true;
315:C        8                           end
316:C        8                         else
317:C        8                           for ac := ac_manager to ac_createlink do
318:C        9                             if token = acstrarray[ac] then
319:C       10                               begin
320:C       10                                 capabilities[ac] := true;
321:C       10                                 ok := true;
322:C       10                               end;
323:C        6                       if not ok then
324:C        7                         setioresult(ord(ibadvalue));
325:C        6                     end;
326:C        6             end;
327:C        4         end;
328:C        2 end;
329:S
330:D    -6  1 (*****************************************************************************)
331:D        1 procedure srm_close_fileid(unum          : unitnum;
332:D        2                            var fileid    : integer);
333:C        2 begin
334:C        2   if fileid = 0 then
335:C        3     fileid    := -1
336:C        3   else
337:C        3     if (fileid > 0) and (fileid <> unitable^[unum].dvrtemp) then
338:C        4       with packet_ptr.rhead^ do
339:C        5         begin
340:C        5           closepack(unum,fileid);
341:C        5           if status = 0 then
342:C        6             fileid := -1;
343:C        5         end;
344:C        2 end;
345:S
346:D    -6  1 (*****************************************************************************)
347:D        1 procedure srm_close_pathid(unum          : unitnum;
348:D        2                            var pathid    : integer;
349:D        2                            savepathid    : boolean);
350:C        2 begin
351:C        2   if not savepathid then
352:C        3     srm_close_fileid(unum,pathid);
353:C        2 end;
354:S
355:D    -6  1 (*****************************************************************************)
356:D        1 procedure translatedate(var srmdate      : date_type;
357:D        2                         var systemdate   : daterec;
358:D        2                         var systemtime   : timerec);
359:D        2 var
360:D    -4  2   time          : integer;
```

```
   361:C        2 begin
   362:C        2   with srmdate do
   363:C        3     begin
   364:C        3       with systemdate do
   365:C        4         begin
   366:C        4           month          := date.month;
   367:C        4           day            := date.day;
   368:C        4           year           := date.year;
   369:C        4         end;
   370:C        3       with systemtime do
   371:C        4         begin
   372:C        4           time           := seconds_since_midnight;
   373:C        4           hour           := time div 3600;
   374:C        4           minute         := (time-(hour*3600)) div 60;
   375:C        4           centisecond    := (time mod 60) * 100;
   376:C        4         end;
   377:C        3     end;
   378:C        2 end;
   379:S
   380:D    -6  1 (*****************************************************************************)
   381:D        1 procedure srm_get_dir_info(anyvar dircatentry   : catentry;
   382:D        2                                   var dirid     : integer;
   383:D        2                                       unum      : unitnum;
   384:D        2                                       long      : boolean;
   385:D        2                                       dir_is_dvrtemp: boolean);
   386:D        2 const
   387:D        2   zerodate      = daterec[year:0,day:0,month:0];
   388:D        2   zerotime      = timerec[hour:0,minute:0,centisecond:0];
   389:D        2 var
   390:D    -4  2   n             : integer;
   391:D    -8  2   tempioresult  : integer;
   392:C        2 begin
   393:C        2   with dircatentry, unitable^[unum] do
   394:C        3     begin
   395:C        3       setstrlen(cname,0);
   396:C        3       volpack(unum);
   397:C        3       with packet_ptr.rvol^, packet_ptr.rhead^ do
   398:C        4         begin
   399:C        4           if status = 0 then
   400:C        5             if not exist.value then
   401:C        6               setioresult(ord(ilostunit))  {set ioresult to no volume}
   402:C        6             else
   403:C        6               begin
   404:C        6                 paoc16tostr(volume_name,cname);
   405:C        6                 cextra1          := -1;  (max_file_size div 32)
   406:C        6                 cpsize           := -1;
   407:C        6                 clsize           := -1;
   408:C        6                 cextra2          := interleave;
   409:C        6                 cstart           := -1;
   410:C        6                 cblocksize       := 1;
   411:C        6                 ccreatedate      := zerodate;
   412:C        6                 ccreatetime      := zerotime;
   413:C        6                 clastdate        := zerodate;
   414:C        6                 clasttime        := zerotime;
   415:C        6                 setstrlen(cinfo,0);
   416:C        6                 tempioresult     := ioresult;
   417:C        6                 strwrite(cinfo,1,n,'SRM  ',sc:1,',',ba:1,',',du:1);
   418:C        6                 ioresult         := tempioresult;
   419:C        6                 if dirid > 0 then
   420:C        7                   begin
```



```
   421:C        7                     fileinfopack(unum,dirid);
   422:C        7                     with packet_ptr.rfileinfo^, file_info do
   423:C        8                       if status <> 0 then
   424:C        9                         begin
   425:C        9                           ioresult := tempioresult;
   426:C        9                           if dir_is_dvrtemp then
   427:C       10                             dirid := 0;
   428:C        9                         end
   429:C        9                       else
   430:C        9                         begin
   431:C        9                           if file_name <> ' ' then
   432:C       10                             paoc16tostr(file_name,cname);
   433:C        9                           if long then
   434:C       10                             begin
   435:C       10                               translatedate(creation_date,ccreatedate,ccreatetime);
   436:C       10                               translatedate(last_access_date,clastdate,clasttime);
   437:C       10                             end;
   438:C        9                         end;
   439:C        7                   end;
   440:C        6               end;
   441:C        4           end;
   442:C        3       end;
   443:C        2 end;
   444:S
   445:D    -6  1 (*****************************************************************************)
   446:D        1 procedure srm_get_vol_name(anyvar f     : vid;
   447:D        2                                unum  : unitnum);
   448:D        2 var
   449:D   -80  2   dircatentry   : catentry;
   450:C        2 begin
   451:C        2   srm_get_dir_info(dircatentry,unitable^[unum].dvrtemp,unum,false,true);
   452:C        2   f := dircatentry.cname;
   453:C        2 end;
   454:S
   455:D    -6  1 (*****************************************************************************)
   456:D        1 procedure srm_set_pass(anyvar f         : fib;
   457:D        2                            unum      : unitnum);
   458:D        2 type
   459:D        2   catarray      = array[0..maxint] of passentry;
   460:D        2   catarrayptr   = ^catarray;
   461:D        2 var
   462:D   -16  2   volpass       : name_type;
   463:D   -52  2   fns           : file_name_set;
   464:D   -56  2   nps           : integer;
   465:D   -57  2   done          : boolean;
   466:D   -62  2   i             : integer;
   467:D   -66  2   j             : integer;
   468:D   -70  2   catindx       : integer;
   469:D   -74  2   catentryindx  : integer;
   470:D   -74  2   tempcapbits   : record case boolean of
   471:D   -74  2                     true  : (i  : integer);
   472:D   -74  2                     false : (b  : access_capabilities);
   473:D   -78  2                   end;
   474:D  -654  2   psa           : protectcode_set_array;
   475:C        2 begin
   476:C        2   with f do
   477:C        3     begin
   478:C        3       setup_fns(f,fns);
   479:C        3       if ioresult = ord(inoerror) then
   480:C        4         begin
```

```
481:C        4           for i := 1 to fpeof do
482:C        5             with psa[i], catarrayptr(fwindow)^[i-1] do
483:C        6               begin
484:C        6                 strtopaoc16(pword,password);
485:C        6                 tempcapbits.i   := pbits;
486:C        6                 capabilities    := tempcapbits.b;
487:C        6                 nps             := i;
488:C        6               end;
489:C        4           strtopaoc16(fvid,volpass);
490:C        4           check_protectcode_set_array(nps,psa);
491:C        4           if ioresult = ord(inoerror) then
492:C        5             changeprotectpack(unum,1,addr(fns),start_alternate,pathid,
493:C        5                               volpass,nps,addr(psa));
494:C        4         end;
495:C        3     end;
496:C        2 end;
497:S
498:D    -6  1 (*************************************************************************)
499:D        1 procedure srm_cat_pass(anyvar f         : fib;
500:D        2                        unum      : unitnum);
501:D        2 type
502:D        2   catarray      = array[0..maxint] of passentry;
503:D        2   catarrayptr   = ^catarray;
504:D        2 var
505:D   -16  2   volpass       : name_type;
506:D   -52  2   fns           : file_name_set;
507:D   -53  2   done          : boolean;
508:D   -58  2   i             : integer;
509:D   -62  2   j             : integer;
510:D   -66  2   catindx       : integer;
511:D   -70  2   catentryindx  : integer;
512:D   -70  2   tempcapbits   : record case boolean of
513:D   -70  2                    true  : (i  : integer);
514:D   -70  2                    false : (b  : access_capabilities);
515:D   -74  2                  end;
516:C        2 begin
517:C        2   catentryindx  := 0;
518:C        2   done  := false;
519:C        2   with f, packet_ptr.rcatpass^ do
520:C        3     begin
521:C        3       setup_fns(f, fns);
522:C        3       if ioresult = ord(inoerror) then
523:C        4         begin
524:C        4           strtopaoc16(fvid,volpass);
525:C        4           foptstring    := anyptr(passwordarrayptr);
526:C        4           catindx   := fpos + 1;
527:C        4           while (catentryindx < fpeof) and (not done) and (ioresult = ord(inoerror)) do
528:C        5             begin
529:C        5               catpasspack(unum,1,addr(fns),start_alternate,
530:C        5                           pathid,volpass,24,catindx);
531:C        5               if ioresult = ord(inoerror) then
532:C        6                 begin
533:C        6                   i  := 1;
534:C        6                   if actual_num_passwords < 24 then
535:C        7                     done      := true;
536:C        6                   while i <= actual_num_passwords do
537:C        7                     if catentryindx < fpeof then
538:C        8                       begin
539:C        8                         with password_info[i], catarrayptr(fwindow)^[catentryindx] do
540:C        9                           begin
```



```
541:C        9                             paoc16tostr(password,pword);
542:C        9                             tempcapbits.b   := capabilities;
543:C        9                             pbits           := tempcapbits.i;
544:C        9                           end;
545:C        8                       i             := i + 1;
546:C        8                       catentryindx  := catentryindx + 1;
547:C        8                     end
548:C        8                   else
549:C        8                     begin
550:C        8                       i             := 25;
551:C        8                       done          := true;
552:C        8                     end;
553:C        6                   catindx     := catindx + 24;
554:C        6                 end;
555:C        5             end;
556:C        4         end;
557:C        3       fpeof := catentryindx;
558:C        3     end;
559:C        2 end;
560:S
561:D    -6  1 (*************************************************************************)
562:D        1 procedure srm_catalog(anyvar f      : fib;
563:D        2                       unum   : unitnum);
564:D        2 type
565:D        2   catarray      = array[0..maxint] of catentry;
566:D        2   catarrayptr   = ^catarray;
567:D        2   ac_char_arr   = array [ac_manager..ac_createlink] of char;
568:D        2 const
569:D        2   ac_chars      = ac_char_arr['M','R','W','S','P','C'];
570:D        2 var
571:D   -16  2   volpass       : name_type;
572:D   -52  2   fns           : file_name_set;
573:D   -53  2   done          : boolean;
574:D   -58  2   i             : integer;
575:D   -62  2   j             : integer;
576:D   -66  2   catindx       : integer;
577:D   -70  2   catentryindx  : integer;
578:D   -72  2   ac            : access_code_type;
579:C        2 begin
580:C        2   catentryindx  := 0;
581:C        2   done  := false;
582:C        2   with f, packet_ptr.rcat^ do
583:C        3     begin
584:C        3       strtopaoc16(fvid,volpass);
585:C        3       catindx   := fpos + 1;
586:C        3       while (catentryindx < fpeof) and not done do
587:C        4         begin
588:C        4           catpack(unum,0,addr(fns),start_alternate,
589:C        4                   pathid,volpass,7,catindx);
590:C        4           if ioresult <> ord(inoerror) then
591:C        5             done      := true
592:C        5           else
593:C        5             begin
594:C        5               i  := 1;
595:C        5               if actual_num_files < 7 then
596:C        6                 done      := true;
597:C        5               while i <= actual_num_files do
598:C        6                 if catentryindx < fpeof then
599:C        7                   begin
600:C        7                     with cat_info[i], catarrayptr(fwindow)^[catentryindx] do
```

```
601:C        8                        begin
602:C        8                          paoc16tostr(file_name,cname);
603:C        8                          ceft       := file_code.si2;
604:C        8                          ckind      := mapfkind(file_code);
605:C        8                          cpsize     := physical_size;
606:C        8                          clsize     := logical_eof;
607:C        8                          cstart     := -1;
608:C        8                          translatedate(creation_date,ccreatedate,ccreatetime);
609:C        8                          translatedate(last_access_date,clastdate,clasttime);
610:C        8                          cblocksize:= -1;
611:C        8                          cextra1    := -1;
612:C        8                          cextra2    := -1;
613:C        8                          setstrlen(cinfo,ord(ac_createlink)+1);
614:C        8                          for ac := ac_manager to ac_createlink do
615:C        9                            if capabilities[ac] then
616:C       10                              cinfo[ord(ac) + 1] := ac_chars[ac]
617:C       10                            else
618:C       10                              cinfo[ord(ac) + 1] := ' ';
619:C        8                          case share_code of
620:C        9                            exclusive_share_code  : cinfo := cinfo + ' EXCLUSIVE';
621:C        9                            shared_share_code     : cinfo := cinfo + ' SHARED';
622:S                                     (
623:S                                     closed_share_code     : cinfo := cinfo + ' CLOSED';
624:C        9                            )
625:C        9                            corrupt_share_code    : cinfo := cinfo + ' CORRUPT';
626:C        9                            otherwise               cinfo := cinfo + ' CLOSED';
627:C        9                          end;
628:C        8                        end;
629:C        7                      i := i + 1;
630:C        7                      catentryindx := catentryindx + 1;
631:C        7                    end
632:C        7                  else
633:C        7                    begin
634:C        7                      i       := 8;
635:C        7                      done    := true;
636:C        7                    end;
637:C        5                catindx     := catindx + 7;
638:C        5              end;
639:C        4          end;
640:C        3        fpeof := catentryindx;
641:C        3      end;
642:C        2 end;
643:S
644:D    -6  1 (*****************************************************************************)
645:D        1 procedure srm_open_dir(anyvar f         : fib;
646:D        2                                unum      : unitnum;
647:D        2                                opentype  : gang_open_type;
648:D        2                                openparent:boolean);
649:D        2 var
650:D   -16  2   volpass       : name_type;
651:D   -20  2   lentitle      : integer;
652:D   -24  2   sindx         : integer;
653:D   -28  2   pindx         : integer;
654:D   -32  2   nindx         : integer;
655:D   -36  2   i             : integer;
656:D   -38  2   path          : path_start_type;
657:D   -42  2   saveid        : file_id_type;
658:D   -46  2   origpathid    : file_id_type;
659:D   -47  2   last          : boolean;
660:D   -48  2   alreadyopen   : boolean;
```



```
661:D   -52  2   nfns          : integer;
662:D  -268  2   nsa           : name_set_array;
663:S
664:D        2 procedure getpaoc(anyvar paoc   : name_type;
665:D        3                       del1   : char;
666:D        3                       del2   : char);
667:D        3 var
668:D    -1  3   done  : boolean;
669:C        3 begin
670:C        3   with f do
671:C        4     begin
672:C        4       done  := false;
673:C        4       while (sindx <= lentitle) and (pindx <= name_type_len) and (not done) do
674:C        5         if  (ftitle[sindx] = del1) or (ftitle[sindx] = del2) then
675:C        6           done           := true
676:C        6         else
677:C        6           begin
678:C        6             paoc[pindx]   := ftitle[sindx];
679:C        6             pindx         := pindx + 1;
680:C        6             sindx         := sindx + 1;
681:C        6           end;
682:C        4       if (sindx > lentitle) then
683:C        5         begin
684:C        5           if del1 = '>' then
685:C        6             ioresult := ord(ibadpass);
686:C        5         end
687:C        5       else
688:C        5         begin
689:C        5           if  (ftitle[sindx] <> del1) and (ftitle[sindx] <> del2) then
690:C        6             begin
691:C        6               if del1 = '>' then
692:C        7                 ioresult := ord(ibadpass)
693:C        7               else
694:C        7                 ioresult := ord(ibadtitle);
695:C        6             end
696:C        6           else
697:C        6             if (del1 = '>') then
698:C        7               if (ftitle[sindx] <> '>') then
699:C        8                 ioresult := ord(ibadpass);
700:C        5         end;
701:C        4     end;
702:C        3 end;
703:S
704:S
705:S
706:C        2 begin
707:C        2   last          := false;
708:C        2   alreadyopen   := false;
709:C        2   sindx         := 1;
710:S
711:C        2   with f do
712:C        3     begin
713:C        3       origpathid        := pathid;
714:C        3       lentitle          := strlen(ftitle);
715:C        3       setstrlen(ftid,0);
716:C        3       if pathid = -1 then
717:C        4         begin
718:C        4           setstrlen(fvid,0);
719:C        4           setstrlen(ffpw,0);
720:C        4         end;
```

```
721:C        3       if (sindx <= lentitle) then
722:C        4         if ftitle[sindx] = '<' then       {get volume password}
723:C        5           begin
724:C        5             sindx   := sindx + 1;
725:C        5             pindx   := 1;
726:C        5             volpass := '';
727:C        5             getpaoc(volpass,'>','>');
728:C        5             paoc16tostr(volpass,fvid);
729:C        5             sindx   := sindx + 1;
730:C        5           end;
731:S
732:C        3       path := start_alternate;
733:C        3       if ioresult = ord(inoerror) then
734:C        4         if (sindx <= lentitle) then
735:C        5           if ftitle[sindx] = '/' then
736:C        6             begin
737:C        6               path  := start_root;
738:C        6               sindx := sindx + 1;
739:C        6             end;
740:C        3       if pathid = -1 then
741:C        4         if path = start_root then
742:C        5           pathid := 0
743:C        5         else
744:C        5           pathid := unitable^[unum].dvrtemp;
745:S
746:C        3       if sindx > lentitle then
747:C        4         begin
748:C        4           last      := true;
749:C        4           setstrlen(ftitle,0);
750:C        4         end
751:C        4       else
752:C        4         if (ftitle[lentitle] = '/') then
753:C        5           setioresult(ord(ibadtitle));
754:S
755:C        3       with packet_ptr.ropen^ do
756:C        4         while (not last) and (ioresult = ord(inoerror)) do
757:C        5           begin
758:C        5             nfns        := 0;
759:C        5             while (sindx <= lentitle) and (nfns < 6) and (ioresult = ord(inoerror)) do
760:C        6               begin
761:C        6                 with nsa[nfns+1] do
762:C        7                   begin
763:C        7                     if (ftitle[sindx] = '/') then
764:C        8                       ioresult      := ord(ibadtitle)
765:C        8                     else
766:C        8                       begin
767:C        8                         file_name       := ' ';
768:C        8                         password        := ' ';
769:C        8                         pindx           := 1;
770:C        8                         getpaoc(file_name,'<','/');
771:C        8                         nindx           := pindx;
772:C        8                         if ioresult = ord(inoerror) then
773:C        9                           if (sindx <= lentitle) then
774:C       10                             if (ftitle[sindx] = '<') then
775:C       11                               begin
776:C       11                                 sindx   := sindx + 1;
777:C       11                                 pindx   := 1;
778:C       11                                 getpaoc(password,'>','>');
779:C       11                                 sindx   := sindx + 1;
780:C       11                                 if sindx <= lentitle then
```



```
781:C       12                                   if ftitle[sindx] <> '/' then
782:C       13                                     if nindx > lentitle then
783:C       14                                       ioresult := ord(ibadtitle)
784:C       14                                     else
785:C       14                                       begin
786:C       14                                         pindx := nindx;
787:C       14                                         getpaoc(file_name,'/','/');
788:C       14                                       end;
789:C       11                                 end;
790:C        8                           if ioresult = ord(inoerror) then
791:C        9                             if (sindx > lentitle) then
792:C       10                               begin
793:C       10                                 last    := true;
794:C       10                                 setstrlen(ftitle,0);
795:C       10                               end;
796:C        8                         end;
797:C        7                     end;
798:C        6                   if ioresult = ord(inoerror) then
799:C        7                     nfns := nfns + 1;
800:C        6                   sindx   := sindx + 1;
801:C        6                 end;
802:S
803:C        5               if ioresult = ord(inoerror) then
804:C        6                 begin
805:C        6                   if not (last and openparent) then
806:C        7                     begin
807:C        7                       openpack(unum,nfns,addr(nsa),path,
808:C        7                                pathid,volpass,shared_share_code,opentype);
809:C        7                       if ioresult <> ord(inoerror) then
810:C        8                         begin
811:C        8                           if last then
812:C        9                             begin
813:C        9                               ioresult := ord(inoerror);
814:C        9                               openparent := true;
815:C        9                             end;
816:C        8                         end
817:C        8                       else
818:C        8                         begin
819:C        8                           saveid        := file_id;
820:C        8                           if alreadyopen then
821:C        9                             srm_close_pathid(unum,pathid,false)
822:C        9                           else
823:C        9                             alreadyopen := true;
824:C        8                           pathid        := saveid;
825:C        8                           path          := start_alternate;
826:C        8                         end;
827:C        7                     end;
828:C        6                   if ioresult = ord(inoerror) then
829:C        7                     begin
830:C        7                       if last and openparent then
831:C        8                         begin
832:C        8                           if (nfns <= 1) then
833:C        9                             begin
834:C        9                               if pathid = -1 then
835:C       10                                 setioresult(ord(inodirectory));
836:C        9                             end
837:C        9                           else
838:C        9                             begin
839:C        9                               openpack(unum,nfns-1,addr(nsa),path,
840:C        9                                        pathid,volpass,shared_share_code,opentype);
```

```
841:C        9                              if ioresult <> ord(inoerror) then
842:C       10                                begin
843:C       10                                  if alreadyopen then
844:C       11                                    srm_close_pathid(unum,pathid,false);
845:C       10                                end
846:C       10                              else
847:C       10                                begin
848:C       10                                  saveid        := file_id;
849:C       10                                  if alreadyopen then
850:C       11                                    srm_close_pathid(unum,pathid,false);
851:C       10                                  pathid        := saveid;
852:C       10                                end;
853:C        9                            end;
854:C        8                          if ioresult = ord(inoerror) then
855:C        9                            with nsa[nfns] do
856:C       10                              begin
857:C       10                                paoc16tostr(file_name,ftid);
858:C       10                                ftitle      := ftid;
859:C       10                                if password <> ' ' then
860:C       11                                  paoc16tostr(password,ffpw);
861:C       10                              end;
862:C        8                        end;
863:C        7                    end;
864:C        6                end;
865:C        5            end;
866:C        3        if ((origpathid <> -1) and (origpathid = pathid))
867:C        4        or (pathid = unitable^[unum].dvrtemp) then
868:C        4          fsavepathid     := true
869:C        4        else
870:C        4          fsavepathid     := false;
871:C        3      end;
872:C        2 end;
873:S
874:D    -6  1 (*****************************************************************************)
875:D        1 procedure srm_set_unit_prefix(anyvar f   : fib;
876:D        2                               unum      : unitnum);
877:D        2 var
878:D    -4  2   savpathid     : integer;
879:C        2 begin
880:C        2   with f, unitable^[unum] do
881:C        3     begin
882:C        3       srm_open_dir(f,unum,open_protectec_directory,false);
883:C        3       if ioresult = ord(inoerror) then
884:C        4         begin
885:C        4           if strlen(ftitle) > 0 then
886:C        5             setioresult(ord(inounit))
887:C        5           else
888:C        5             begin
889:C        5               savpathid := pathid;
890:C        5               pathid    := dvrtemp;
891:C        5               dvrtemp   := savpathid;
892:C        5             end;
893:C        4           srm_close_pathid(unum,pathid,false);
894:C        4         end;
895:C        3       srm_get_vol_name(uvid,unum);
896:C        3     end;
897:C        2 end;
898:S
899:D    -6  1 (*****************************************************************************)
900:D        1 procedure doopenpack(unum       : unitnum;
```



```
901:D        2                       var f      : fib;
902:D        2                       nfns       : integer;
903:D        2                       anyvar nsa : name_set_array;
904:D        2                       path       : path_start_type;
905:D        2                       volpass    : name_type;
906:D        2                       sharecode  : integer;
907:D   -16  2                       lockable   : boolean);
908:D   -16  2 type
909:D   -16  2   trickrec      = record case boolean of
910:D   -16  2                     true  :(i   : integer);
911:D   -16  2                     false :(chs : packed array [1..2] of char;
912:D   -16  2                             si2 : shortint);
913:D   -16  2                   end;
914:D   -16  2 var
915:D   -20  2   temprec       : trickrec;
916:C        2 begin
917:C        2       with f, packet_ptr.ropen^ do
918:C        3         begin
919:C        3           if lockable and fistextvar then
920:C        4             setioresult(ord(inotlockable))
921:C        4           else
922:C        4             openpack(unum,nfns,addr(nsa),path,pathid,volpass,sharecode,open_data);
923:C        3           if ioresult = ord(inoerror) then
924:C        4             if file_code.si2 = 3 then   {directory}
925:C        5               begin
926:C        5                 setioresult(ord(inotondir));
927:C        5                 closepack(unum,file_id);
928:C        5               end
929:C        5             else
930:C        5               begin
931:C        5                 fileid        := file_id;
932:C        5                 fpeof         := open_logical_eof;
933:C        5                 fleof         := open_logical_eof;
934:C        5                 feft          := file_code.si2;
935:C        5                 fkind         := mapfkind(file_code);
936:C        5                 flockable     := lockable;
937:C        5                 flocked       := not lockable;  {default to locked unless lockable}
938:C        5                 if (feft = BDATTYPE)            {BDAT file}
939:C        6                 or (feft = BASICBINTYPE)        {BIN  file}
940:C        6                 or (feft = BASICPROGTYPE) then  {PROG file}
941:C        6                   begin
942:C        6                     temprec.chs   := '  ';
943:C        6                     temprec.si2   := max_record_size div 2;
944:C        6                     fstartaddress := temprec.i;
945:C        6                   end
946:C        6                 else
947:C        6                   fstartaddress   := boot_start_address;
948:C        5                 if not fbuffered   then     am := amtable^[untypedfile]
949:C        6                 else if fistextvar then     am := amtable^[fkind]
950:C        7                 else                        am := amtable^[datafile];
951:C        5               end;
952:C        3         end;
953:C        2 end;
954:S
955:D    -6  1 (*****************************************************************************)
956:D        1 procedure srm_open_file(anyvar f        : fib;
957:D        2                         unum     : unitnum);
958:D        2 type
959:D        2   trickrec      = record case boolean of
960:D        2                     true  :(i   : integer);
```

```
 961:D        2                     false :(si1 : shortint;
 962:D        2                            si2 : shortint);
 963:D        2                   end;
 964:D        2 var
 965:D   -16  2   volpass       : name_type;
 966:D   -52  2   fns           : file_name_set;
 967:D   -56  2   temprec       : trickrec;
 968:D   -60  2   sharemode     : integer;
 969:D   -64  2   nps           : integer;
 970:D  -640  2   psa           : protectcode_set_array;
 971:D  -641  2   lockable      : boolean;
 972:C        2 begin
 973:C        2   with f do
 974:C        3     begin
 975:C        3       setup_fns(f, fns);
 976:C        3       parseoptparm(foptstring,sharemode,lockable,nps,psa,true);
 977:C        3       if ioresult = ord(inoerror) then
 978:C        4         begin
 979:C        4           strtopaoc16(fvid,volpass);
 980:C        4           doopenpack(unum,f,1,fns,start_alternate,volpass,sharemode,lockable);
 981:C        4         end;
 982:C        3     end;
 983:C        2 end;
 984:S
 985:D    -6  1 (*****************************************************************************)
 986:D        1 procedure srm_create_dir (anyvar f      : fib;
 987:D        2                           unum   : unitnum);
 988:D        2 type
 989:D        2   catentryptr   = ^catentry;
 990:D        2 const
 991:D        2   dirfilecode   = gang_file_codes[i:3];
 992:D        2 var
 993:D   -16  2   volpass       : name_type;
 994:D   -52  2   fns           : file_name_set;
 995:C        2 begin
 996:C        2   with f, catentryptr(fwindow)^ do
 997:C        3     if strlen(cname) = 0 then
 998:C        4       setioresult(ord(ibadtitle))
 999:C        4     else
1000:C        4       begin
1001:C        4         with fns do
1002:C        5           begin
1003:C        5             password      := ' ';
1004:C        5             strtopaoc16(cname,file_name);
1005:C        5           end;
1006:C        4         strtopaoc16(fvid,volpass);
1007:C        4         createpack(unum,i,addr(fns),start_alternate,pathid,volpass,0,nil,
1008:C        4                    dirfilecode,directory_records,0,0,0,0);
1009:C        4       end;
1010:C        2 end;
1011:S
1012:D    -6  1 (*****************************************************************************)
1013:D        1 procedure srm_create_file(anyvar f      : fib;
1014:D        2                           unum   : unitnum);
1015:D        2 const
1016:D        2   dirfilecode   = gang_file_codes[i:3];
1017:D        2 type
1018:D        2   trickrec      = record case boolean of
1019:D        2                     true  :(i    : integer);
1020:D        2                     false :(si1 : shortint;
```



```
1021:D        2                             si2 : shortint);
1022:D        2                   end;
1023:D        2 var
1024:D   -16  2   volpass       : name_type;
1025:D  -124  2   nsa           : name_set_array_three;
1026:D  -128  2   ext1          : integer;
1027:D  -132  2   temprec       : trickrec;
1028:D  -136  2   sharemode     : integer;
1029:D  -140  2   maxrec        : integer;
1030:D  -144  2   nps           : integer;
1031:D  -148  2   usefeft       : gang_file_codes;
1032:D  -724  2   psa           : protectcode_set_array;
1033:D  -728  2   i             : integer;
1034:D  -730  2   ac            : ac_manager..ac_purgelink;
1035:D  -731  2   lockable      : boolean;
1036:S
1037:C        2 begin
1038:C        2   with f, nsa[3] do
1039:C        3     begin
1040:C        3       strtopaoc16(fvid,volpass);
1041:C        3       repeat
1042:C        4         ioresult        := ord(inoerror);
1043:C        4         fanonctr        := tempcounter;
1044:C        4         tempcounter     := tempcounter + 1;
1045:C        4         usefeft.i       := feft;
1046:S
1047:C        4         if feft = BDATTYPE then        (BDAT file)
1048:C        5           begin
1049:C        5             temprec.i       := fstartaddress;
1050:C        5             maxrec          := temprec.si2 * 2;
1051:C        5             if maxrec < 1 then
1052:C        6               maxrec := 1;
1053:C        5           end
1054:C        5         else
1055:C        5           maxrec    := 256;
1056:C        4         if fpos > 0 then
1057:C        5           ext1 := fpos
1058:C        5         else
1059:C        5           ext1 := extentsize;
1060:C        4         parseoptparm(foptstring,sharemode,lockable,nps,psa,false);
1061:C        4         check_protectcode_set_array(nps,psa);
1062:C        4         if (nps > 0) and (nps < 24) and (ioresult = ord(inoerror)) then
1063:C        5           begin
1064:C        5             nps := nps + 1;
1065:C        5             with psa[nps] do
1066:C        6               begin
1067:C        6                 password        := temp_file_pass;
1068:C        6                 capabilities    := nocapabilities;
1069:C        6                 if nps > 1 then
1070:C        7                   for i := 1 to nps-1 do
1071:C        8                     for ac := ac_manager to ac_purgelink do
1072:C        9                       if psa[i].capabilities[ac] then
1073:C       10                         capabilities[ac] := true;
1074:C        6               end;
1075:C        5           end;
1076:S
1077:C        4         if (not fanonymous) and (ioresult = ord(inoerror)) then
1078:C        5           begin
1079:C        5             setup_fns(f,nsa);
1080:C        5             foldfileid := -1;
```

```
  1081:C        5              openpack(unum,1,addr(nsa),start_alternate,pathid,volpass,sharemode,open_data);
  1082:C        5              if ioresult = ord(inofile) then
  1083:C        6                begin
  1084:C        6                  ioresult := ord(inoerror);
  1085:C        6                  nsa[1].password := ' ';
  1086:C        6                  createpack(unum,1,addr(nsa),start_alternate,pathid,volpass,
  1087:C        6                             nps,addr(psa),usefeft,data_records,
  1088:C        6                             maxrec,ext1,extentsize,fstartaddress);
  1089:C        6                  if ioresult = ord(inoerror) then
  1090:C        7                    if feft <> SYSTMTYPE then
  1091:C        8                      fnosrmtemp := true
  1092:C        8                    else
  1093:C        8                      begin                {SYSTM files must go through temp first}
  1094:C        8                        nsa[1].password := temp_file_pass;
  1095:C        8                        openpack(unum,1,addr(nsa),start_alternate,pathid,
  1096:C        8                                 volpass,sharemode,open_data);
  1097:C        8                      end;
  1098:C        6                end;
  1099:C        5              if (ioresult = ord(inoerror)) and (not fnosrmtemp) then
  1100:C        6                begin
  1101:C        6                  foldfileid := packet_ptr.ropen^.file_id;
  1102:C        6                  fileinfopack(unum,foldfileid);
  1103:C        6                  with packet_ptr.rfileinfo^.file_info do
  1104:C        7                    begin
  1105:C        7                      if  (not capabilities[ac_manager])
  1106:C        8                      and (not capabilities[ac_purgelink]) then
  1107:C        8                        setioresult(ord(ibadpass))   {won't be able to purge old}
  1108:C        8                      else           {this test added in version 2.2 on 4-May-83}
  1109:C        8                        if file_code.si2 = 3 then {disallow rewrite on directory}
  1110:C        9                          setioresult(ord(inotondir));
  1111:C        7                      if ioresult <> ord(inoerror) then
  1112:C        8                        srm_close_fileid(unum,foldfileid);
  1113:C        7                    end;
  1114:C        6                end;
  1115:C        5            end;
  1116:S
  1117:C        4          if (ioresult = ord(inoerror)) and (not fnosrmtemp) then
  1118:C        5            begin
  1119:C        5              setup_fns3(f,nsa);
  1120:C        5              if ioresult = ord(inoerror) then
  1121:C        6                begin
  1122:C        6                  password        := ' ';
  1123:C        6                  createpack(unum,3,addr(nsa),start_root,pathid,volpass,
  1124:C        6                             nps,addr(psa),usefeft,data_records,
  1125:C        6                             maxrec,ext1,extentsize,fstartaddress);
  1126:C        6                  if ioresult = ord(inofile) then
  1127:C        7                    begin
  1128:C        7                      ioresult := ord(inoerror);
  1129:C        7                      createpack(unum,2,addr(nsa),start_root,pathid,volpass,
  1130:C        7                                 0,nil,dirfilecode,directory_records,0,0,0,0);
  1131:C        7                      if ioresult <> ord(inoerror) then
  1132:C        8                        ioresult := ord(ineedtempdir)
  1133:C        8                      else
  1134:C        8                        createpack(unum,3,addr(nsa),start_root,pathid,volpass,
  1135:C        8                                   nps,addr(psa),usefeft,data_records,
  1136:C        8                                   maxrec,ext1,extentsize,fstartaddress);
  1137:C        7                    end
  1138:C        7                  else
  1139:C        7                    if ioresult = ord(idupfile) then
  1140:C        8                      begin
```



```
  1141:C        8                        ioresult := ord(inoerror);
  1142:C        8                        password := temp_file_pass;
  1143:C        8                        purgepack(unum,3,addr(nsa),start_root,pathid,volpass);
  1144:C        8                        ioresult := ord(inoerror);
  1145:C        8                        password := ' ';
  1146:C        8                        createpack(unum,3,addr(nsa),start_root,pathid,volpass,
  1147:C        8                                   nps,addr(psa),usefeft,data_records,
  1148:C        8                                   maxrec,ext1,extentsize,fstartaddress);
  1149:C        8                      end;
  1150:C        6                end;
  1151:C        5              end;
  1152:C        4          until (ioresult <> ord(idupfile));
  1153:C        3          if ioresult <> ord(inoerror) then
  1154:C        4            srm_close_fileid(unum,foldfileid)
  1155:C        4          else
  1156:C        4            if fnosrmtemp then
  1157:C        5              begin
  1158:C        5                nsa[1].password := temp_file_pass;
  1159:C        5                doopenpack(unum,f,1,nsa,start_alternate,volpass,sharemode,lockable);
  1160:C        5              end
  1161:C        5            else
  1162:C        5              begin
  1163:C        5                password := temp_file_pass;
  1164:C        5                doopenpack(unum,f,3,nsa,start_root,volpass,exclusive_share_code,lockable);
  1165:C        5              end;
  1166:C        3      end;
  1167:C        2  end;
  1168:S
  1169:D     -6  1  (*****************************************************************************)
  1170:D         1  procedure srm_change_name(anyvar f      : fib;
  1171:D         2                               unum   : unitnum);
  1172:D         2  type
  1173:D         2    fidptr         = ^fid;
  1174:D         2  var
  1175:D    -16  2    volpass        : name_type;
  1176:D    -18  2    path1          : path_start_type;
  1177:D    -22  2    nfns1          : integer;
  1178:D    -58  2    fns1           : file_name_set;
  1179:D    -60  2    path2          : path_start_type;
  1180:D    -64  2    nfns2          : integer;
  1181:D   -100  2    fns2           : file_name_set;
  1182:C         2  begin
  1183:C         2    with f, unitable^[unum] do
  1184:C         3      begin
  1185:C         3        srm_open_dir(f,unum,open_directory,true);
  1186:C         3        if ioresult = ord(inoerror) then
  1187:C         4          setup_fns(f, fns1);
  1188:C         3        if ioresult = ord(inoerror) then
  1189:C         4          with fns2 do
  1190:C         5            begin
  1191:C         5              file_name   := ' ';
  1192:C         5              password    := ' ';
***WARNING: (line 1193): STRPOS does not conform to HP standard, see $SWITCH_STRPOS$
  1193:C         5              if strpos('/',fidptr(fwindow)^) = 0 then
***WARNING: (line 1194): STRPOS does not conform to HP standard, see $SWITCH_STRPOS$
  1194:C         6                if (strpos('<',fidptr(fwindow)^) = 0) then
  1195:C         7                  strtopaoc16(fidptr(fwindow)^,file_name);
  1196:C         5              if file_name = ' ' then
  1197:C         6                setioresult(ord(ibadtitle));
  1198:C         5            end;
```

```
1199:C        3       if ioresult = ord(inoerror) then
1200:C        4         begin
1201:C        4           strtopacc16(fvid,volpass);
1202:C        4           createlinkpack(unum,1,addr(fns1),start_alternate,pathid,volpass,
1203:C        4                          1,addr(fns2),start_alternate,pathid,volpass,true);
1204:C        4         end;
1205:C        3       srm_close_pathid(unum,pathid,fsavepathid);
1206:C        3     end;
1207:C        2 end;
1208:S
1209:D    -6  1 (*****************************************************************************)
1210:D        1 procedure srm_dup_link(anyvar f          : fib;
1211:D        2                        unum      : unitnum);
1212:D        2 var
1213:D   -16  2   volpass       : name_type;
1214:D   -32  2   volpass2      : name_type;
1215:D   -34  2   path1         : path_start_type;
1216:D   -38  2   nfns1         : integer;
1217:D   -74  2   fns1          : file_name_set;
1218:D   -76  2   path2         : path_start_type;
1219:D   -80  2   nfns2         : integer;
1220:D  -116  2   fns2          : file_name_set;
1221:C        2 begin
1222:C        2   with f, unitable^[unum] do
1223:C        3     begin
1224:C        3       srm_open_dir(f,unum,open_directory,true);
1225:C        3       if ioresult = ord(inoerror) then
1226:C        4         setup_fns(f, fns1);
1227:C        3       if ioresult = ord(inoerror) then
1228:C        4         srm_open_dir(fibp(fwindow)^,unum,open_directory,true);
1229:C        3       if ioresult = ord(inoerror) then
1230:C        4         setup_fns(fibp(fwindow)^, fns2);
1231:C        3       if ioresult = ord(inoerror) then
1232:C        4         begin
1233:C        4           strtopacc16(fvid,volpass);
1234:C        4           strtopacc16(fibp(fwindow)^.fvid,volpass2);
1235:C        4           createlinkpack(unum,1,addr(fns1),start_alternate,pathid,volpass,
1236:C        4               1,addr(fns2),start_alternate,fibp(fwindow)^.pathid,volpass2,
1237:C        4               fpurgeoldlink);
1238:C        4         end;
1239:C        3       srm_close_pathid(unum,pathid,fsavepathid);
1240:C        3       srm_close_pathid(unum,fibp(fwindow)^.pathid,fibp(fwindow)^.fsavepathid);
1241:C        3     end;
1242:C        2 end;
1243:S
1244:D    -6  1 (*****************************************************************************)
1245:D        1 procedure srm_purge_name(anyvar f        : fib;
1246:D        2                          unum    : unitnum);
1247:D        2 var
1248:D   -16  2   volpass       : name_type;
1249:D   -52  2   fns           : file_name_set;
1250:C        2 begin
1251:C        2   with f do
1252:C        3       begin
1253:C        3         setup_fns(f, fns);
1254:C        3         if ioresult = ord(inoerror) then
1255:C        4           begin
1256:C        4             strtopaoc16(fvid,volpass);
1257:C        4             purgepack(unum,1,addr(fns),start_alternate,pathid,volpass);
1258:C        4           end;
```



```
1259:C        3       end;
1260:C        2 end;
1261:S
1262:D    -6  1 (*****************************************************************************)
1263:D        1 procedure srm_purge_file(anyvar f        : fib;
1264:D        2                          unum    : unitnum);
1265:D        2 var
1266:D   -16  2   volpass       : name_type;
1267:D   -18  2   path          : path_start_type;
1268:D   -22  2   nfns          : integer;
1269:D  -130  2   nsa           : name_set_array_three;
1270:C        2 begin
1271:C        2   with f do
1272:C        3     if (strlen(ftid) = 0) and not fisnew then
1273:C        4       setioresult(ord(ibadtitle))
1274:C        4     else
1275:C        4       begin
1276:C        4         if fmodified then
1277:C        5           if not (flockable and not flocked) then
1278:C        6             begin
1279:C        6               seteofpack(funit,fileid,false,fleof);
1280:C        6               if ioresult = ord(ilostfile) then
1281:C        7                 fileid := -1;
1282:C        6             end;
1283:C        4         if fisnew and (not fanonymous) and (not fnosrmtemp) then
1284:C        5           srm_close_fileid(unum,foldfileid);
1285:C        4         srm_close_fileid(unum,fileid);
1286:C        4         setup_fns3(f,nsa);
1287:C        4         if ioresult <> ord(ibadtitle) then
1288:C        5           begin
1289:C        5             if (fisnew) and (not fnosrmtemp) then
1290:C        6               begin
1291:C        6                 path        := start_root;
1292:C        6                 nfns        := 3;
1293:C        6               end
1294:C        6             else
1295:C        6               begin
1296:C        6                 path        := start_alternate;
1297:C        6                 nfns        := 1;
1298:C        6               end;
1299:C        5             strtopaoc16(fvid,volpass);
1300:C        5             if fisnew then
1301:C        6               nsa[nfns].password := temp_file_pass;
1302:C        5             purgepack(unum,nfns,addr(nsa),path,pathid,volpass);
1303:C        5           end;
1304:C        4         if not (fisnew and fanonymous) then
1305:C        5           srm_close_pathid(unum,pathid,fsavepathid);
1306:C        4       end;
1307:C        2 end;
1308:S
1309:D    -6  1 (*****************************************************************************)
1310:D        1 procedure srm_stretch(anyvar f          : fib;
1311:D        2                       unum       : unitnum);
1312:D        2 var
1313:D   -16  2   volpass       : name_type;
1314:D   -20  2   neweof        : integer;
1315:C        2 begin
1316:C        2   with f do
1317:C        3     begin
1318:C        3       neweof            := ((fpos div extentsize) + 1) * extentsize;
```

```
1319:C        3       seteofpack(funit,fileid,false,neweof);
1320:C        3       if ioresult = ord(inoerror) then
1321:C        4         begin
1322:C        4           fpeof          := neweof;
1323:C        4           fmodified      := true;
1324:C        4         end;
1325:C        3     end;
1326:C        2 end;
1327:S
1328:D     -6 1 (*****************************************************************************)
1329:D        1 procedure srm_close_file(anyvar f       : fib;
1330:D        2                          unum    : unitnum);
1331:D        2 var
1332:D   -16  2   volpass       : name_type;
1333:D  -124  2   nsal          : name_set_array_three;
1334:D  -160  2   fns           : file_name_set;
1335:D  -164  2   tempioresult  : integer;
1336:D  -168  2   ext1          : integer;
1337:D  -172  2   saveleof      : integer;
1338:D  -176  2   savefileid    : file_id_type;
1339:D  -180  2   usefeft       : gang_file_codes;
1340:D  -204  2   pcs           : protect_code_set;
1341:C        2 begin
1342:C        2   with f do
1343:C        3     begin
1344:C        3       if fmodified then
1345:C        4         if not (flockable and not flocked) then
1346:C        5           begin
1347:C        5             seteofpack(funit,fileid,false,fleof);
1348:C        5             if ioresult = ord(ilostfile) then
1349:C        6               fileid := -1;
1350:C        5           end;
1351:C        3       if not fisnew then
1352:C        4         srm_close_fileid(unum,fileid)
1353:C        4       else
1354:C        4         if ioresult <> ord(inoerror) then
1355:C        5           srm_purge_file(f,unum)
1356:C        5         else
1357:C        5           begin
1358:C        5             strtopaoc16(fvid,volpass);
1359:C        5             setup_fns3(f,nsal);
1360:C        5             setup_fns(f,fns);
1361:C        5             if (ioresult = ord(inoerror)) and (not fnosrmtemp) then
1362:C        6               if not foverwritten then
1363:C        7                 begin
1364:C        7                   if not fanonymous then
1365:C        8                     begin
1366:C        8                       srm_close_fileid(unum,foldfileid);
1367:C        8                       purgepack(unum,1,addr(fns),start_alternate,pathid,volpass);
1368:C        8                     end;
1369:C        7                   if ioresult = ord(inofile) then
1370:C        8                     ioresult := ord(inoerror);
1371:C        7                   if ioresult <> ord(inoerror) then
1372:C        8                     srm_purge_file(f,unum);
1373:C        7                 end
1374:C        7               else
1375:C        7                 begin
1376:C        7                   if foldfileid < 0 then
1377:C        8                     foverwritten := false
1378:C        8                   else
```



```
1379:C        8                     begin
1380:C        8                       exchangepack(unum,foldfileid,fileid);
1381:C        8                       srm_close_fileid(unum,foldfileid);
1382:C        8                       srm_close_pathid(unum,pathid,fsavepathid);
1383:C        8                     end;
1384:C        7                   srm_purge_file(f,unum);
1385:C        7                 end;
1386:C        5             if (ioresult = ord(inoerror)) and (not foverwritten) then
1387:C        6               if feft <> SYSTMTYPE then   (not SYSTM file)
1388:C        7                 begin
1389:C        7                   srm_close_fileid(unum,fileid);
1390:C        7                   if (ioresult = ord(inoerror)) and (not fnosrmtemp) then
1391:C        8                     createlinkpack(unum,3,addr(nsal),start_root,pathid,volpass,
1392:C        8                                    1,addr(fns),start_alternate,pathid,volpass,true);
1393:C        7                   if ioresult <> ord(inoerror) then
1394:C        8                     srm_purge_file(f,unum)
1395:C        8                   else
1396:C        8                     begin
1397:C        8                       fns.password := temp_file_pass;
1398:C        8                       with pcs do
1399:C        9                         begin
1400:C        9                           password        := temp_file_pass;
1401:C        9                           capabilities    := nocapabilities;
1402:C        9                         end;
1403:C        8                       tempioresult := ioresult;
1404:C        8                       changeprotectpack(unum,1,addr(fns),start_alternate,
1405:C        8                                         pathid,volpass,1,addr(pcs));
1406:C        8                       ioresult := tempioresult;
1407:C        8                     end;
1408:C        7                 end
1409:C        7               else
1410:C        7                 begin                 (SYSTM file)
1411:C        7                   savefileid  := fileid;
1412:C        7                   saveleof    := fleof;
1413:C        7                   fpos        := saveleof;
1414:C        7                   if fpos > 0 then
1415:C        8                     ext1 := fpos
1416:C        8                   else
1417:C        8                     ext1 := extentsize;
1418:C        7                   usefeft.i   := feft;
1419:C        7                   fns.password := ' ';
1420:C        7                   createpack(unum,1,addr(fns),start_alternate,pathid,
1421:C        7                              volpass,0,nil,usefeft,data_records,
1422:C        7                              256,ext1,extentsize,fstartaddress);
1423:C        7                   if ioresult = ord(inoerror) then
1424:C        8                     doopenpack(unum,f,1,fns,start_alternate,volpass,exclusive_share_code,false);
1425:C        7                   if ioresult = ord(inoerror) then
1426:C        8                     copypack(unum,savefileid,0,fileid,0,saveleof);
1427:C        7                   srm_close_fileid(unum,fileid);
1428:C        7                   srm_close_pathid(unum,pathid,fsavepathid);
1429:C        7                   fileid      := savefileid;
1430:C        7                   srm_purge_file(f,unum);
1431:C        7                 end;
1432:C        5             end;
1433:C        3       srm_close_pathid(unum,pathid,fsavepathid);
1434:C        3     end;
1435:C        2 end;
1436:D     -6 1
1437:D     -6 1 (*****************************************************************************)
1438:D        1 procedure srm_get_vol_date(anyvar f      : datetimerec;
```

```
1439:D        2                                    unum  : unitnum);
1440:D        2 type
1441:D        2   fibptr        = ^fib;
1442:D        2 var
1443:D  -150  2   tempfibspace  : packed array [1..sizeof(fib,0)] of char;
1444:C        2 begin
1445:C        2   with fibptr(addr(tempfibspace))^, packet_ptr.rfileinfo^.file_info do
1446:C        3     begin
1447:C        3       funit     := unum;
1448:C        3       pathid    := -1;
1449:C        3       fileid    := -1;
1450:C        3       fpos      := 0;
1451:C        3       fkind     := datafile;
1452:C        3       feft      := DATATYPE;
1453:C        3       fisnew    := true;
1454:C        3       fanonymous:= true;
1455:C        3       fmodified := false;
1456:C        3       foptstring:= nil;
1457:C        3       fnosrmtemp:= false;
1458:C        3       setstrlen(ftid,0);
1459:C        3       srm_create_file(fibptr(addr(tempfibspace))^,unum);
1460:C        3       if ioresult = ord(inoerror) then
1461:C        4         begin
1462:C        4           fileinfopack(unum,fileid);
1463:C        4           if ioresult = ord(inoerror) then
1464:C        5             with f do
1465:C        6               translatedate(creation_date,date,time);
1466:C        4           srm_purge_file(fibptr(addr(tempfibspace))^,unum);
1467:C        4         end;
1468:C        3     end;
1469:C        2 end;
1470:S
1471:D    -6  1 (*****************************************************************************)
1472:D        1 procedure srm_lock_file(anyvar f        : fib;
1473:D        2                         unum     : unitnum);
1474:C        2 begin
1475:C        2   with f, packet_ptr.rlock^ do
1476:C        3     begin
1477:C        3       lockpack(unum,fileid,fwaitonlock);
1478:C        3       if ioresult = ord(inoerror) then
1479:C        4         if not success.value then
1480:C        5           setioresult(ord(ifilelocked))
1481:C        5         else
1482:C        5           begin
1483:C        5             fileinfopack(unum,fileid);
1484:C        5             if ioresult = ord(inoerror) then
1485:C        6               with packet_ptr.rfileinfo^.file_info do
1486:C        7                 begin
1487:C        7                   fpeof         := logical_eof;
1488:C        7                   fleof         := logical_eof;
1489:C        7                   flocked       := true;
1490:C        7                 end;
1491:C        5           end;
1492:C        3     end;
1493:C        2 end;
1494:S
1495:D    -6  1 (*****************************************************************************)
1496:D        1 procedure srm_unlock_file(anyvar f      : fib;
1497:D        2                           unum   : unitnum);
1498:C        2 begin
```



```
1499:C        2   with f do
1500:C        3     begin
1501:C        3       if ioresult = ord(inoerror) then
1502:C        4         begin
1503:C        4           call(am,addr(f),flush,f,0,0);
1504:C        4           flastpos := -1;
1505:C        4           if ioresult = ord(inoerror) then
1506:C        5             begin
1507:C        5               if fmodified then
1508:C        6                 seteofpack(unum,fileid,false,fleof);
1509:C        5               if ioresult = ord(inoerror) then
1510:C        6                 unlockpack(unum,fileid);
1511:C        5               if ioresult = ord(inoerror) then
1512:C        6                 flocked := false;
1513:C        5             end;
1514:C        4         end;
1515:C        3     end;
1516:C        2 end;
1517:S
1518:D    -6  1 (*****************************************************************************)
1519:D        1 procedure srm_strip(anyvar f : fib);
1520:D        2 var
1521:D  -256  2   s              : string[255];
1522:D  -260  2   findx          : integer;
1523:D  -264  2   sindx          : integer;
1524:D  -268  2   namelen        : integer;
1525:D  -272  2   passlen        : integer;
1526:D  -276  2   i              : integer;
1527:D  -277  2   ch             : char;
1528:D  -278  2   skip           : boolean;
1529:D  -279  2   inpass         : boolean;
1530:D  -280  2   nopassyet      : boolean;
1531:C        2 begin
1532:C        2   namelen        := 0;
1533:C        2   passlen        := 0;
1534:C        2   findx          := 1;
1535:C        2   sindx          := 0;
1536:C        2   setstrlen(s,255);
1537:C        2   inpass         := false;
1538:C        2   nopassyet      := true;
1539:C        2   with f do
1540:C        3     begin
1541:C        3       if ftitle[1] = '<' then   {skip over volume password}
1542:C        4         repeat
1543:C        5           findx := findx + 1;
1544:C        5           if (findx > name_type_len + 3) or (findx > strlen(ftitle)) then
1545:C        6             setioresult(ord(ibadpass));
1546:C        5         until (ftitle[findx-1] = '>') or (ioresult <> ord(inoerror));
1547:S
1548:C        3       while (findx <= strlen(ftitle)) and (ioresult = ord(inoerror)) do
1549:C        4         begin
1550:C        4           skip    := false;
1551:C        4           ch      := ftitle[findx];
1552:S
1553:C        4           if inpass then
1554:C        5             begin
1555:C        5               skip        := true;
1556:C        5               if ch = '>' then
1557:C        6                 inpass    := false
1558:C        6               else
```

```
  1559:C        6                 passlen   := passlen + 1;
  1560:C        5               end
  1561:C        5             else if ch = '/' then
  1562:C        6               begin
  1563:C        6                 nopassyet := true;
  1564:C        6                 inpass    := false;
  1565:C        6                 namelen   := 0;
  1566:C        6                 passlen   := 0;
  1567:C        6               end
  1568:C        6             else if ch = '<' then
  1569:C        7               if nopassyet then
  1570:C        8                 begin
  1571:C        8                   nopassyet := false;
  1572:C        8                   inpass    := true;
  1573:C        8                   skip      := true;
  1574:C        8                 end;
  1575:S
  1576:C        4             if not skip then
  1577:C        5               begin
  1578:C        5                 if ch = '/' then
  1579:C        6                   begin
  1580:C        6                     if s[sindx] = '/' then
  1581:C        7                       setioresult(ord(ibadtitle));
  1582:C        6                   end
  1583:C        6                 else
  1584:C        6                   namelen   := namelen + 1;
  1585:C        5                 sindx       := sindx + 1;
  1586:C        5                 s[sindx]    := ch;
  1587:C        5               end;
  1588:S
  1589:C        4             findx           := findx + 1;
  1590:S
  1591:C        4             if (namelen > name_type_len) then
  1592:C        5               setioresult(ord(ibadtitle))
  1593:C        5             else if (passlen > name_type_len) then
  1594:C        6               setioresult(ord(ibadpass));
  1595:C        4           end;
  1596:C        3
  1597:C        3       if ioresult = ord(inoerror) then
  1598:C        4         begin
  1599:C        4           setstrlen(s,sindx);
  1600:C        4           i := 0;
  1601:C        4           while (s[sindx-i] <> '/') and (i < sindx) do
  1602:C        5             i       := i + 1;
  1603:C        4           if i = 0 then
  1604:C        5             setioresult(ord(ibadtitle))
  1605:C        5           else
  1606:C        5             begin
  1607:C        5               setstrlen(ftid,0);
  1608:C        5               strmove(i,s,sindx-i+1,ftid,1);
  1609:C        5               setstrlen(ftitle,0);
  1610:C        5               strmove(sindx-i,s,1,ftitle,1);
  1611:C        5             end;
  1612:C        4         end;
  1613:C        3       end;
  1614:C        2 end;
  1615:S
  1616:D     -6 1 (*****************************************************************************)
  1617:D        1 procedure srmdaminit;
  1618:C        2 begin
```



```
  1619:C        2   srm_init;
***WARNING: (line 1620): 'ADDR' of a constant may not be supported on other implementations
  1620:C        2   passwordarrayptr := addr(constpassarray);
  1621:C        2 end;
  1622:S
  1623:D     -6 1 (*****************************************************************************)
  1624:D        1 procedure srmdam(anyvar f        : fib;
  1625:D        2                  unum     : unitnum;
  1626:D        2                  request : damrequesttype);
  1627:D        2 var
  1628:D     -4 2   holdpathid            : integer;
  1629:D     -8 2   savepathid            : integer;
  1630:D    -12 2   savefileid            : integer;
  1631:D    -30 2   saveftid              : tid;
  1632:D    -48 2   saveffpw              : passtype;
  1633:D    -66 2   savefvid              : vid;
  1634:D   -188 2   saveftitle            : fid;
  1635:D   -189 2   savefsavepathid       : boolean;
  1636:D   -190 2   fisafib               : boolean;
  1637:C        2 begin
  1638:C        2   ioresult      := ord(inoerror);
  1639:C        2   srmsavesc     := 0;
  1640:C        2   lockup;
  1641:C        2   fisafib       := false;
  1642:C        2   try
  1643:C        3     with f, unitable^[unum] do
  1644:C        4       if offline then
  1645:C        5         ioresult        := ord(znodevice)
  1646:C        5       else
  1647:C        5         begin
  1648:C        5           if request in [opendirectory,
  1649:C        6                          openparentdir,
  1650:C        6                          closedirectory,
  1651:C        6                          catalog,
  1652:C        6                          catpasswords,
  1653:C        6                          setpasswords,
  1654:C        6                          openfile,
  1655:C        6                          createfile,
  1656:C        6                          overwritefile,
  1657:C        6                          makedirectory,
  1658:C        6                          closefile,
  1659:C        6                          changename,
  1660:C        6                          duplicatelink,
  1661:C        6                          purgename,
  1662:C        6                          lockfile,
  1663:C        6                          unlockfile,
  1664:C        6                          purgefile,
  1665:C        6                          setunitprefix,
  1666:C        6                          stretchit              ] then          {f is a fib}
  1667:C        6             begin
  1668:C        6               fisafib   := true;
  1669:C        6               if strlen(ftid) > tidleng then    {fix uninitialized fib strings}
  1670:C        7                 setstrlen(ftid,0);
  1671:C        6               if strlen(ffpw) > passleng then
  1672:C        7                 setstrlen(ffpw,0);
  1673:C        6               if strlen(fvid) > vidleng then
  1674:C        7                 setstrlen(fvid,0);
  1675:S
  1676:C        6               savepathid        := pathid;    {save fib fields to be restored on error}
  1677:C        6               savefileid        := fileid;
```

```
1678:C  6               savefsavepathid   := fsavepathid;
1679:C  6               saveftid          := ftid;
1680:C  6               saveffpw          := ffpw;
1681:C  6               savefvid          := fvid;
1682:C  6               if strlen(ftitle) > fidleng then
1683:C  7                 setstrlen(saveftitle,0)
1684:C  7               else
1685:C  7                 saveftitle      := ftitle;
1686:C  6             end;
1687:C  5           case request of
1688:C  6             opendirectory,
1689:C  6             openparentdir : begin
1690:C  6                               srm_open_dir(f,unum,open_directory,request = openparentdir);
1691:C  6                               if ioresult = ord(inoerror) then
1692:C  7                                 srm_get_dir_info(fwindow^,pathid,unum,true,false);
1693:C  6                             end;
1694:S
1695:C  6             closedirectory : begin
1696:C  6                                fsavepathid       := false;
1697:C  6                               srm_close_pathid(unum,pathid,false);
1698:C  6                              end;
1699:S
1700:C  6             catalog       : srm_catalog(f,unum);
1701:S
1702:C  6             catpasswords  : begin
1703:C  6                               srm_open_dir(f,unum,open_directory,true);
1704:C  6                               if ioresult = ord(inoerror) then
1705:C  7                                 begin
1706:C  7                                   srm_cat_pass(f,unum);
1707:C  7                                   srm_close_pathid(unum,pathid,fsavepathid);
1708:C  7                                 end;
1709:C  6                             end;
1710:S
1711:C  6             setpasswords  : begin
1712:C  6                               srm_open_dir(f,unum,open_directory,true);
1713:C  6                               if ioresult = ord(inoerror) then
1714:C  7                                 begin
1715:C  7                                   srm_set_pass(f,unum);
1716:C  7                                   srm_close_pathid(unum,pathid,fsavepathid);
1717:C  7                                 end;
1718:C  6                             end;
1719:S
1720:C  6             openfile      : begin
1721:C  6                               fisnew            := false;
1722:C  6                               fnosrmtemp        := true;      {default case}
1723:C  6                               srm_open_dir(f,unum,open_directory,true );
1724:C  6                               if ioresult = ord(inoerror) then
1725:C  7                                 begin
1726:C  7                                   srm_open_file(f,unum);
1727:C  7                                   if ioresult <> ord(inoerror) then
1728:C  8                                     srm_close_pathid(unum,pathid,fsavepathid);
1729:C  7                                 end;
1730:C  6                             end;
1731:S
1732:C  6             createfile,
1733:C  6             overwritefile : begin
1734:C  6                               fisnew            := true;
1735:C  6                               fnosrmtemp        := false;
1736:C  6                               foverwritten      := request = overwritefile;
1737:C  6                               if not fanonymous then
```



```
1738:C  7                                 srm_open_dir(f,unum,open_directory,true );
1739:C  6                               if ioresult = ord(inoerror) then
1740:C  7                                 begin
1741:C  7                                   srm_create_file(f,unum);
1742:C  7                                   if ioresult <> ord(inoerror) then
1743:C  8                                     if not fanonymous then
1744:C  9                                       srm_close_pathid(unum,pathid,fsavepathid);
1745:C  7                                 end;
1746:C  6                             end;
1747:S
1748:C  6             makedirectory : begin
1749:C  6                               holdpathid  := pathid;
1750:C  6                               srm_open_dir(f,unum,open_directory,false);
1751:C  6                               if ioresult = ord(inoerror) then
1752:C  7                                 begin
1753:C  7                                   srm_create_dir (f,unum);
1754:C  7                                   srm_close_pathid(unum,pathid,fsavepathid);
1755:C  7                                 end;
1756:C  6                               pathid      := holdpathid;
1757:C  6                             end;
1758:S
1759:C  6             closefile     : if (fisnew and fanonymous) then
1760:C  7                               srm_purge_file(f,unum)
1761:C  7                             else
1762:C  7                               srm_close_file(f,unum);
1763:S
1764:C  6             changename    : srm_change_name(f,unum);
1765:S
1766:C  6             duplicatelink : srm_dup_link(f,unum);
1767:S
1768:C  6             purgename     : begin
1769:C  6                               holdpathid  := pathid;
1770:C  6                               srm_open_dir(f,unum,open_directory,true);
1771:C  6                               if ioresult = ord(inoerror) then
1772:C  7                                 begin
1773:C  7                                   srm_purge_name(f,unum);
1774:C  7                                   srm_close_pathid(unum,pathid,fsavepathid);
1775:C  7                                 end;
1776:C  6                               pathid      := holdpathid;
1777:C  6                             end;
1778:S
1779:C  6             lockfile      : srm_lock_file(f,unum);
1780:S
1781:C  6             unlockfile    : srm_unlock_file(f,unum);
1782:S
1783:C  6             purgefile     : srm_purge_file(f,unum);
1784:S
1785:C  6             setunitprefix : srm_set_unit_prefix(f,unum);
1786:S
1787:C  6             stretchit     : srm_stretch(f,unum);
1788:S
1789:C  6             getvolumename : srm_get_vol_name(f,unum);
1790:S
1791:C  6             getvolumedate : srm_get_vol_date(f,unum);
1792:S
1793:C  6             setvolumedate,
1794:C  6             crunch        : {do nothing, but no error};
1795:S
1796:C  6             stripname     : srm_strip(f);
1797:S
```

```
1798:C        6              otherwise       setioresult(ord(ibadrequest));
1799:C        6            end;
1800:S
1801:C        5            if (ioresult <> ord(inoerror)) and fisafib then {restore fib for subsequent calls}
1802:C        6              begin
1803:C        6                pathid        := savepathid;
1804:C        6                fileid        := savefileid;
1805:C        6                fsavepathid   := savefsavepathid;
1806:C        6                ftid          := saveftid;
1807:C        6                ffpw          := saveffpw;
1808:C        6                fvid          := savefvid;
1809:C        6                if strlen(saveftitle) > 0 then
1810:C        7                  ftitle      := saveftitle;
1811:C        6              end;
1812:C        5            if ioresult = ord(isrmcatchall) then
1813:C        6              if srmsavesc <> 0 then
1814:C        7                escape(srmsavesc);
1815:C        5          end;
1816:C        3    recover
1817:C        3      begin
1818:C        3        if escapecode = ioescapecode then
1819:C        4          setioresult(ord(isrmcatchall))
1820:C        4        else
1821:C        4          begin
1822:C        4            lockdown;
1823:C        4            escape(escapecode);
1824:C        4          end;
1825:C        3      end;
1826:C        2    lockdown;
1827:C        2  end;
1828:S
1829:C        1  end; {srmdammodule}
1830:S
1831:D        1  import
1832:D        1    srmdammodule;
1833:C        1  begin   {program init_srm}
1834:C        1    srmdaminit;
1835:C        1  end.
1836:S

No errors. 3 warnings.
             ***** Nonstandard language features enabled *****
```

Description

SRM_DRV is the low-level support for SRMAM and SRMDAM. Primarily, it serves as an SRM packet constructor, sender and receiver.

Requirements

The I/O library drivers for the 98629 card (DATA_COMM).

```
 1:S                                                    (*
 2:S
 3:S                  (c) Copyright Hewlett-Packard Company, 1983.
 4:S                 All rights are reserved.  Copying or other
 5:S                 reproduction of this program except for archival
 6:S                 purposes is prohibited without the prior
 7:S                 written consent of Hewlett-Packard Company.
 8:S
 9:S
10:S                             RESTRICTED RIGHTS LEGEND
11:S
12:S                 Use, duplication, or disclosure by the Government
13:S                 is subject to restrictions as set forth in
14:S                 paragraph (b) (3) (B) of the Rights in Technical
15:S                 Data and Computer Software clause in
16:S                 DAR 7-104.9(a).
17:S
18:S                 HEWLETT-PACKARD COMPANY
19:D               0 Fort Collins, Colorado                        *)
20:S
21:D               0 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'$
22:D               0 $range off$
23:D               0 $debug off$
24:D               0 $modcal$
25:D               0 $sysprog$
26:D               0 $ucsd$
27:D               0 module srm;
28:D               1 $search 'IOLIB:KERNEL'$
29:D               1 import iodeclarations,
30:D               1        sysglobals,
31:D               1        loader;
32:D               1 export
33:S
34:D               1 $include 'INIT:SRM_TYPES'$
35:D               1 type
36:S                   (
37:D               1   byte          = 0..255;
38:D               1   halfword      = -32768..32767;
39:S
40:D               1 $include 'PACKETS'$
41:S                 {*****************************************************************************
42:S                  * +-------------------------------------------------------------------------+ *
43:S                  * |                                                                         | *
44:S                  * |        G A N G L I A   M E S S A G E   F O R M A T S                     | *
45:S                  * |                                                                         | *
46:S                  * |        A N D   R E L A T E D   D E F I N I T I O N S                     | *
47:S                  * |                                                                         | *
48:D               1  * +-------------------------------------------------------------------------+ *
49:D               1  *****************************************************************************}
50:D               1
51:D               1 CONST
52:D               1
53:S                     name_type_len = 16;
54:S
55:S                     {+-----------------------------------------------------------------------+
56:D               1      | Message Packet Sizes For Messages That Are Sent To Ganglia            |
57:D               1      +-----------------------------------------------------------------------+}
58:D               1     size_to_are_you_alive =     12;
```



```
59:D               1     size_to_catprotect =       128;   (* plus 36 * # file name sets *)
60:S                     size_to_changeprotect =    116;   (* plus (36 * # file name sets) +
61:D               1                                               (24 * # file name sets) *)
62:D               1     size_to_change_vol_label= 132;
63:D               1     size_to_close =            56;
64:D               1     size_to_copy =             32;
65:S                     size_to_create =           152;   (* plus (36 * # file name sets) +
66:D               1                                               (24 * # protect code sets)  *)
67:D               1     size_to_createlink =       144;   (* plus 36 * # file name sets *)
68:D               1     size_to_flock =            20;
69:D               1     size_to_funlock =          20;
70:D               1     size_to_gang_cleanup =     16;
71:D               1     size_to_info =             20;
72:D               1     size_to_init_vol =         172;   (* plus 24 * # protect code sets *)
73:D               1     size_to_open =             132;   (* plus 36 * # file name sets *)
74:D               1     size_to_position =         28;
75:D               1     size_to_purge =            112;   (* plus 36 * # file name sets *)
76:D               1     size_to_read =             40;
77:D               1     size_to_set_eof =          28;
78:D               1     size_to_set_date =         44;
79:D               1     size_to_volstatus =        84;
80:D               1     size_to_write =            48;    (* plus the size of the record <= 512 *)
81:D               1     size_to_xchg_data =        140;   (* plus 36 * # file name sets *)
82:D               1     size_to_xchg_open =        20;
83:D               1
84:D               1
85:S                     {+-----------------------------------------------------------------------+
86:S                      | Message Packet Sizes For Messages Ganglia Will Send Back              |
87:D               1      +-----------------------------------------------------------------------+}
88:D               1
89:D               1     size_from_are_you_alive  = 16;
90:D               1     size_from_cat =            24; (* plus 68 * # entries returned *)
91:D               1     size_from_catprotect =     24; ( plus 20 * # prot code entries returned )
92:D               1     size_from_changeprotect =  16;
93:D               1     size_from_change_vol_label=16;
94:D               1     size_from_close =          16;
95:D               1     size_from_copy =           20;
96:D               1     size_from_create =         16;
97:D               1     size_from_createlink =     16;
98:D               1     size_from_flock =          20;
99:D               1     size_from_funlock =        16;
100:D              1     size_from_gang_cleanup =   16;
101:D              1     size_from_gang_error =     16;
102:D              1     size_from_info =           88;
103:D              1     size_from_init_vol =       16;
104:D              1     size_from_open =           52;
105:D              1     size_from_position =       16;
106:D              1     size_from_purge =          16;
107:D              1     size_from_read =           36;    (* plus the size of the record <= 512 *)
108:D              1     size_from_set_eof =        16;
109:D              1     size_from_set_date =       16;
110:D              1     size_from_volstatus =      40;
111:D              1     size_from_write =          20;
112:D              1     size_from_xchg_data =      16;
113:D              1     size_from_xchg_open =      16;
114:D              1
115:D              1
116:S                    {+-----------------------------------------------------------------------+
117:S                     | Ganglia Message Request Types, Modes, And Codes                       |
118:D              1      +-----------------------------------------------------------------------+}
```

```
   119:D        1
   120:D        1    {*** File Access Method Requests ***}
   121:D        1    req_old_read          =0;  {Old read packet}
   122:D        1    req_write             =1;  {Write a record to an open file.}
   123:D        1    req_position          =2;  {Position an open file.}
   124:D        1    req_read              =3;  {Read a record from an open file.}
   125:D        1    req_set_eof           =4;
   126:D        1
   127:D        1    {*** Sharing Requests ***}
   128:D        1    req_flock             =8;  {Lock an open file for exclusive use.}
   129:D        1    req_funlock           =9;  {Unlock an open file.}
   130:D        1    req_info              =10;
   131:D        1    req_close             =13; {Close an open file.}
   132:D        1
   133:D        1    {*** Directory Access Requests ***}
   134:D        1    req_open              =14;
   135:D        1    req_purgelink         =15;
   136:D        1    req_catalog           =16; {List all the files in a directory.}
   137:D        1    req_create            =17;
   138:D        1    req_createlink        =18;
   139:D        1    req_changeprotect     =19; {Change the passwords on a file.}
   140:D        1    req_catprotect        =20; {List all the passwords on a file.}
   141:D        1    req_xchg_data         =21;
   142:S
   143:D        1    {*** Volume Access Requests ***}
   144:D        1    req_volstatus         =22;
   145:D        1    req_init_vol          =23;
   146:D        1    req_label             =24;
   147:D        1
   148:D        1    {*** File Transfer Requests ***}
   149:D        1    req_xchg_open         =29;
   150:D        1    req_copy              =30;
   151:S
   152:D        1    {*** Ganglia Requests ***}
   153:D        1    req_gang_cleanup      =1000;
   154:D        1    req_are_you_alive     =1001;
   155:D        1
   156:D        1
   157:D        1    {*** Record Modes ***}
   158:D        1    data_records =        0;
   159:D        1    directory_records =   1;
   160:D        1
   161:D        1    {*** Share Codes ***}
   162:D        1    exclusive_share_code = 0;
   163:D        1    shared_share_code    = 1;
   164:D        1    closed_share_code    = 2;
   165:D        1    corrupt_share_code   = 3;
   166:S
   167:D        1    {*** Access Codes ***}
   168:D        1    random_access =       0;
   169:D        1    sequential_access =   1;
   170:D        1    primitive_access =    2;
   171:D        1
   172:D        1
   173:D        1  TYPE
   174:D        1
   175:S             {+-------------------------------------------------------------------------+
   176:S              | Enumerated Types And Minor Structures Used To Make Up More              |
   177:S              | Major Structures Described In The Next Section                          |
   178:D        1     +-------------------------------------------------------------------------+}
```



```
   179:D        1
   180:D        1    access_code_type =
   181:D        1       (ac_manager,
   182:D        1        ac_read,
   183:D        1        ac_write,
   184:D        1        ac_search,
   185:D        1        ac_purgelink,
   186:D        1        ac_createlink,
   187:D        1        ac_execute,
   188:D        1        ac_generic1,ac_generic09,ac_generic17,
   189:D        1        ac_generic2,ac_generic10,ac_generic18,
   190:D        1        ac_generic3,ac_generic11,ac_generic19,
   191:D        1        ac_generic4,ac_generic12,ac_generic20,
   192:D        1        ac_generic5,ac_generic13,ac_generic21,
   193:D        1        ac_generic6,ac_generic14,ac_generic22,
   194:D        1        ac_generic7,ac_generic15,ac_generic23,
   195:D        1        ac_generic8,ac_generic16,ac_generic24,
   196:D        1        ac_generic25);
   197:D        1
   198:D        1    file_id_type = integer;
   199:D        1
   200:D        1    {*** Filler Definitions Used In Padding Out Message Packet Layouts ***}
   201:D        1    gang_boolean_filler = boolean;
   202:D        1    gang_32bit_filler = integer;
   203:D        1    gang_16bit_filler = -32768..32767;
   204:D        1
   205:D        1    long_boolean = record case integer of
   206:D        1         1:      (i      : integer);
   207:D        1         2:      (fill1: gang_16bit_filler;
   208:D        1                  fill2: gang_boolean_filler;
   209:D        1                  value: boolean);
   210:D        1       end;
   211:S
   212:D        1    gang_file_codes = packed record
   213:D        1       case integer of
   214:D        1       0 :  (i: integer);
   215:D        1       1 :  (si1: shortint;
   216:D        1             si2: shortint);
   217:D        1       2 : (firstword: -32768..32767;
   218:D        1            division_code: 0..63;
   219:D        1            dcd_system_type: 0..15;
   220:D        1            dcd_file_type: 0..63);
   221:S
   222:D        1    end;
   223:D        1
   224:D        1    gang_open_type =
   225:D        1       (open_data,                 {Normal data file}
   226:D        1        open_protected_directory, {Directory not closed on cleanup}
   227:D        1        open_directory            {Directory closed on cleanup}
   228:D        1       );
   229:D        1
   230:D        1    name_type = packed array [1..name_type_len] of char;
   231:D        1
   232:D        1    path_start_type =
   233:D        1       (start_root,             {Start search at root directory}
   234:D        1        start_alternate         {Start search at a working directory.}
   235:D        1       );
   236:D        1
   237:D        1    position_type =
   238:D        1       (pos_absolute,      {Position at absolute location in file)
```

```
239:D  1        pos_seq            {Position relative to current record}
240:D  1       );
241:D  1
242:D  1
243:S          {+-----------------------------------------------------------------------+
244:S           | Data Structures Which Are Part Of Message Packet Layouts              |
245:D  1        +-----------------------------------------------------------------------+}
246:D  1
247:D  1    access_capabilities = packed array [ access_code_type ] of boolean;
248:D  1
249:D  1    date_type = record
250:D  1       fill: gang_16bit_filler;
251:D  1       date: packed record
252:D  1                month: 0..12;
253:D  1                day: 0..31;
254:D  1                year: 0..100;
255:D  1             end;
256:D  1       seconds_since_midnight : integer
257:D  1    end;
258:D  1
259:D  1    device_address_type = record
260:D  1       address1   : integer;
261:D  1       haddress   : integer;
262:D  1       unit_num   : integer;
263:D  1       volume_num : integer
264:D  1    end;
265:D  1
266:D  1    file_header_type = record
267:D  1       num_file_name_sets : integer;
268:D  1       working_directory  : file_id_type;
269:D  1       filler1            : gang_16bit_filler;
270:D  1       path_type          : path_start_type;
271:D  1       root_password      : name_type;
272:D  1    end;
273:D  1
274:D  1    file_info_type = record
275:D  1       file_name          : name_type;
276:D  1       open_flag          : long_boolean;
277:D  1       share_code         : integer;
278:D  1       file_code          : gang_file_codes;
279:D  1       record_mode        : integer;
280:D  1       max_record_size    : integer;
281:D  1       max_file_size      : integer;
282:D  1       creation_date      : date_type;
283:D  1       last_access_date   : date_type;
284:D  1       capabilities       : access_capabilities;
285:D  1       logical_eof        : integer;
286:D  1       physical_size      : integer;
287:D  1    end;
288:S
289:D  1    file_name_set = record
290:D  1       file_name : name_type;
291:D  1       password  : name_type;
292:D  1       filler    : gang_32bit_filler
293:D  1    end;
294:D  1
295:D  1    linkfillertype = packed record
296:D  1       requeue:           boolean;
297:D  1       zit1:              byte;       { unused }
298:D  1       zit2:              integer;    { unused }
```



```
299:D  1       oddbytefiller:    byte;       { unused }
300:D  1       destaddr:         byte;
301:D  1       sourceaddr:       byte;   { These three bytes     }
302:D  1       len_lobyte:       byte;   { are filled in by      }
303:D  1       len_hibyte:       byte;   { the transmitting card }
304:D  1       level:            byte;
305:D  1    end;
306:D  1
307:D  1    msg_packet_type = packed array [1..800 (or so) ] of char;
308:D  1    msg_packet_ptr = ^msg_packet_type;
309:D  1
310:D  1    owner_id_type = record
311:D  1       id : integer;
312:D  1    end;
313:D  1
314:D  1    protect_code_set = record
315:D  1       capabilities   : access_capabilities;
316:D  1       password       : name_type;
317:D  1       filler         : gang_32bit_filler
318:D  1    end;
319:D  1
320:D  1    return_header_type = record
321:D  1       linkfiller          : linkfillertype;
322:D  1       message_length      : integer;
323:D  1       return_request_type : integer;
324:D  1       user_sequencing_field : integer;
325:D  1       status              : integer
326:D  1    end;
327:D  1
328:D  1    send_header_type = record
329:D  1       linkfiller          : linkfillertype;
330:D  1       message_length      : integer;
331:D  1       send_request_type   : integer;
332:D  1       user_sequencing_field : integer
333:D  1    end;
334:D  1
335:D  1    volume_header_type = record
336:D  1       filler1               : gang_32bit_filler;
337:D  1       driver_name           : name_type;
338:D  1       catalogue_organization : name_type;
339:D  1       device_address_present : long_boolean;
340:D  1       device_address        : device_address_type;
341:D  1       volume_name           : name_type;
342:D  1    end;
343:D  1
344:D  1    volume_info_type = record
345:D  1       free_blocks   : integer;
346:D  1       bad_blocks    : integer;
347:D  1       media_origin  : integer;
348:D  1       interleave    : integer;
349:D  1       volume_label  : name_type;
350:D  1    end;
351:D  1
352:D  1
353:D  1
354:S          {+-----------------------------------------------------------------------+
355:S           | Send and return record layouts for request: Are You Alive ?!          |
356:D  1        +-----------------------------------------------------------------------+}
357:D  1
358:D  1    send_are_you_alive = record
```

```
   359:D        1       send_mess_header   : send_header_type;
   360:D        1    end;
   361:D        1
   362:D        1    return_are_you_alive = record
   363:D        1       return_mess_header : return_header_type;
   364:D        1    end;
   365:D        1
   366:D        1
   367:S             {+-------------------------------------------------------------------------+
   368:S              | Send and return record layouts for request: Catalogue                   |
   369:D        1     +-------------------------------------------------------------------------+}
   370:D        1
   371:D        1    send_catalogue = record
   372:D        1       send_mess_header   : send_header_type;
   373:D        1       max_num_files      : integer;
   374:D        1       file_index         : integer;
   375:D        1       filler1            : gang_32bit_filler; {actual_num_files}
   376:D        1       volume_name_header : volume_header_type;
   377:D        1       file_name_header   : file_header_type;
   378:D        1       filler2            : gang_32bit_filler;
   379:D        1       start_name_sets    : integer
   380:D        1    end;
   381:D        1
   382:D        1    return_catalogue = record
   383:D        1       return_mess_header : return_header_type;
   384:D        1       filler1            : gang_32bit_filler; {file_index}
   385:D        1       actual_num_files   : integer;
   386:D        1       cat_info           : array [1..7] of file_info_type
   387:D        1    end;
   388:D        1
   389:D        1
   390:S             {+-------------------------------------------------------------------------+
   391:S              | Send and return record layouts for request: Catalogue Protect Codes     |
   392:D        1     +-------------------------------------------------------------------------+}
   393:D        1
   394:D        1    send_cat_passwords = record
   395:D        1       send_mess_header        : send_header_type;
   396:D        1       max_num_passwords       : integer;
   397:D        1       filler1                 : gang_32bit_filler;  {actual_passwords}
   398:D        1       password_index          : integer;
   399:D        1       volume_name_header      : volume_header_type;
   400:D        1       file_name_header        : file_header_type;
   401:D        1       filler2                 : gang_32bit_filler;
   402:D        1       start_name_sets         : integer
   403:D        1    end;
   404:D        1
   405:D        1    return_cat_passwords = record
   406:D        1       return_mess_header   : return_header_type;
   407:D        1       actual_num_passwords : integer;
   408:D        1       filler1              : gang_32bit_filler;
   409:D        1       password_info        : array [1..24] of record
   410:D        1                                 password     : name_type;
   411:D        1                                 capabilities : access_capabilities
   412:D        1                              end
   413:D        1    end;
   414:D        1
   415:D        1
   416:S             {+-------------------------------------------------------------------------+
   417:S              | Send and return record layouts for request: Change Protect Codes        |
   418:D        1     +-------------------------------------------------------------------------+}
```



```
   419:D        1
   420:D        1    send_change_protect_codes = record
   421:D        1       send_mess_header        : send_header_type;
   422:D        1       volume_name_header      : volume_header_type;
   423:D        1       file_name_header        : file_header_type;
   424:D        1       num_protect_code_sets   : integer;
   425:D        1       start_name_sets         : integer
   426:D        1    end;
   427:D        1
   428:D        1    return_change_protect_codes = record
   429:D        1       return_mess_header : return_header_type
   430:D        1    end;
   431:D        1
   432:D        1
   433:S             {+-------------------------------------------------------------------------+
   434:S              | Send and return record layouts for request: Change Volume Label         |
   435:D        1     +-------------------------------------------------------------------------+}
   436:D        1
   437:D        1    send_change_volume_label = record
   438:D        1       send_mess_header        : send_header_type;
   439:D        1       volume_name_header      : volume_header_type;
   440:D        1       password                : name_type;
   441:D        1       new_volume_name         : name_type;
   442:D        1       new_vol_password        : name_type;
   443:D        1    end;
   444:D        1
   445:D        1    return_change_volume_label = record
   446:D        1       return_mess_header : return_header_type
   447:D        1    end;
   448:D        1
   449:D        1
   450:S             {+-------------------------------------------------------------------------+
   451:S              | Send and return record layouts for request: Close A File                |
   452:D        1     +-------------------------------------------------------------------------+}
   453:D        1
   454:D        1    send_closeafile = record
   455:D        1       send_mess_header   : send_header_type;
   456:D        1       file_id            : file_id_type;
   457:D        1       directory_password : name_type;
   458:D        1       file_password      : name_type;
   459:D        1       filler5            : long_boolean;
   460:D        1       nodeallocate       : long_boolean;
   461:D        1    end;
   462:D        1
   463:D        1    return_closeafile = record
   464:D        1       return_mess_header : return_header_type
   465:D        1    end;
   466:D        1
   467:D        1
   468:S             {+-------------------------------------------------------------------------+
   469:S              | Send and return record layouts for request: Copy File To File           |
   470:D        1     +-------------------------------------------------------------------------+}
   471:D        1
   472:D        1    send_copy = record
   473:D        1       send_mess_header       : send_header_type;
   474:D        1       source_file_id         : file_id_type;
   475:D        1       source_offset          : integer;
   476:D        1       destination_file_id    : file_id_type;
   477:D        1       destination_offset     : integer;
   478:D        1       requested              : integer;
```

```
479:D        1    end;
480:D        1
481:D        1    return_copy = record
482:D        1       return_mess_header   : return_header_type;
483:D        1       actual               : integer;
484:D        1    end;
485:D        1
486:D        1
487:S             {+-------------------------------------------------------------------+
488:S              | Send and return record layouts for request: Create A File         |
489:D        1     +-------------------------------------------------------------------+}
490:D        1
491:D        1    send_createafile = record
492:D        1       send_mess_header         : send_header_type;
493:D        1       volume_name_header       : volume_header_type;
494:D        1       file_name_header         : file_header_type;
495:D        1       file_code                : gang_file_codes;
496:D        1       record_mode              : integer;
497:D        1       max_record_size          : integer;
498:D        1       first_extent             : integer;    {Size in logical records.}
499:D        1       contiguous_first_extent  : long_boolean;
500:D        1       secondary_extent         : integer;
501:D        1       max_file_size            : integer;
502:D        1       boot_start_address       : integer;
503:D        1       num_protect_code_sets    : integer;
504:D        1       label_included_flag      : long_boolean;
505:D        1       start_name_sets          : integer
506:D        1    end;
507:D        1
508:D        1    return_createafile = record
509:D        1       return_mess_header : return_header_type
510:D        1    end;
511:D        1
512:D        1
513:S             {+-------------------------------------------------------------------+
514:S              | Send and return record layouts for request: Create A Link (Rename)|
515:D        1     +-------------------------------------------------------------------+}
516:D        1
517:D        1    send_createlink = record
518:D        1       send_mess_header      : send_header_type;
519:D        1       volume_name_header    : volume_header_type;
520:D        1       old_file_name_header  : file_header_type;
521:D        1       new_file_name_header  : file_header_type;
522:D        1       purge_old_link        : long_boolean;
523:D        1       start_name_sets       : integer
524:D        1    end;
525:D        1
526:D        1    return_createlink = record
527:D        1       return_mess_header    : return_header_type;
528:D        1    end;
529:D        1
530:D        1
531:S             {+-------------------------------------------------------------------+
532:S              | Send and return record layouts for request: Exchange files        |
533:D        1     +-------------------------------------------------------------------+}
534:D        1
535:D        1    send_xchg_data = record
536:D        1       send_mess_header      : send_header_type;
537:D        1       volume_name_header    : volume_header_type;
538:D        1       file_name_header1     : file_header_type;
```



```
539:D        1       file_name_header2     : file_header_type;
540:D        1       start_name_sets       : integer;
541:D        1    end;
542:D        1
543:D        1    return_xchg_data = record
544:D        1       return_mess_header    : return_header_type;
545:D        1    end;
546:D        1
547:D        1
548:D        1
549:S             {+-------------------------------------------------------------------+
550:S              | Send and return record layouts for request: Exchange open files   |
551:D        1     +-------------------------------------------------------------------+}
552:D        1
553:D        1    send_xchg_open = record
554:D        1       send_mess_header      : send_header_type;
555:D        1       file_id_1             : file_id_type;
556:D        1       file_id_2             : file_id_type;
557:D        1    end;
558:D        1
559:D        1    return_xchg_open = record
560:D        1       return_mess_header    : return_header_type;
561:D        1    end;
562:D        1
563:D        1
564:D        1
565:S             {+-------------------------------------------------------------------+
566:S              | Send and return record layouts for request: File Information      |
567:D        1     +-------------------------------------------------------------------+}
568:D        1
569:D        1    send_fileinfo = record
570:D        1       send_mess_header  : send_header_type;
571:D        1       implicit_unlock   : long_boolean;
572:D        1       file_id           : file_id_type
573:D        1    end;
574:D        1
575:D        1    return_fileinfo = record
576:D        1       return_mess_header : return_header_type;
577:D        1       current_record     : integer;
578:D        1       file_info          : file_info_type;
579:D        1    end;
580:D        1
581:D        1
582:D        1
583:S             {+-------------------------------------------------------------------+
584:S              | Send and return record layouts for request: Ganglia Cleanup       |
585:D        1     +-------------------------------------------------------------------+}
586:D        1
587:D        1    send_gang_cleanup = record
588:D        1       send_mess_header          : send_header_type;
589:D        1       keep_protected_directories: long_boolean;
590:D        1    end;
591:D        1
592:D        1    return_gang_cleanup = record
593:D        1       return_mess_header : return_header_type
594:D        1    end;
595:D        1
596:D        1
597:D        1
598:S             {+-------------------------------------------------------------------+
```

```
599:S                | Send and return record layouts for request: Initialize A Volume      |
600:D        1       +-------------------------------------------------------------------------+)
601:D        1
602:D        1    send_initialize = record
603:D        1       send_mess_header         : send_header_type;
604:D        1       volume_name_header       : volume_header_type;
605:D        1       password                 : name_type;
606:D        1       num_protect_code_sets    : integer;
607:D        1       new_volume_name          : name_type;
608:D        1       new_cat_organization     : name_type;
609:D        1       sectors_per_block        : integer;
610:D        1       format_media             : long_boolean;
611:D        1       interleave_factor        : integer;
612:D        1       filler3                  : gang_32bit_filler;
613:D        1       physical_sector_size     : integer;
614:D        1       new_password             : {global_}name_type;
615:D        1       start_protect_code_sets  : integer;
616:D        1       init_foot_password       : {global_}name_type;
617:D        1    end;
618:D        1
619:D        1    return_initialize = record
620:D        1       return_mess_header : return_header_type
621:D        1    end;
622:D        1
623:D        1
624:S                {+-------------------------------------------------------------------------+
625:S                 | Send and return record layouts for request: Lock A File                 |
626:D        1        +-------------------------------------------------------------------------+)
627:D        1
628:D        1    send_lockafile = record
629:D        1       send_mess_header : send_header_type;
630:D        1       file_id          : file_id_type;
631:D        1       wait_for_lock    : long_boolean
632:D        1    end;
633:D        1
634:D        1    return_lockafile = record
635:D        1       return_mess_header : return_header_type;
636:D        1       success            : long_boolean
637:D        1    end;
638:D        1
639:D        1
640:S                {+-------------------------------------------------------------------------+
641:S                 | Send and return record layouts for request: Open A File                 |
642:D        1        +-------------------------------------------------------------------------+)
643:D        1
644:D        1    send_openafile = record
645:D        1       send_mess_header       : send_header_type;
646:D        1       volume_name_header     : volume_header_type;
647:D        1       file_name_header       : file_header_type;
648:D        1       filler2                : integer;
649:D        1       filler3                : integer;
650:D        1       share_code             : integer;
651:D        1       filler4                : owner_id_type;
652:D        1       filler1                : gang_16bit_filler;
653:D        1       open_type              : gang_open_type;
654:D        1       start_name_sets        : integer
655:D        1    end;
656:D        1
657:D        1    return_openafile = record
658:D        1       return_mess_header : return_header_type;
```



```
659:D        1       file_id             : file_id_type;
660:D        1       record_mode         : integer;
661:D        1       max_record_size     : integer;
662:D        1       max_file_size       : integer;
663:D        1       file_code           : gang_file_codes;
664:D        1       open_logical_eof    : integer;
665:D        1       sharebits           : integer;
666:D        1       sec_ext_size        : integer;
667:D        1       boot_start_address  : integer;
668:D        1    end;
669:D        1
670:D        1
671:S                {+-------------------------------------------------------------------------+
672:S                 | Send and return record layouts for request: Position To A Record        |
673:D        1        +-------------------------------------------------------------------------+)
674:D        1
675:D        1    send_positiontoarecord = record
676:D        1       send_mess_header : send_header_type;
677:D        1       implicit_unlock  : long_boolean;
678:D        1       file_id          : file_id_type;
679:D        1       filler3          : gang_16bit_filler;
680:D        1       type_of_position : position_type;
681:D        1       byte_offset      : integer
682:D        1    end;
683:D        1
684:D        1    return_positiontoarecord = record
685:D        1       return_mess_header : return_header_type
686:D        1    end;
687:D        1
688:D        1
689:S                {+-------------------------------------------------------------------------+
690:S                 | Send and return record layouts for request: Purge Link                  |
691:D        1        +-------------------------------------------------------------------------+)
692:D        1
693:D        1    send_purgelink = record
694:D        1       send_mess_header   : send_header_type;
695:D        1       volume_name_header : volume_header_type;
696:D        1       file_name_header   : file_header_type;
697:D        1       start_name_sets    : integer
698:D        1    end;
699:D        1
700:D        1    return_purgelink = record
701:D        1       return_mess_header : return_header_type
702:D        1    end;
703:D        1
704:D        1
705:S                {+-------------------------------------------------------------------------+
706:S                 | Send and return record layouts for request:  Read A Record              |
707:D        1        +-------------------------------------------------------------------------+)
708:D        1
709:D        1    send_readarecord = record
710:D        1       send_mess_header  : send_header_type;
711:D        1       implicit_unlock   : long_boolean;
712:D        1       file_id           : file_id_type;
713:D        1       access_code       : integer;
714:D        1       filler3           : array [1..2] of gang_32bit_filler;
715:D        1       requested         : integer;
716:D        1       offset            : integer
717:D        1    end;
718:D        1
```

```
    return_readarecord = record
       return_mess_header  : return_header_type;
       actual              : integer;
       filler1             : array [1..4] of gang_32bit_filler;
       data                : packed array[1..512] of char;
    end;


    {+-------------------------------------------------------------------+
     | Send and return record layouts for request: Set Date              |
     +-------------------------------------------------------------------+}

    send_set_date = record
       send_mess_header : send_header_type;
       implicit_unlock  : long_boolean;
       file_id          : file_id_type;
       set_access_date  : long_boolean;
       access_date      : date_type;
       set_creation_date: long_boolean;
       creation_date    : date_type
    end;

    return_set_date = record
       return_mess_header : return_header_type
    end;


    {+-------------------------------------------------------------------+
     | Send and return record layouts for request: Set Eof               |
     +-------------------------------------------------------------------+}

    send_set_eof = record
       send_mess_header : send_header_type;
       implicit_unlock  : long_boolean;
       file_id          : file_id_type;
       use_current_ptr  : long_boolean;
       byte_offset      : integer
    end;

    return_set_eof = record
       return_mess_header : return_header_type
    end;


    {+-------------------------------------------------------------------+
     | Send and return record layouts for request: Unlock A File         |
     +-------------------------------------------------------------------+}

    send_unlockafile = record
       send_mess_header : send_header_type;
       file_id          : file_id_type;
       explicit_unlock  : long_boolean;
    end;

    return_unlockafile = record
       return_mess_header : return_header_type;
    end;


    {+-------------------------------------------------------------------+
```



```
     | Send and return record layouts for request: Volume Status         |
     +-------------------------------------------------------------------+}

    send_volume_status = record
       send_mess_header         : send_header_type;
       volume_name_header       : volume_header_type;
    end;

    return_volume_status = record
       return_mess_header   : return_header_type;
       exist                : long_boolean;
       interleave           : integer;
       volume_name          : name_type
    end;


    {+-------------------------------------------------------------------+
     | Send and return record layouts for request: Write A Record        |
     +-------------------------------------------------------------------+}

    send_writearecord = record
       send_mess_header  : send_header_type;
       implicit_unlock   : long_boolean;
       file_id           : file_id_type;
       access_code       : integer;
       filler3           : array [1..2] of gang_32bit_filler;
       requested         : integer;
       offset            : integer;
       filler8           : long_boolean;
       flush_buffer      : long_boolean;
       data              : packed array [1..512] of char
    end;

    return_writearecord = record
       return_mess_header  : return_header_type;
       actual              : integer;
    end;


  {^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^}

  $include 'PACKETS'$

  type
    ptrmp             = msg_packet_ptr;
    ptrsareyoualive   = ^send_are_you_alive;
    ptrrareyoualive   = ^return_are_you_alive;
    ptrscat           = ^send_catalogue;
    ptrrcat           = ^return_catalogue;
    ptrscatpass       = ^send_cat_passwords;
    ptrrcatpass       = ^return_cat_passwords;
    ptrschangeprotect= ^send_change_protect_codes;
    ptrrchangeprotect= ^return_change_protect_codes;
    ptrschangevolume = ^send_change_volume_label;
    ptrrchangevolume = ^return_change_volume_label;
    ptrsclose         = ^send_closeafile;
    ptrrclose         = ^return_closeafile;
    ptrscopy          = ^send_copy;
    ptrrcopy          = ^return_copy;
    ptrscreatefile    = ^send_createafile;
```

```
   839:D        1    ptrrcreatefile    = ^return_createafile;
   840:D        1    ptrscreatelink    = ^send_createlink;
   841:D        1    ptrrcreatelink    = ^return_createlink;
   842:D        1    ptrsexchange      = ^send_xchg_open;
   843:D        1    ptrrexchange      = ^return_xchg_open;
   844:D        1    ptrsfileinfo      = ^send_fileinfo;
   845:D        1    ptrrfileinfo      = ^return_fileinfo;
   846:D        1    ptrsgangclean     = ^send_gang_cleanup;
   847:D        1    ptrrgangclean     = ^return_gang_cleanup;
   848:D        1    ptrsinit          = ^send_initialize;
   849:D        1    ptrrinit          = ^return_initialize;
   850:D        1    ptrslock          = ^send_lockafile;
   851:D        1    ptrrlock          = ^return_lockafile;
   852:D        1    ptrsopen          = ^send_openafile;
   853:D        1    ptrropen          = ^return_openafile;
   854:D        1    ptrspos           = ^send_positiontoarecord;
   855:D        1    ptrrpos           = ^return_positiontoarecord;
   856:D        1    ptrspurge         = ^send_purgelink;
   857:D        1    ptrrpurge         = ^return_purgelink;
   858:D        1    ptrsread          = ^send_readarecord;
   859:D        1    ptrrread          = ^return_readarecord;
   860:D        1    ptrssetdate       = ^send_set_date;
   861:D        1    ptrrsetdate       = ^return_set_date;
   862:D        1    ptrsseteof        = ^send_set_eof;
   863:D        1    ptrrseteof        = ^return_set_eof;
   864:D        1    ptrsunlock        = ^send_unlockafile;
   865:D        1    ptrrunlock        = ^return_unlockafile;
   866:D        1    ptrsvol           = ^send_volume_status;
   867:D        1    ptrrvol           = ^return_volume_status;
   868:D        1    ptrswrite         = ^send_writearecord;
   869:D        1    ptrrwrite         = ^return_writearecord;
   870:S
   871:D        1    ptrshead          = ^send_header_type;
   872:D        1    ptrrhead          = ^return_header_type;
   873:S
   874:D        1    pk_ptr = record case integer of
   875:D        1                0 : (mp             : ptrmp);
   876:D        1                1 : (sareyoualive   : ptrsareyoualive);
   877:D        1                2 : (rareyoualive   : ptrrareyoualive);
   878:D        1                3 : (scat           : ptrscat);
   879:D        1                4 : (rcat           : ptrrcat);
   880:D        1                5 : (scatpass       : ptrscatpass);
   881:D        1                6 : (rcatpass       : ptrrcatpass);
   882:D        1                7 : (schangeprotect: ptrschangeprotect);
   883:D        1                8 : (rchangeprotect: ptrrchangeprotect);
   884:D        1                9 : (schangevolume : ptrschangevolume);
   885:D        1               10 : (rchangevolume : ptrrchangevolume);
   886:D        1               11 : (sclose        : ptrsclose);
   887:D        1               12 : (rclose        : ptrrclose);
   888:D        1               13 : (scopy         : ptrscopy);
   889:D        1               14 : (rcopy         : ptrrcopy);
   890:D        1               15 : (screatefile   : ptrscreatefile);
   891:D        1               16 : (rcreatefile   : ptrrcreatefile);
   892:D        1               17 : (screatelink   : ptrscreatelink);
   893:D        1               18 : (rcreatelink   : ptrrcreatelink);
   894:D        1               19 : (sexchange     : ptrsexchange);
   895:D        1               20 : (rexchange     : ptrrexchange);
   896:D        1               21 : (sfileinfo     : ptrsfileinfo);
   897:D        1               22 : (rfileinfo     : ptrrfileinfo);
   898:D        1               23 : (sgangclean    : ptrsgangclean);
```



```
   899:D        1               24 : (rgangclean    : ptrrgangclean);
   900:D        1               25 : (sinit         : ptrsinit);
   901:D        1               26 : (rinit         : ptrrinit);
   902:D        1               27 : (slock         : ptrslock);
   903:D        1               28 : (rlock         : ptrrlock);
   904:D        1               29 : (sopen         : ptrsopen);
   905:D        1               30 : (ropen         : ptrropen);
   906:D        1               31 : (spos          : ptrspos);
   907:D        1               32 : (rpos          : ptrrpos);
   908:D        1               33 : (spurge        : ptrspurge);
   909:D        1               34 : (rpurge        : ptrrpurge);
   910:D        1               35 : (sread         : ptrsread);
   911:D        1               36 : (rread         : ptrrread);
   912:D        1               37 : (ssetdate      : ptrssetdate);
   913:D        1               38 : (rsetdate      : ptrrsetdate);
   914:D        1               39 : (sseteof       : ptrsseteof);
   915:D        1               40 : (rseteof       : ptrrseteof);
   916:D        1               41 : (sunlock       : ptrsunlock);
   917:D        1               42 : (runlock       : ptrrunlock);
   918:D        1               43 : (svol          : ptrsvol);
   919:D        1               44 : (rvol          : ptrrvol);
   920:D        1               45 : (swrite        : ptrswrite);
   921:D        1               46 : (rwrite        : ptrrwrite);
   922:D        1               47 : (shead         : ptrshead);
   923:D        1               48 : (rhead         : ptrrhead);
   924:D        1             end;
   925:S
   926:D        1    name_set_array_three  = array [1..3] of file_name_set;
   927:D        1    name_set_array        = array [1..6] of file_name_set;
   928:D        1    pnsa                  = ^name_set_array;
   929:S
   930:D        1    protectcode_set_array = array [1..24] of protect_code_set;
   931:D        1    ppsa                  = ^protectcode_set_array;
   932:S
   933:D        1 $include 'INIT:SRM_TYPES'$
   933:D        1 $include 'INIT:SRM_ERRS'$
   934:D        1 CONST
   935:D        1       ios_error_base                      = 31000;
   936:D        1       ios_software_bug                    = 31000;
   937:D        1       ios_bad_select_code                 = 31001;
   938:D        1       ios_unallocated_extent              = 31002;
   939:D        1       ios_ds_rom_missing                  = 31003;
   940:D        1       ios_unsupported_dam                 = 31004;
   941:D        1       ios_device_drivers_dont_match       = 31005;
   942:D        1       ios_invalid_ios_request             = 31006;
   943:D        1       ios_attach_table_full               = 31007;
   944:D        1       ios_improper_mass_storage_device    = 31008;
   945:D        1       ios_directory_formats_dont_match    = 31009;
   946:D        1       ios_invalid_file_size               = 31010;
   947:D        1       ios_invalid_file_id                 = 31011;
   948:D        1       ios_volume_recoverable_error        = 31012;
   949:D        1       ios_volume_io_error                 = 31013;
   950:D        1       ios_file_pathname_missing           = 31014;
   951:D        1       ios_illegal_byte_number             = 31015;
   952:D        1       ios_corrupt_directory               = 31016;
   953:D        1       ios_successful_completion           = 31017;
   954:D        1       ios_system_down                     = 31018;
   955:D        1       ios_file_unopened                   = 31019;
   956:D        1       ios_volume_offline                  = 31020;
   957:D        1       ios_volume_labels_dont_match        = 31021;
```

```
   958:D        1      ios_password_not_allowed             = 31022;
   959:D        1      ios_access_to_file_not_allowed       = 31023;
   960:D        1      ios_unsupported_directory_operation = 31024;
   961:D        1      ios_conflicting_share_modes          = 31025;
   962:D        1      ios_bad_file_name                    = 31026;
   963:D        1      ios_file_in_use                      = 31027;
   964:D        1      ios_insufficient_disk_space          = 31028;
   965:D        1      ios_duplicate_filenames              = 31029;
   966:D        1      ios_phys_eof_encountered             = 31030;
   967:D        1      ios_no_capability_for_file           = 31031;
   968:D        1      ios_file_not_found                   = 31032;
   969:D        1      ios_volume_in_use                    = 31033;
   970:D        1      ios_file_not_directory               = 31034;
   971:D        1      ios_directory_not_empty              = 31035;
   972:D        1      ios_volume_not_found                 = 31036;
   973:D        1      ios_invalid_protect_code             = 31037;
   974:D        1      ios_volume_unrecoverable_error       = 31038;
   975:D        1      ios_password_not_found               = 31039;
   976:D        1      ios_duplicate_passwords              = 31040;
   977:D        1      ios_deadlock_detected                = 31041;
   978:D        1      ios_link_to_directory_not_allowed    = 31042;
   979:D        1      ios_rename_across_volumes            = 31043;
   980:D        1      ios_volume_down                      = 31044;
   981:D        1      ios_eof_encountered                  = 31045;
   982:D        1      ios_invalid_file_code                = 31046;
   983:D        1      ios_file_locked_please_retry         = 31047;
   984:D        1      ios_no_reply                         = 31048;
   985:D        1      ios_purge_on_open                    = 31049;
   986:S
   987:D        1      ios_error_top                        = 31049;
   988:D        1      { '--------------' errors start at 31050 }
   989:S
   990:D        1 $include 'INIT:SRM_ERRS'$
   991:S
   992:D        1 var
   993:D    -4  1   packet_ptr            : pk_ptr;
   994:D    -8  1   defaulttimeout        : integer;      {timeout values in milliseconds}
   995:D   -12  1   waitforlocktimeout    : integer;
   996:D   -16  1   copytimeout           : integer;
   997:D   -18  1   srmsavesc             : shortint;
   998:S
   999:D   -18  1 procedure srm_init;
  1000:D   -18  1 procedure resetcard(unum : unitnum);
  1001:D   -18  1 procedure packetout(unum          : unitnum);
  1002:D        1 procedure packetin(unum          : unitnum;
  1003:D   -18  2                    sendreq       : integer);
  1004:D   -18  1 procedure areyoualivepack(unum          : unitnum);
  1005:D        1 procedure catpack(unum          : unitnum;
  1006:D        2                   nfns          : integer;
  1007:D        2                   nsaptr        : pnsa;
  1008:D        2                   path          : path_start_type;
  1009:D        2                   wd            : file_id_type;
  1010:D        2                   rtpass        : name_type;
  1011:D        2                   max           : integer;
  1012:D   -18  2                   indx          : integer);
  1013:D        1 procedure catpasspack(unum      : unitnum;
  1014:D        2                       nfns      : integer;
  1015:D        2                       nsaptr    : pnsa;
  1016:D        2                       path      : path_start_type;
  1017:D        2                       wd        : file_id_type;
```



```
  1018:D        2                       rtpass    : name_type;
  1019:D        2                       max       : integer;
  1020:D   -18  2                       indx      : integer);
  1021:D        1 procedure changeprotectpack(unum       : unitnum;
  1022:D        2                             nfns       : integer;
  1023:D        2                             nsaptr     : pnsa;
  1024:D        2                             path       : path_start_type;
  1025:D        2                             wd         : file_id_type;
  1026:D        2                             rtpass     : name_type;
  1027:D        2                             nps        : integer;
  1028:D   -18  2                             psaptr     : ppsa);
  1029:D        1 procedure changevolpack(unum          : unitnum;
  1030:D        2                         vpass         : name_type;
  1031:D        2                         newname       : name_type;
  1032:D   -18  2                         newpass       : name_type);
  1033:D        1 procedure closepack(unum        : unitnum;
  1034:D   -18  2                     fid         : file_id_type);
  1035:D        1 procedure copypack(unum         : unitnum;
  1036:D        2                   srcfid       : file_id_type;
  1037:D        2                   srcoff       : integer;
  1038:D        2                   destfid      : file_id_type;
  1039:D        2                   destoff      : integer;
  1040:D   -18  2                   req          : integer);
  1041:D        1 procedure createpack(unum       : unitnum;
  1042:D        2                     nfns        : integer;
  1043:D        2                     nsaptr      : pnsa;
  1044:D        2                     path        : path_start_type;
  1045:D        2                     wd          : file_id_type;
  1046:D        2                     rtpass      : name_type;
  1047:D        2                     nps         : integer;
  1048:D        2                     psaptr      : ppsa;
  1049:D        2                     ftype       : gang_file_codes;
  1050:D        2                     mode        : integer;
  1051:D        2                     maxrec      : integer;
  1052:D        2                     ext1        : integer;
  1053:D        2                     ext2        : integer;
  1054:D   -18  2                     xaddr       : integer);
  1055:D        1 procedure createlinkpack(unum          : unitnum;
  1056:D        2                          oldnfns       : integer;
  1057:D        2                          oldnsaptr     : pnsa;
  1058:D        2                          oldpath       : path_start_type;
  1059:D        2                          oldwd         : file_id_type;
  1060:D        2                          oldrtpass     : name_type;
  1061:D        2                          newnfns       : integer;
  1062:D        2                          newnsaptr     : pnsa;
  1063:D        2                          newpath       : path_start_type;
  1064:D        2                          newwd         : file_id_type;
  1065:D        2                          newrtpass     : name_type;
  1066:D   -18  2                          purgeold      : boolean);
  1067:D        1 procedure exchangepack(unum          : unitnum;
  1068:D        2                        fid1          : file_id_type;
  1069:D   -18  2                        fid2          : file_id_type);
  1070:D        1 procedure fileinfopack(unum          : unitnum;
  1071:D   -18  2                        fid           : file_id_type);
  1072:D        1 procedure gangcleanpack(unum         : unitnum;
  1073:D   -18  2                         savewd       : boolean);
  1074:D        1 procedure lockpack(unum         : unitnum;
  1075:D        2                   fid          : file_id_type;
  1076:D   -18  2                   wait         : boolean);
  1077:D        1 procedure openpack(unum         : unitnum;
```

```
1078:D        2                    nfns          : integer;
1079:D        2                    nsaptr        : pnsa;
1080:D        2                    path          : path_start_type;
1081:D        2                    wd            : file_id_type;
1082:D        2                    rtpass        : name_type;
1083:D        2                    share         : integer;
1084:D   -18  2                    opn           : gang_open_type);
1085:D        1 procedure pospack(unum           : unitnum;
1086:D        2                   fid            : file_id_type;
1087:D        2                   typepos        : position_type;
1088:D   -18  2                   boffset        : integer);
1089:D        1 procedure purgepack(unum         : unitnum;
1090:D        2                     nfns         : integer;
1091:D        2                     nsaptr       : pnsa;
1092:D        2                     path         : path_start_type;
1093:D        2                     wd           : file_id_type;
1094:D   -18  2                     rtpass       : name_type);
1095:D        1 procedure sendreadpack(unum      : unitnum;
1096:D        2                        fid       : file_id_type;
1097:D        2                        access    : integer;
1098:D        2                        req       : integer;
1099:D        2                        off       : integer;
1100:D   -18  2                        dat       : anyptr);
1101:D        1 procedure seteofpack(unum        : unitnum;
1102:D        2                      fid         : file_id_type;
1103:D        2                      usecurptr   : boolean;
1104:D   -18  2                      boffset     : integer);
1105:D        1 procedure unlockpack(unum        : unitnum;
1106:D   -18  2                      fid         : file_id_type);
1107:D   -18  1 procedure volpack(unum  : unitnum);
1108:D        1 procedure sendwritepack(unum     : unitnum;
1109:D        2                         fid      : file_id_type;
1110:D        2                         access   : integer;
1111:D        2                         req      : integer;
1112:D        2                         off      : integer;
1113:D   -18  2                         dat      : anyptr);
1114:D   -18  1 procedure setdefaulttimeout(time         : integer);
1115:D   -18  1 procedure setcopytimeout(time            : integer);
1116:D   -18  1 procedure setwaitforlocktimeout(time     : integer);
1117:S
1118:D   -18  1 implement
1119:S
1120:D   -18  1 const
1121:D   -18  1   packetouttimeout      = 1000;         {timeout values in milliseconds}
1122:D   -18  1   ayatimeout            = 1000;
1123:S
1124:D   -18  1 var
1125:D   -22  1   dumbuf        : ^buf_info_type;
1126:D   -23  1   srm_inited    : boolean;
1127:D   -24  1   waitingforlock: boolean;
1128:S
1129:D   -24  1 (*****************************************************************************)
1130:S
1131:D        1 procedure maptoioresult(status : integer);
1132:C        2 begin
1133:C        2   if ioresult = ord(inoerror) then
1134:C        3     if status <> 0 then
1135:C        4       case status of
1136:C        5         ios_bad_select_code                : ioresult := ord(ibadunit);
1137:C        5         ios_attach_table_full              : ioresult := ord(itoomanyopen);
```



```
1138:C        5         ios_invalid_file_size              : ioresult := ord(inotvalidsize);
1139:C        5         ios_invalid_file_id                : ioresult := ord(ilostfile);
1140:S
1141:C        5         ios_bad_file_name,
1142:C        5         ios_file_pathname_missing          : ioresult := ord(ibadtitle);
1143:S
1144:C        5         ios_illegal_byte_number            : ioresult := ord(ibadvalue);
1145:S
1146:C        5         ios_successful_completion,
1147:C        5         ios_no_reply                       : ioresult := ord(inoerror);
1148:S
1149:C        5         ios_system_down,
1150:C        5         ios_volume_offline,
1151:C        5         ios_volume_not_found,
1152:C        5         ios_volume_down                    : ioresult := ord(znodevice);
1153:S
1154:C        5         ios_file_unopened                  : ioresult := ord(inotopen);
1155:S
1156:C        5         ios_password_not_allowed,
1157:C        5         ios_no_capability_for_file,
1158:C        5         ios_invalid_protect_code,
1159:C        5         ios_password_not_found,
1160:C        5         ios_duplicate_passwords            : ioresult := ord(ibadpass);
1161:S
1162:C        5         ios_access_to_file_not_allowed     : ioresult := ord(inoaccess);
1163:S
1164:C        5         ios_unsupported_directory_operation,
1165:C        5         ios_link_to_directory_not_allowed  : ioresult := ord(inotondir);
1166:S
1167:C        5         ios_deadlock_detected,
1168:C        5         ios_conflicting_share_modes,
1169:C        5         ios_file_locked_please_retry       : ioresult := ord(ifilelocked);
1170:S
1171:C        5         ios_file_in_use,
1172:C        5         ios_purge_on_open                  : ioresult := ord(inotclosed);
1173:S
1174:C        5         ios_insufficient_disk_space        : ioresult := ord(inoroom);
1175:C        5         ios_duplicate_filenames            : ioresult := ord(idupfile);
1176:S
1177:C        5         ios_phys_eof_encountered,
1178:C        5         ios_eof_encountered                : ioresult := ord(ieof);
1179:S
1180:C        5         ios_file_not_found                 : ioresult := ord(inofile);
1181:C        5         ios_volume_in_use                  : ioresult := ord(znotready);
1182:C        5         ios_file_not_directory             : ioresult := ord(ifilenotdir);
1183:C        5         ios_directory_not_empty            : ioresult := ord(idirnotempty);
1184:C        5         ios_invalid_file_code              : ioresult := ord(ibadfiletype);
1185:S
1186:C        5         otherwise                            ioresult := ord(isrmcatchall);
1187:C        5       end;
1188:C        2 end;
1189:D   -24  1 (*****************************************************************************)
1190:S
1191:D        1 procedure initdumbuf;
1192:C        2 begin
1193:C        2   with dumbuf^ do
1194:C        3     begin
1195:C        3       buf_ptr           := nil;
1196:C        3       act_tfr           := no_tfr;
1197:C        3       active_isc        := no_isc;
```

```
1198:C        3       buf_size          := 0;
1199:C        3       buf_empty         := nil;
1200:C        3       buf_fill          := nil;
1201:C        3       drv_tmp_ptr       := nil;
1202:C        3       eot_proc.dummy_sl := nil;
1203:C        3       eot_proc.dummy_pr := nil;
1204:C        3       eot_parm          := nil;
1205:C        3       dma_priority      := false;
1206:C        3     end;
1207:C        2 end;
1208:D   -24  1 (*****************************************************************************)
1209:S
1210:D   -24  1 (*****************************************************************************)
1211:S
1212:D        1 procedure srm_init;
1213:C        2 begin
1214:C        2   if not srm_inited then
1215:C        3     begin
1216:C        3       new(packet_ptr.mp);
1217:C        3       new(dumbuf);
1218:C        3       markuser;
1219:C        3       srm_inited := true;
1220:C        3     end;
1221:C        2   initdumbuf;
1222:C        2   defaulttimeout        := 30000;       {timeout values in milliseconds}
1223:C        2   waitforlocktimeout    := 0;
1224:C        2   copytimeout           := 120000;
1225:C        2 end;
1226:D   -24  1 (*****************************************************************************)
1227:S
1228:D        1 procedure resetcard(unum : unitnum);
1229:C        2 begin
1230:C        2   with isc_table[unitable^[unum].sc] do
1231:C        3     call(io_drv_ptr^.iod_init, io_tmp_ptr);
1232:C        2 end;
1233:D   -24  1 (*****************************************************************************)
1234:S
1235:D        1 procedure setdefaulttimeout(time: integer);
1236:C        2 begin
1237:C        2   defaulttimeout := time;                          {time is in milliseconds}
1238:C        2 end;
1239:D   -24  1 (*****************************************************************************)
1240:S
1241:D        1 procedure setwaitforlocktimeout(time: integer);
1242:C        2 begin
1243:C        2   waitforlocktimeout := time;                      {time is in milliseconds}
1244:C        2 end;
1245:D   -24  1 (*****************************************************************************)
1246:S
1247:D        1 procedure setcopytimeout(time: integer);
1248:C        2 begin
1249:C        2   copytimeout := time;                             {time is in milliseconds}
1250:C        2 end;
1251:D   -24  1 (*****************************************************************************)
1252:S
1253:D        1 procedure setintegertimeout(sc   : type_isc;
1254:D        2                             time: integer);
1255:C        2 begin
1256:C        2   with isc_table[sc] do
1257:C        3     begin
```



```
1258:C        3       user_time := time;                         {time is in milliseconds}
1259:C        3       if io_tmp_ptr <> nil then
1260:C        4         io_tmp_ptr^.timeout := time;
1261:C        3     end;
1262:C        2 end;
1263:D   -24  1 (*****************************************************************************)
1264:S
1265:D        1 function do_buffer_data(var b_info : buf_info_type) : integer;
1266:C        2 begin
1267:C        2   with b_info do
1268:C        3     do_buffer_data := integer(buf_fill) - integer(buf_empty);
1269:C        2 end;
1270:D   -24  1 (*****************************************************************************)
1271:S
1272:D        1 procedure do_buffer_reset(var b_info : buf_info_type);
1273:C        2 begin
1274:C        2   with b_info do
1275:C        3     if active_isc <> no_isc then
1276:C        4       io_escape(ioe_buf_busy,no_isc)
1277:C        4     else
1278:C        4       begin
1279:C        4         buf_fill        := buf_ptr;
1280:C        4         buf_empty       := buf_ptr;
1281:C        4       end;
1282:C        2 end;
1283:D   -24  1 (*****************************************************************************)
1284:S
1285:D        1 function do_buffer_space(var b_info : buf_info_type) : integer;
1286:C        2 begin
1287:C        2   with b_info do
1288:C        3     begin
1289:C        3       if  (do_buffer_data(b_info) = 0)
1290:C        4       and (active_isc = no_isc) then
1291:C        4         do_buffer_reset(b_info);
1292:C        3       do_buffer_space := buf_size + integer(buf_ptr) - integer(buf_fill);
1293:C        3     end;
1294:C        2 end;
1295:D   -24  1 (*****************************************************************************)
1296:S
1297:D        1 procedure do_transfer(sc        : type_isc;
1298:D        2                       dir       : dir_of_tfr;
1299:D        2                       count     : integer;
1300:D        2                       tfr_end   : boolean);
1301:D        2 var
1302:D    -4  2   lcount        : integer;
1303:C        2 begin
1304:C        2   with isc_table[sc], dumbuf^ do
1305:C        3     begin
1306:C        3       if not tfr_end then
1307:C        4         lcount := count
1308:C        4       else
1309:C        4         if dir = from_memory then
1310:C        5           lcount := do_buffer_data(dumbuf^)
1311:C        5         else
1312:C        5           lcount := do_buffer_space(dumbuf^);
1313:C        3       if io_tmp_ptr = nil then
1314:C        4         io_escape(ioe_no_driver,sc);
1315:C        3       if lcount = 0 then
1316:C        4         io_escape(ioe_bad_cnt,no_isc);
```

```
1318:C        3       if active_isc <> no_isc then
1319:C        4         io_escape(ioe_buf_busy,no_isc);
1320:S
1321:C        3       if do_buffer_data(dumbuf^) = 0 then
1322:C        4         do_buffer_reset(dumbuf^);
1323:S
1324:C        3       with io_tmp_ptr^ do
1325:C        4         if dir = to_memory then
1326:C        5           begin
1327:C        5             if in_bufptr <> nil then
1328:C        6               io_escape(ioe_isc_busy,sc);
1329:C        5             if do_buffer_space(dumbuf^) < lcount then
1330:C        6               io_escape(ioe_no_space,sc);
1331:C        5             in_bufptr := dumbuf;
1332:C        5           end
1333:C        5         else
1334:C        5           begin
1335:C        5             if out_bufptr <> nil then
1336:C        6               io_escape(ioe_isc_busy,sc);
1337:C        5             if do_buffer_data(dumbuf^) < lcount then
1338:C        6               io_escape(ioe_no_data,sc);
1339:C        5             out_bufptr := dumbuf;
1340:C        5           end;
1341:S
1342:C        3       drv_tmp_ptr       := io_tmp_ptr;
1343:C        3       act_tfr           := no_tfr;
1344:C        3       usr_tfr           := serial_fastest;
1345:C        3       b_w_mode          := false;              {byte mode}
1346:C        3       direction         := dir;
1347:C        3       term_char         := -1;                 {no term char}
1348:C        3       term_count        := lcount;
1349:C        3       end_mode          := tfr_end;
1350:S
1351:C        3       call(io_drv_ptr^.iod_tfr,io_tmp_ptr,dumbuf);
1352:C        3     end;
1353:C        2 end;
1354:D   -24  1 (*****************************************************************************)
1355:S
1356:D        1 procedure dorecover(unum        : unitnum);
1357:C        2 begin
1358:C        2   ioresult := ord(isrmcatchall);
1359:C        2   if (escapecode <> ioescapecode) then
1360:C        3     begin
1361:C        3       if srmsavesc = 0 then
1362:C        4         srmsavesc     := escapecode
1363:C        4     end
1364:C        3   else
1365:C        3     if (ioe_result = ioe_timeout) then
1366:C        4       ioresult := ord(ztimeout);
1367:C        2   resetcard(unum);
1368:C        2 end;
1369:D   -24  1 (*****************************************************************************)
1370:S
1371:D        1 procedure packetout(unum        : unitnum);
1372:D        2 var
1373:D    -4  2   ip    : ^integer;
1374:C        2 begin
1375:C        2   with unitable^[unum] do
1376:C        3     try
1377:C        4       initdumbuf;
```



```
1378:C        4       setintegertimeout(sc,packetouttimeout);
1379:C        4       packet_ptr.mp^[8]  := chr(unitable^[unum].ba);
1380:C        4       packet_ptr.mp^[9]  := chr(0);
1381:C        4       packet_ptr.mp^[10] := chr(0);
1382:C        4       packet_ptr.mp^[11] := chr(0);
1383:C        4       ip := addr(packet_ptr.mp^[17]);  {request type}
1384:C        4       if ip^ = req_are_you_alive then
1385:C        5         packet_ptr.mp^[12] := chr(2)
1386:C        5       else
1387:C        5         packet_ptr.mp^[12] := chr(7);
1388:C        4       dumbuf^.buf_ptr   := anyptr(packet_ptr.mp);
1389:C        4       dumbuf^.buf_size  := sizeof(msg_packet_type);
1390:C        4       dumbuf^.buf_empty := addr(packet_ptr.mp^[8]);
1391:C        4       ip := addr(packet_ptr.mp^[13]);  {packet length}
1392:C        4       dumbuf^.buf_fill  := addr(packet_ptr.mp^[12+1+ip^]);
1393:C        4       do_transfer(sc,from_memory,0,true);
1394:C        4     recover
1395:C        4       dorecover(unum);
1396:C        2 end;
1397:D   -24  1 (*****************************************************************************)
1398:S
1399:D        1 procedure packetin(unum         : unitnum;
1400:D        2                    sendreq      : integer);
1401:D        2 type
1402:D        2   ayastatustype = packed record
1403:D        2                     srmnode     : byte;
1404:D        2                     linkerrs    : byte;
1405:D        2                     computerid  : shortint;
1406:D        2                   end;
1407:D        2   ayastatusptr  = ^ayastatustype;
1408:D        2 var
1409:D    -4  2   count : integer;
1410:C        2 begin
1411:C        2   with unitable^[unum] do
1412:C        3     try
1413:C        4       initdumbuf;
1414:C        4       if sendreq = req_are_you_alive then
1415:C        5         setintegertimeout(sc,ayatimeout)
1416:C        5       else if (sendreq = req_flock) and (waitingforlock) then
1417:C        6         setintegertimeout(sc,waitforlocktimeout)
1418:C        6       else if (sendreq = req_copy) or
1419:C        7               (sendreq = req_create) then        { 3.0 BUG FIX 3/16/84 }
1420:C        7         setintegertimeout(sc,copytimeout)
1421:C        7       else
1422:C        7         setintegertimeout(sc,defaulttimeout);
1423:C        4       with packet_ptr.rhead^, packet_ptr.rread^ do
1424:C        5         begin
1425:C        5           repeat
1426:C        6             fillchar(linkfiller, 28, chr(0));
1427:S
1428:C        6             dumbuf^.buf_ptr   := anyptr(packet_ptr.mp);
1429:C        6             dumbuf^.buf_size  := sizeof(msg_packet_type);
1430:C        6             dumbuf^.buf_empty := anyptr(packet_ptr.mp);
1431:C        6             dumbuf^.buf_fill  := addr(packet_ptr.mp^[9]);
1432:S
1433:C        6             if sendreq <> req_read then
1434:C        7               begin
1435:C        7                 do_transfer(sc,to_memory,0,true);
1436:C        7               end
1437:C        7             else
```

```
1438:C        7                begin
1439:C        7                  count := size_from_gang_error + 4;
1440:C        7                  do_transfer(sc,to_memory,count,false);
1441:S
1442:C        7                  if message_length > size_from_gang_error then
1443:C        8                    if  return_request_type <> -req_read then
1444:C        9                      begin
1445:C        9                        do_transfer(sc,to_memory,0,true);
1446:C        9                      end
1447:C        9                    else
1448:C        9                      begin
1449:C        9                        count := size_from_read - size_from_gang_error;
1450:C        9                        do_transfer(sc,to_memory,count,false);
1451:S
1452:C        9                        count := actual;
1453:S
1454:C        9                        if count > 0 then
1455:C       10                          begin
1456:C       10                            dumbuf^.buf_ptr   := anyptr(user_sequencing_field);
1457:C       10                            dumbuf^.buf_size  := 512;
1458:C       10                            dumbuf^.buf_empty := anyptr(user_sequencing_field);
1459:C       10                            dumbuf^.buf_fill  := anyptr(user_sequencing_field);
1460:C       10                            do_transfer(sc,to_memory,count,false);
1461:C       10                          end;
1462:C        9                      end;
1463:C        7                end;
1464:C        6            until return_request_type = -sendreq;
1465:S
1466:C        5            if sendreq <> req_are_you_alive then
1467:C        6              maptoioresult(status)
1468:C        6            else
1469:C        6              if ayastatusptr(addr(status))^.srmnode <> 1 then
1470:C        7                ioresult := ord(znodevice);
1471:C        5          end;
1472:C        4      recover
1473:C        4        dorecover(unum);
1474:C        2  end;
1475:D   -24   1  (*****************************************************************************)
1476:S
1477:D        1  procedure setup_smh(var smh      : send_header_type;
1478:D        2                         ml,
1479:D        2                         srt,
1480:D        2                         usf       : integer);
1481:C        2  begin
1482:C        2    with smh do
1483:C        3      begin
1484:C        3        message_length    := ml;
1485:C        3        send_request_type := srt;
1486:C        3        user_sequencing_field := usf;
1487:C        3      end;
1488:C        2  end;
1489:D   -24   1  (*****************************************************************************)
1490:S
1491:D        1  procedure setup_vnh(var vnh      : volume_header_type;
1492:D        2                         unum      : unitnum);
1493:C        2  begin
1494:C        2    with vnh do
1495:C        3      begin
1496:C        3        filler1                   := 0;
1497:C        3        driver_name               := ' ';
```



```
1498:C        3        catalogue_organization    := ' ';
1499:C        3        device_address_present.i  := 1;
1500:C        3        with device_address do
1501:C        4          begin
1502:C        4            address1                := unitable^[unum].du;  {unit number}
1503:C        4            haddress                := 0;
1504:C        4            unit_num                := 0;
1505:C        4            volume_num              := 0;
1506:C        4          end;
1507:C        3        volume_name := ' ';
1508:C        3      end;
1509:C        2  end;
1510:D   -24   1  (*****************************************************************************)
1511:S
1512:D        1  procedure setup_fnh(var fnh      : file_header_type;
1513:D        2                         num       : integer;
1514:D        2                         wd        : file_id_type;
1515:D        2                         pt        : path_start_type;
1516:D   -16  2                         rp        : name_type);
1517:C        2  begin
1518:C        2    with fnh do
1519:C        3      begin
1520:C        3        num_file_name_sets := num;
1521:C        3        working_directory  := wd;
1522:C        3        filler1            := 0;
1523:C        3        path_type          := pt;
1524:C        3        root_password      := rp;
1525:C        3      end;
1526:C        2  end;
1527:D   -24   1  (*****************************************************************************)
1528:S
1529:D        1  procedure areyoualivepack(unum          : unitnum);
1530:C        2  begin
1531:C        2    with packet_ptr.sareyoualive^ do
1532:C        3      begin
1533:C        3        setup_smh(send_mess_header,
1534:C        3                  size_to_are_you_alive,
1535:C        3                  req_are_you_alive,
1536:C        3                  0);
1537:C        3      end;
1538:S
1539:C        2    packetout(unum);
1540:C        2    if ioresult = ord(inoerror) then
1541:C        3      packetin(unum,req_are_you_alive);
1542:C        2    if ioresult <> ord(inoerror) then
1543:C        3      packet_ptr.rareyoualive^.return_mess_header.status := 0;
1544:C        2  end;
1545:D   -24   1  (*****************************************************************************)
1546:S
1547:D        1  procedure catpack(unum          : unitnum;
1548:D        2                       nfns          : integer;
1549:D        2                       nsaptr        : pnsa;
1550:D        2                       path          : path_start_type;
1551:D        2                       wd            : file_id_type;
1552:D        2                       rtpass        : name_type;
1553:D        2                       max           : integer;
1554:D   -16  2                       indx          : integer);
1555:C        2  begin
1556:C        2    with packet_ptr.scat^ do
1557:C        3      begin
```

```
1558:C        3       pnsa(addr(start_name_sets))^ := nsaptr^;
1559:S
1560:C        3       setup_smh(send_mess_header,
1561:C        3                 size_to_cat+nfns*36,
1562:C        3                 req_catalog,
1563:C        3                 0);
1564:S
1565:C        3       max_num_files := max;
1566:C        3       file_index    := indx;
1567:C        3       filler1       := 0;
1568:S
1569:C        3       setup_vnh(volume_name_header,unum);
1570:S
1571:C        3       filler2 := 0;
1572:S
1573:C        3       setup_fnh(file_name_header,
1574:C        3                 nfns,
1575:C        3                 wd,      {working directory}
1576:C        3                 path,
1577:C        3                 rtpass);{root password}
1578:S
1579:C        3     end;
1580:C        2   packetout(unum);
1581:C        2   packetin(unum,req_catalog);
1582:C        2 end;
1583:D   -24   1 (*****************************************************************************)
1584:S
1585:D        1 procedure catpasspack(unum      : unitnum;
1586:D        2                       nfns      : integer;
1587:D        2                       nsaptr    : pnsa;
1588:D        2                       path      : path_start_type;
1589:D        2                       wd        : file_id_type;
1590:D        2                       rtpass    : name_type;
1591:D        2                       max       : integer;
1592:D   -16  2                       indx      : integer);
1593:C        2 begin
1594:C        2   with packet_ptr.scatpass^ do
1595:C        3     begin
1596:C        3       pnsa(addr(start_name_sets))^ := nsaptr^;
1597:S
1598:C        3       setup_smh(send_mess_header,
1599:C        3                 size_to_catprotect+nfns*36,
1600:C        3                 req_catprotect,
1601:C        3                 0);
1602:S
1603:C        3       max_num_passwords := max;
1604:C        3       filler1           := 0;
1605:C        3       password_index    := indx;
1606:S
1607:C        3       setup_vnh(volume_name_header,unum);
1608:S
1609:C        3       filler2 := 0;
1610:S
1611:C        3       setup_fnh(file_name_header,
1612:C        3                 nfns,
1613:C        3                 wd,      {working directory}
1614:C        3                 path,
1615:C        3                 rtpass);{root password}
1616:S
1617:C        3     end;
```



```
1618:C        2   packetout(unum);
1619:C        2   packetin(unum,req_catprotect);
1620:C        2 end;
1621:D   -24   1 (*****************************************************************************)
1622:S
1623:D        1 procedure changeprotectpack(unum       : unitnum;
1624:D        2                             nfns       : integer;
1625:D        2                             nsaptr     : pnsa;
1626:D        2                             path       : path_start_type;
1627:D        2                             wd         : file_id_type;
1628:D        2                             rtpass     : name_type;
1629:D        2                             nps        : integer;
1630:D   -16  2                             psaptr     : ppsa);
1631:C        2 begin
1632:C        2   with packet_ptr.schangeprotect^ do
1633:C        3     begin
1634:C        3       pnsa(addr(start_name_sets))^ := nsaptr^;
1635:C        3       ppsa(addr(pnsa(addr(start_name_sets))^[nfns+1]))^ := psaptr^;
1636:S
1637:C        3       setup_smh(send_mess_header,
1638:C        3                 size_to_changeprotect+nfns*36+nps*24,
1639:C        3                 req_changeprotect,
1640:C        3                 0);
1641:S
1642:C        3       setup_vnh(volume_name_header,unum);
1643:S
1644:C        3       setup_fnh(file_name_header,
1645:C        3                 nfns,
1646:C        3                 wd,      {working directory}
1647:C        3                 path,
1648:C        3                 rtpass);{root password}
1649:S
1650:C        3       num_protect_code_sets     := nps;
1651:C        3     end;
1652:S
1653:C        2   packetout(unum);
1654:C        2   packetin(unum,req_changeprotect);
1655:C        2 end;
1656:D   -24   1 (*****************************************************************************)
1657:S
1658:D        1 procedure changevolpack(unum            : unitnum;
1659:D        2                         vpass           : name_type;
1660:D        2                         newname         : name_type;
1661:D   -48  2                         newpass         : name_type);
1662:C        2 begin
1663:C        2   with packet_ptr.schangevolume^ do
1664:C        3     begin
1665:C        3       setup_smh(send_mess_header,
1666:C        3                 size_to_change_vol_label,
1667:C        3                 req_label,
1668:C        3                 0);
1669:S
1670:C        3       setup_vnh(volume_name_header,unum);
1671:C        3       password              := vpass;
1672:C        3       new_volume_name       := newname;
1673:C        3       new_vol_password      := newpass;
1674:C        3     end;
1675:S
1676:C        2   packetout(unum);
1677:C        2   packetin(unum,req_label);
```

```
1678:C       2 end;
1679:D   -24 1 (*****************************************************************************)
1680:S
1681:D       1 procedure closepack(unum        : unitnum;
1682:D       2                     fid         : file_id_type);
1683:C       2 begin
1684:C       2   with packet_ptr.sclose^ do
1685:C       3     begin
1686:C       3       setup_smh(send_mess_header,
1687:C       3                 size_to_close,
1688:C       3                 req_close,
1689:C       3                 0);
1690:C       3       file_id          := fid;
1691:C       3       directory_password:= ' ';
1692:C       3       file_password    := ' ';
1693:C       3       filler5.i        := 0;
1694:C       3       nodeallocate.i   := 0;
1695:C       3     end;
1696:S
1697:C       2   packetout(unum);
1698:C       2   packetin(unum,req_close);
1699:C       2 end;
1700:D   -24 1 (*****************************************************************************)
1701:S
1702:D       1 procedure copypack(unum         : unitnum;
1703:D       2                    srcfid       : file_id_type;
1704:D       2                    srcoff       : integer;
1705:D       2                    destfid      : file_id_type;
1706:D       2                    destoff      : integer;
1707:D       2                    req          : integer);
1708:C       2 begin
1709:C       2   with packet_ptr.scopy^ do
1710:C       3     begin
1711:C       3       setup_smh(send_mess_header,
1712:C       3                 size_to_copy,
1713:C       3                 req_copy,
1714:C       3                 0);
1715:S
1716:C       3       source_file_id           := srcfid;
1717:C       3       source_offset            := srcoff;
1718:C       3       destination_file_id      := destfid;
1719:C       3       destination_offset       := destoff;
1720:C       3       requested                := req;
1721:C       3     end;
1722:S
1723:C       2   packetout(unum);
1724:C       2   packetin(unum,req_copy);
1725:C       2 end;
1726:D   -24 1 (*****************************************************************************)
1727:S
1728:D       1 procedure createpack(unum       : unitnum;
1729:D       2                      nfns       : integer;
1730:D       2                      nsaptr     : pnsa;
1731:D       2                      path       : path_start_type;
1732:D       2                      wd         : file_id_type;
1733:D       2                      rtpass     : name_type;
1734:D       2                      nps        : integer;
1735:D       2                      psaptr     : ppsa;
1736:D       2                      ftype      : gang_file_codes;
1737:D       2                      mode       : integer;
```



```
1738:D       2                      maxrec     : integer;
1739:D       2                      ext1       : integer;
1740:D       2                      ext2       : integer;
1741:D   -16 2                      xaddr      : integer);
1742:C       2 begin
1743:C       2   with packet_ptr.screatefile^ do
1744:C       3     begin
1745:C       3       pnsa(addr(start_name_sets))^ := nsaptr^;
1746:C       3       if nps > 0 then
1747:C       4         ppsa(addr(pnsa(addr(start_name_sets))^[nfns+1]))^ := psaptr^;
1748:S
1749:C       3       setup_smh(send_mess_header,
1750:C       3                 size_to_create+nfns*36+nps*24,
1751:C       3                 req_create,
1752:C       3                 0);
1753:S
1754:C       3       setup_vnh(volume_name_header,unum);
1755:S
1756:C       3       setup_fnh(file_name_header,
1757:C       3                 nfns,
1758:C       3                 wd,      {working directory}
1759:C       3                 path,
1760:C       3                 rtpass);{root password}
1761:S
1762:C       3       file_code                 := ftype;
1763:C       3       record_mode               := mode;
1764:S
1765:C       3       max_record_size           := maxrec;
1766:C       3       first_extent              := ext1;
1767:C       3       contiguous_first_extent.i := 0;          {false}
1768:C       3       secondary_extent          := ext2;
1769:C       3       max_file_size             := maxint;
1770:C       3       boot_start_address        := xaddr;
1771:C       3       num_protect_code_sets     := nps;
1772:C       3       label_included_flag.i     := 0;          {false}
1773:C       3     end;
1774:S
1775:C       2   packetout(unum);
1776:C       2   packetin(unum,req_create);
1777:C       2 end;
1778:D   -24 1 (*****************************************************************************)
1779:S
1780:D       1 procedure createlinkpack(unum           : unitnum;
1781:D       2                          oldnfns        : integer;
1782:D       2                          oldnsaptr      : pnsa;
1783:D       2                          oldpath        : path_start_type;
1784:D       2                          oldwd          : file_id_type;
1785:D       2                          oldrtpass      : name_type;
1786:D       2                          newnfns        : integer;
1787:D       2                          newnsaptr      : pnsa;
1788:D       2                          newpath        : path_start_type;
1789:D       2                          newwd          : file_id_type;
1790:D       2                          newrtpass      : name_type;
1791:D   -32 2                          purgeold       : boolean);
1792:C       2 begin
1793:C       2   with packet_ptr.screatelink^ do
1794:C       3     begin
1795:C       3       pnsa(addr(start_name_sets))^ := oldnsaptr^;
1796:C       3       pnsa(addr(pnsa(addr(start_name_sets))^[oldnfns+1]))^ := newnsaptr^;
1797:S
```

```
1798:C        3       setup_smh(send_mess_header,
1799:C        3                 size_to_createlink+oldnfns*36+newnfns*36,
1800:C        3                 req_createlink,
1801:C        3                 0);
1802:S
1803:C        3       setup_vnh(volume_name_header,unum);
1804:S
1805:C        3       setup_fnh(old_file_name_header,
1806:C        3                 oldnfns,
1807:C        3                 oldwd,      {working directory}
1808:C        3                 oldpath,
1809:C        3                 oldrtpass);{root password}
1810:S
1811:C        3       setup_fnh(new_file_name_header,
1812:C        3                 newnfns,
1813:C        3                 newwd,      {working directory}
1814:C        3                 newpath,
1815:C        3                 newrtpass);{root password}
1816:S
1817:C        3       purge_old_link.i  := ord(purgeold);
1818:C        3     end;
1819:S
1820:C        2   packetout(unum);
1821:C        2   packetin(unum,req_createlink);
1822:C        2 end;
1823:D   -24  1 (*****************************************************************************)
1824:S
1825:D        1 procedure exchangepack(unum             : unitnum;
1826:D        2                        fid1             : file_id_type;
1827:D        2                        fid2             : file_id_type);
1828:C        2 begin
1829:C        2   with packet_ptr.sexchange^ do
1830:C        3     begin
1831:C        3       setup_smh(send_mess_header,
1832:C        3                 size_to_xchg_open,
1833:C        3                 req_xchg_open,
1834:C        3                 0);
1835:S
1836:C        3       file_id_1 := fid1;
1837:C        3       file_id_2 := fid2;
1838:C        3     end;
1839:S
1840:C        2   packetout(unum);
1841:C        2   packetin(unum,req_xchg_open);
1842:C        2 end;
1843:D   -24  1 (*****************************************************************************)
1844:S
1845:D        1 procedure fileinfopack(unum             : unitnum;
1846:D        2                        fid              : file_id_type);
1847:C        2 begin
1848:C        2   with packet_ptr.sfileinfo^ do
1849:C        3     begin
1850:C        3       setup_smh(send_mess_header,
1851:C        3                 size_to_info,
1852:C        3                 req_info,
1853:C        3                 0);
1854:S
1855:C        3       implicit_unlock.i := 1;
1856:C        3       file_id := fid;
1857:C        3     end;
```



```
1858:S
1859:C        2   packetout(unum);
1860:C        2   packetin(unum,req_info);
1861:C        2 end;
1862:D   -24  1 (*****************************************************************************)
1863:S
1864:D        1 procedure gangcleanpack(unum            : unitnum;
1865:D        2                         savewd          : boolean);
1866:C        2 begin
1867:C        2   with packet_ptr.sgangclean^ do
1868:C        3     begin
1869:C        3       setup_smh(send_mess_header,
1870:C        3                 size_to_gang_cleanup,
1871:C        3                 req_gang_cleanup,
1872:C        3                 0);
1873:S
1874:C        3       keep_protected_directories.i := ord(savewd);
1875:C        3     end;
1876:S
1877:C        2   packetout(unum);
1878:C        2 end;
1879:D   -24  1 (*****************************************************************************)
1880:S
1881:D        1 procedure lockpack(unum         : unitnum;
1882:D        2                    fid          : file_id_type;
1883:D        2                    wait         : boolean);
1884:C        2 begin
1885:C        2   waitingforlock := wait;
1886:C        2   with packet_ptr.slock^ do
1887:C        3     begin
1888:C        3       setup_smh(send_mess_header,
1889:C        3                 size_to_flock,
1890:C        3                 req_flock,
1891:C        3                 0);
1892:S
1893:C        3       file_id          := fid;
1894:C        3       wait_for_lock.i  := ord(wait);
1895:C        3     end;
1896:S
1897:C        2   packetout(unum);
1898:C        2   packetin(unum,req_flock);
1899:C        2 end;
1900:D   -24  1 (*****************************************************************************)
1901:S
1902:D        1 procedure openpack(unum         : unitnum;
1903:D        2                    nfns         : integer;
1904:D        2                    nsaptr       : pnsa;
1905:D        2                    path         : path_start_type;
1906:D        2                    wd           : file_id_type;
1907:D        2                    rtpass       : name_type;
1908:D        2                    share        : integer;
1909:D   -16  2                    ppn          : gang_open_type);
1910:C        2 begin
1911:C        2   with packet_ptr.sopen^ do
1912:C        3     begin
1913:C        3       pnsa(addr(start_name_sets))^ := nsaptr^;
1914:S
1915:C        3       setup_smh(send_mess_header,
1916:C        3                 size_to_open+nfns*36,
1917:C        3                 req_open,
```

```
1918:C        3                0);
1919:S
1920:C        3       setup_vnh(volume_name_header,unum);
1921:S
1922:C        3       setup_fnh(file_name_header,
1923:C        3                 nfns,
1924:C        3                 wd,      {working directory}
1925:C        3                 path,
1926:C        3                 rtpass);{root password}
1927:S
1928:C        3       filler2           := 0;
1929:C        3       filler3           := 0;
1930:C        3       share_code        := share;
1931:C        3       filler4.id        := 0;
1932:C        3       filler1           := 0;
1933:C        3       open_type         := opn;
1934:C        3     end;
1935:S
1936:C        2   packetout(unum);
1937:C        2   packetin(unum,req_open);
1938:C        2 end;
1939:D   -24  1 (*****************************************************************************)
1940:S
1941:D        1 procedure pospack(unum          : unitnum;
1942:D        2                  fid           : file_id_type;
1943:D        2                  typepos       : position_type;
1944:D        2                  boffset       : integer);
1945:C        2 begin
1946:C        2   with packet_ptr.spos^ do
1947:C        3     begin
1948:C        3       setup_smh(send_mess_header,
1949:C        3                 size_to_position,
1950:C        3                 req_position,
1951:C        3                 0);
1952:S
1953:C        3       implicit_unlock.i := 1;
1954:C        3       file_id := fid;
1955:C        3       filler3     := 0;
1956:C        3       type_of_position := typepos;
1957:C        3       byte_offset := boffset;
1958:C        3     end;
1959:S
1960:C        2   packetout(unum);
1961:C        2   packetin(unum,req_position);
1962:C        2 end;
1963:D   -24  1 (*****************************************************************************)
1964:S
1965:D        1 procedure purgepack(unum        : unitnum;
1966:D        2                     nfns        : integer;
1967:D        2                     nsaptr      : pnsa;
1968:D        2                     path        : path_start_type;
1969:D        2                     wd          : file_id_type;
1970:D   -16  2                     rtpass      : name_type);
1971:C        2 begin
1972:C        2   with packet_ptr.spurge^ do
1973:C        3     begin
1974:C        3       pnsa(addr(start_name_sets))^ := nsaptr^;
1975:S
1976:C        3       setup_smh(send_mess_header,
1977:C        3                 size_to_purge+nfns*36,
```



```
1978:C        3                 req_purgelink,
1979:C        3                 0);
1980:S
1981:C        3       setup_vnh(volume_name_header,unum);
1982:S
1983:C        3       setup_fnh(file_name_header,
1984:C        3                 nfns,
1985:C        3                 wd,      {working directory}
1986:C        3                 path,
1987:C        3                 rtpass);{root password}
1988:C        3     end;
1989:S
1990:C        2   packetout(unum);
1991:C        2   packetin(unum,req_purgelink);
1992:C        2 end;
1993:D   -24  1 (*****************************************************************************)
1994:S
1995:D        1 procedure sendreadpack(unum     : unitnum;
1996:D        2                        fid      : file_id_type;
1997:D        2                        access   : integer;
1998:D        2                        req      : integer;
1999:D        2                        off      : integer;
2000:D        2                        dat      : anyptr);
2001:C        2 begin
2002:C        2   with packet_ptr.sread^ do
2003:C        3     begin
2004:C        3       setup_smh(send_mess_header,
2005:C        3                 size_to_read,
2006:C        3                 req_read,
2007:C        3                 integer(dat));
2008:S
2009:C        3       implicit_unlock.i := 1;
2010:C        3       file_id           := fid;
2011:C        3       access_code       := access;
2012:C        3       filler3[1]        := 0;
2013:C        3       filler3[2]        := 0;
2014:C        3       requested         := req;
2015:C        3       offset            := off;
2016:C        3     end;
2017:S
2018:C        2   packetout(unum);
2019:C        2 end;
2020:D   -24  1 (*****************************************************************************)
2021:S
2022:D        1 procedure seteofpack(unum       : unitnum;
2023:D        2                      fid        : file_id_type;
2024:D        2                      usecurptr  : boolean;
2025:D        2                      boffset    : integer);
2026:C        2 begin
2027:C        2   with packet_ptr.sseteof^ do
2028:C        3     begin
2029:C        3       setup_smh(send_mess_header,
2030:C        3                 size_to_set_eof,
2031:C        3                 req_set_eof,
2032:C        3                 0);
2033:S
2034:C        3       implicit_unlock.i := 1;
2035:C        3       file_id           := fid;
2036:C        3       use_current_ptr.i := ord(usecurptr);
2037:C        3       byte_offset       := boffset;
```

```
2038:C        3     end;
2039:S
2040:C        2   packetout(unum);
2041:C        2   packetin(unum,req_set_eof);
2042:C        2 end;
2043:D   -24  1 (*****************************************************************************)
2044:S
2045:D        1 procedure unlockpack(unum       : unitnum;
2046:D        2                      fid        : file_id_type);
2047:C        2 begin
2048:C        2   with packet_ptr.sunlock^ do
2049:C        3     begin
2050:C        3       setup_smh(send_mess_header,
2051:C        3                 size_to_funlock,
2052:C        3                 req_funlock,
2053:C        3                 0);
2054:S
2055:C        3       file_id           := fid;
2056:C        3       explicit_unlock.i := ord(true);
2057:C        3     end;
2058:S
2059:C        2   packetout(unum);
2060:C        2   packetin(unum,req_funlock);
2061:C        2 end;
2062:D   -24  1 (*****************************************************************************)
2063:S
2064:D        1 procedure volpack(unum  : unitnum);
2065:C        2 begin
2066:C        2   with packet_ptr.svol^ do
2067:C        3     begin
2068:C        3       setup_smh(send_mess_header,
2069:C        3                 size_to_volstatus,
2070:C        3                 req_volstatus,
2071:C        3                 0);
2072:S
2073:C        3       setup_vnh(volume_name_header,unum);
2074:C        3     end;
2075:S
2076:C        2   packetout(unum);
2077:C        2   packetin(unum,req_volstatus);
2078:C        2 end;
2079:D   -24  1 (*****************************************************************************)
2080:S
2081:D        1 procedure sendwritepack(unum    : unitnum;
2082:D        2                         fid     : file_id_type;
2083:D        2                         access  : integer;
2084:D        2                         req     : integer;
2085:D        2                         off     : integer;
2086:D        2                         dat     : anyptr);
2087:C        2 begin
2088:C        2   with packet_ptr.swrite^ do
2089:C        3     begin
2090:C        3       setup_smh(send_mess_header,
2091:C        3                 size_to_write + req,
2092:C        3                 req_write,
2093:C        3                 integer(dat));
2094:S
2095:C        3       implicit_unlock.i := 1;
2096:C        3       file_id           := fid;
2097:C        3       access_code       := access;
```



```
2098:C        3       filler3[1]        := 0;
2099:C        3       filler3[2]        := 0;
2100:C        3       requested         := req;
2101:C        3       offset            := off;
2102:C        3       filler8.i         := 0;
2103:C        3       flush_buffer.i    := 1;
2104:C        3       moveleft(charptr(dat)^,data,req);
2105:C        3     end;
2106:S
2107:C        2   packetout(unum);
2108:C        2 end;
2109:D   -24  1 (*****************************************************************************)
2110:S
2111:C        1 end{module srm}.
2112:S
2113:S

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

SWVOL

Description

SWVOL prompts for disc changes during stream files. Input is taken from the keyboard unit rather than the stream file.

```
Pascal [Rev 3.0M  6/ 4/84] SWVOL.TEXT              26-Dec-84 22:57:13 Page 1   *** File name: SWVOL ***

     1:D        0 $copyright 'COPYRIGHT (C) 1983 BY HEWLETT-PACKARD COMPANY'$
     2:D        0 $MODCAL$
     3:D        0 program switchvols(input, output);
     4:D        1 import sysglobals;
     5:D        1 var
     6:D     -1 1   ch: char;
     7:D    -36 1   vol_name : string[34];
     8:D    -40 1   unit_number : integer;
     9:S
    10:C        1 begin
    11:C        1   readln(vol_name);
    12:C        1   if vol_name = '*' then
    13:C        2       begin vol_name := syvid; unit_number := sysunit; end
    14:C        2   else readln(unit_number);
    15:S
    16:C        1   page;
    17:C        1   writeln; writeln; writeln;
    18:C        1   if unit_number > 0 then writeln('Please put ',vol_name,
    19:C        2                ' in unit #',unit_number:1)
    20:C        2   else writeln('Please make sure ',vol_name,' is on line');
    21:C        1   write  ('  and press the X key...');
    22:C        1   with unitable^[fibp(gfiles[2])^.funit] do
    23:C        2     repeat
    24:C        3       call(tm, fibp(gfiles[2]), readbytes, ch, 1, 0);
    25:C        3     until (ch='x') or (ch='X');
    26:C        1   writeln(ch);
    27:C        1 end.

No errors. No warnings.
          ***** Nonstandard language features enabled *****
```

SYSDEVS

Description

SYSDEVS provides declarations and "hooks" for the keyboard, the CRT, timers, interrupts, and typeahead buffer declarations and functionality.

Requirements

SYSGLOBALS.

Notes

SYSDEVS is described in Chapter 14 of the *Pascal 3.0 Procedure Library* manual.

```
     1:S          (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1984.
     4:S          All rights are reserved.  Copying or other
     5:S          reproduction of this program except for archival
     6:S          purposes is prohibited without the prior
     7:S          written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                      RESTRICTED RIGHTS LEGEND
    11:S
    12:S          Use, duplication, or disclosure by the Government
    13:S          is subject to restrictions as set forth in
    14:S          paragraph (b) (3) (B) of the Rights in Technical
    15:S          Data and Computer Software clause in
    16:S          DAR 7-104.9(a).
    17:S
    18:S          HEWLETT-PACKARD COMPANY
    19:D        0 Fort Collins, Colorado                          *)
    20:S
    21:D        0 $SYSPROG$
    22:D        0 $heap_dispose off$
    23:D        0 $iocheck off$
    24:D        0 $range off$ $ovflcheck off$
    25:D        0 $debug off$
    26:D        0 $stackcheck off$
    27:S
    28:D        0 PROGRAM SYSDEVICES;
    29:D        1 MODULE SYSDEVS;
    30:S
    31:D        1 $SEARCH 'INITLOAD'$
    32:S
    33:D        1 IMPORT SYSGLOBALS;
    34:D        1 EXPORT
    35:D        1  (* DUMMY DECLARATIONS ***********************************)
    36:D        1  TYPE
    37:D        1    KBDHOOKTYPE  = PROCEDURE(VAR STATBYTE,DATABYTE: BYTE;
    38:D        1                             VAR DOIT: BOOLEAN);
    39:D        1    OUT2TYPE     = PROCEDURE(VALUE1,VALUE2: BYTE);
    40:D        1    REQUEST1TYPE = PROCEDURE(CMD: BYTE; VAR VALUE: BYTE);
    41:D        1    BOOLPROC     = PROCEDURE(B:BOOLEAN);
    42:S
    43:D        1 (* CRT ************************************************)
    44:D        1 (***** THIS SECTION HAS HARD OFFSET REFERENCES *********)
    45:D        1 (      IN MODULES CRTB (ASSY FILE GASSM)               )
    46:D        1 TYPE
    47:D        1   CRTWORD = RECORD CASE INTEGER OF
    48:D        1               1:(HIGHLIGHTBYTE,CHARACTER: CHAR);
    49:D        1               2:(WHOLEWORD: SHORTINT);
    50:D        1               END;
    51:D        1   CRTLLOPS =(CLLPUT,CLLSHIFTL,CLLSHIFTR,CLLCLEAR,CLLDISPLAY,PUTSTATUS);
    52:D        1   CRTLLTYPE=PROCEDURE(OP:CRTLLOPS; ANYVAR POSITION:INTEGER; C:CHAR);
    53:D        1   DBCRTOPS =(DBINFO,DBEXCG,DBGOTOXY,DBPUT,DBINIT,DBCLEAR,DBCLINE,DBSCROLLUP,
    54:D        1              DBSCROLLDN,DBSCROLLL,DBSCROLLR,DBHIGHL);
    55:D        1   DBCINFO  = RECORD
    56:D        1               SAVEAREA : WINDOWP;
    57:D        1               SAVESIZE : INTEGER;
    58:D        1               DCURSORADDR : INTEGER;
    59:D        1               XMIN,XMAX,YMIN,YMAX : SHORTINT;
    60:D        1               CURSX,CURSY         : SHORTINT;
```



```
    61:D        1               C : CHAR;
    62:D        1               AREAISDBCRT : BOOLEAN;
    63:D        1             END;
    64:D        1   DBCRTTYPE=PROCEDURE(OP:DBCRTOPS; VAR DBCRT:DBCINFO);
    65:S
    66:D        1   crtconsttype = packed array [0..11] of byte;
    67:S
    68:D        1   crtfrec = packed record
    69:D        1               nobreak,stupid,slowterm,hasxycrt,
    70:D        1               haslccrt(built in crt),hasclock,
    71:D        1               canupscroll,candownscroll      :    boolean;
    72:D        1             end;
    73:S
    74:D        1   b9 = packed array[0..8] of boolean;
    75:D        1   b14= packed array[0..13] of boolean;
    76:D        1   crtcrec = packed record                              (* CRT CONTROL CHARS *)
    77:D        1               rlf,ndfs,eraseeol,
    78:D        1               eraseeos,home,
    79:D        1               escape           : char;
    80:D        1               backspace        : char;
    81:D        1               fillcount        : 0..255;
    82:D        1               clearscreen,
    83:D        1               clearline        : char;
    84:D        1               prefixed         : b9
    85:D        1           end;
    86:S
    87:D        1   crtirec = packed record                          (* CRT INFO & INPUT CHARS *)
    88:D        1               width,height      : shortint;
    89:D        1               crtmemaddr,crtcontroladdr,
    90:D        1               keybufferaddr,progstateinfoaddr:integer;
    91:D        1               keybuffersize: shortint;
    92:D        1               crtcon           : crtconsttype;
    93:D        1               right,left,down,up: char;
    94:D        1               badch,chardel,stop,
    95:D        1               break,flush,eof   : char;
    96:D        1               altmode,linedel   : char;
    97:D        1               backspace,
    98:D        1               etx,prefix        : char;
    99:D        1               prefixed          : b14 ;
   100:D        1               cursormask        : integer;
   101:D        1               spare             : integer;
   102:D        1           end;
   103:S
   104:D        1   environ = record
   105:D        1               miscinfo: crtfrec;
   106:D        1               crttype: integer;
   107:D        1               crtctrl: crtcrec;
   108:D        1               crtinfo: crtirec;
   109:D        1           end;
   110:S
   111:D        1   environptr    = ^environ;
   112:S
   113:D        1   crtkinds = (NOCRT, ALPHATYPE, BITMAPTYPE, SPECIALCRT1, SPECIALCRT2);
   114:S
   115:D        1 VAR
   116:D    -4  1   SYSCOM: ENVIRONPTR;
   117:D    -4  1   ALPHASTATE['ALPHAFLAG']       : BOOLEAN;
   118:D    -4  1   GRAPHICSTATE['GRAPHICSFLAG']  : BOOLEAN;
   119:D   -12  1   CRTIOHOOK             : AMTYPE;
   120:D   -20  1   TOGGLEALPHAHOOK        : PROCEDURE;
```

```
121:D   -28  1   TOGGLEGRAPHICSHOOK    : PROCEDURE;
122:D   -36  1   DUMPALPHAHOOK         : PROCEDURE;
123:D   -44  1   DUMPGRAPHICSHOOK      : PROCEDURE;
124:D   -52  1   UPDATECURSORHOOK      : PROCEDURE;
125:D   -60  1   CRTINITHOOK           : PROCEDURE;
126:D   -68  1   CRTLLHOOK             : CRTLLTYPE;
127:D   -76  1   DBCRTHOOK             : DBCRTTYPE;
128:D   -78  1   XPOS                  : SHORTINT; { CURSOR X POSITION }
129:D   -80  1   YPOS                  : SHORTINT; { CURSOR Y POSITION }
130:D   -82  1   CURRENTCRT            : CRTKINDS; { ACTIVE ALPHA DRIVER TYPE }
131:D   -86  1   BITMAPADDR            : INTEGER;  { ADDRESS OF BITMAP CONTROL SPACE }
132:D   -90  1   FRAMEADDR             : INTEGER;  { ADDRESS OF BITMAP FRAME BUFFER }
133:D   -92  1   REPLREGCOPY           : SHORTINT; { REGISTER COPIES FOR BITMAP DISPLAY }
134:D   -94  1   WINDOWREGCOPY         : SHORTINT; { MUST BE IN GLOBALS BECAUSE REGISTERS }
135:D   -96  1   WRITEREGCOPY          : SHORTINT; { ARE NOT READABLE -- MAY BE UNDEFINED }
136:S
137:D   -96  1  {* KEYBOARD ***************************************}
138:D   -96  1  CONST
139:D   -96  1    KBD_ENABLE     = 0; KBD_DISABLE    = 1;
140:D   -96  1    SET_AUTO_DELAY = 2; SET_AUTO_REPEAT= 3;
141:D   -96  1    GET_AUTO_DELAY = 4; GET_AUTO_REPEAT= 5;
142:D   -96  1    SET_KBDTYPE    = 6; SET_KBDLANG    = 7;
143:D   -96  1  TYPE
144:D   -96  1    STRING80PTR = ^STRING80;
145:D   -96  1    KEYBOARDTYPE = (NOKBD,LARGEKBD,SMALLKBD,ITFKBD,SPECIALKBD1,SPECIALKBD2);
146:D   -96  1    LANGTYPE = (NO_KBD,FINISH_KBD,BELGIAN_KBD,CDN_ENG_KBD,CDN_FR_KBD,
147:D   -96  1                NORWEGIAN_KBD,DANISH_KBD,DUTCH_KBD,SWISS_GR_KBD,SWISS_FR_KBD,
148:D   -96  1                SPANISH_EUR_KBD,SPANISH_LATIN_KBD,UK_KBD,ITALIAN_KBD,
149:D   -96  1                FRENCH_KBD,GERMAN_KBD,SWEDISH_KBD,SPANISH_KBD,
150:D   -96  1                KATAKANA_KBD,US_KBD,ROMAN8_KBD,NS1_KBD,NS2_KBD,NS3_KBD);
151:D   -96  1    MENUTYPE = (M_NONE,M_SYSNORM,M_SYSSHIFT,M_U1,M_U2,M_U3,M_U4);
152:D   -96  1  VAR
153:D  -104  1    KBDREQHOOK    : REQUEST1TYPE;
154:D  -112  1    KBDIOHOOK     : AMTYPE;
155:D  -120  1    KBDISRHOOK    : KBDHOOKTYPE;
156:D  -128  1    KBDPOLLHOOK   : BOOLPROC;
157:D  -130  1    KBDTYPE       : KEYBOARDTYPE;
158:D  -132  1    KBDCONFIG     : BYTE;         { KEYBOARD CONFIGURATION JUMPER }
159:D  -134  1    KBDLANG       : LANGTYPE;
160:D  -138  1    SYSMENU       : STRING80PTR;
161:D  -142  1    SYSMENUSHIFT  : STRING80PTR;
162:D  -144  1    MENUSTATE     : MENUTYPE;
163:S
164:D  -144  1 {* ENABLE / DISABLE ***********************************}
165:D  -144  1  CONST
166:D  -144  1    KBDMASK=1;RESETMASK=2;TIMERMASK=4;PSIMASK=8;FHIMASK=16;
167:D  -144  1  VAR
168:D  -152  1    MASKOPSHOOK : OUT2TYPE; { ENABLE, DISABLE }
169:S
170:D  -152  1 {* BEEPER ***************************************************}
171:D  -152  1  VAR
172:D  -160  1    BEEPERHOOK: OUT2TYPE;
173:D  -164  1    BFREQUENCY, BDURATION: BYTE;
174:S
175:D  -164  1 {* RPG ****************************************************}
176:D  -164  1  CONST
177:D  -164  1    RPG_ENABLE   = 0; RPG_DISABLE = 1;
178:D  -164  1    SET_RPG_RATE = 2; GET_RPG_RATE =3;
179:D  -164  1  VAR
180:D  -172  1    RPGREQHOOK: REQUEST1TYPE;
```



```
181:D  -180  1    RPGISRHOOK: KBDHOOKTYPE;
182:S
183:D  -180  1 {* BATTERY *************************************************}
184:D  -180  1 TYPE
185:D          BATCMDTYPE = PROCEDURE(CMD: BYTE; NUMDATA: INTEGER;
186:D  -180  1                         B1, B2, B3, B4, B5: BYTE);
187:D  -180  1   BATREADTYPE= PROCEDURE(VAR DATA: BYTE);
188:D  -180  1 VAR
189:D  -180  1   BATTERYPRESENT[-563]: BOOLEAN;
190:D  -188  1   BATCMDHOOK : BATCMDTYPE;
191:D  -196  1   BATREADHOOK: BATREADTYPE;
192:S
193:D  -196  1 {* CLOCK *************************************************}
194:D  -196  1 TYPE
195:D  -196  1   RTCTIME = PACKED RECORD
196:D  -196  1              PACKEDTIME,PACKEDDATE:INTEGER;
197:D  -196  1            END;
198:D  -196  1   CLOCKFUNC = (CGETDATE,CGETTIME,CSETDATE,CSETTIME);
199:D  -196  1   CLOCKOP   = (CGET,CSET);
200:D  -196  1   CLOCKDATA = RECORD
201:D  -196  1                CASE BOOLEAN OF
202:D  -196  1                 TRUE :(TIMETYPE:TIMEREC);
203:D  -196  1                 FALSE:(DATETYPE:DATEREC);
204:D  -196  1               END;
205:D  -196  1   CLOCKREQTYPE = PROCEDURE(CMD:CLOCKFUNC; ANYVAR DATA:CLOCKDATA);
206:D  -196  1   CLOCKIOTYPE  = PROCEDURE(CMD:CLOCKOP  ; VAR DATA:RTCTIME);
207:D  -196  1 VAR
208:D  -204  1   CLOCKREQHOOK : CLOCKREQTYPE;  { CLOCK MODULE INTERFACE }
209:D  -212  1   CLOCKIOHOOK  : CLOCKIOTYPE;   { CARD DRIVER INTERFACE }
210:S
211:D  -212  1 {* TIMER *************************************************}
212:D  -212  1 TYPE
213:D  -212  1   TIMERTYPES = (CYCLICT,PERIODICT,DELAYT,DELAY?T,MATCHT);
214:D  -212  1   TIMEROPTYPE = (SETT,READT,GETTINFO);
215:D  -212  1   TIMERDATA = RECORD
216:D  -212  1                CASE INTEGER OF
217:D  -212  1                0: (COUNT: INTEGER);
218:D  -212  1                1: (MATCH: TIMEREC);
219:D  -212  1                2: (RESOLUTION,RANGE:INTEGER);
220:D  -212  1                END;
221:D  -212  1   TIMERIOTYPE = PROCEDURE(TIMER: TIMERTYPES;OP: TIMEROPTYPE;VAR TD: TIMERDATA);
222:D  -212  1 VAR
223:D  -220  1   TIMERIOHOOK  : TIMERIOTYPE;
224:D  -228  1   TIMERISRHOOK : KBDHOOKTYPE;
225:S
226:S
227:D  -228  1 {* KEYBUFFER ********************************************}
228:D  -228  1 CONST
229:D  -228  1   KMAXBUFSIZE = 255;
230:D  -228  1 TYPE
231:S
232:D  -228  1   KOPTYPE = (KGETCHAR,KAPPEND,KNONADVANCE,KCLEAR,KDISPLAY,
233:D  -228  1                KGETLAST,KPUTFIRST);
234:D  -228  1   KBUFTYPE= PACKED ARRAY[0..KMAXBUFSIZE] OF CHAR;
235:D  -228  1   KBUFPTR = ^KBUFTYPE;
236:D  -228  1   KBUFRECPTR = ^KBUFREC;
237:D  -228  1   KBUFREC = RECORD
238:D  -228  1               ECHO: BOOLEAN;
239:D  -228  1               NON_CHAR: CHAR;
240:D  -228  1               MAXSIZE,SIZE,INP,OUTP: INTEGER;
```

```
241:D  -228  1              BUFFER: KBUFPTR;
242:D  -228  1            END;
243:S
244:D  -228  1 VAR
245:D  -232  1   KEYBUFFER : KBUFRECPTR;
246:D  -240  1   KBDWAITHOOK: PROCEDURE;
247:D  -248  1   KBDRELEASEHOOK: PROCEDURE;
248:D  -256  1   STATUSLINE: PACKED ARRAY[0..7] OF CHAR;
249:D  -256  1   {0  s or f = STEP/FLASH IN PROGRESS (WAITING FOR TRAP #0)}
250:D  -256  1   {1..5  last executed/current line number }
251:D  -256  1   {6  S=SYSTEM  U=USER  DEFINITION FOR ITF SOFT KEYS}
252:D  -256  1   {   BLANK FOR NON ITF KEYBOARDS }
253:D  -256  1   {7  RUNLIGHT }
254:S
255:D  -256  1 {* KEY TRANSLATION SERVICES ********************************}
256:D  -256  1 TYPE
257:D  -256  1   KEYTRANSTYPE =(KPASSTHRU,KSHIFT_EXTC,KPASS_EXTC);
258:D  -256  1   KEYTYPE = (ALPHA_KEY,NONADV_KEY,SPECIAL_KEY,IGNORED_KEY,NONA_ALPHA_KEY);
259:D  -256  1   { ADDED NONA_ALPHA_KEY 5/9/84 RQ/SFB }
260:S
261:D  -256  1   LANGCOMREC = RECORD
262:D  -256  1                STATUS : BYTE;
263:D  -256  1                DATA   : BYTE;
264:D  -256  1                KEY    : CHAR;
265:D  -256  1                RESULT : KEYTYPE;
266:D  -256  1                SHIFT,CONTROL,EXTENSION: BOOLEAN;
267:D  -256  1              END;
268:D  -256  1   LANGKEYREC = RECORD
269:D  -256  1                NO_CAPSLOCK: BOOLEAN;
270:D  -256  1                NO_SHIFT   : BOOLEAN;
271:D  -256  1                NO_CONTROL : BOOLEAN;
272:D  -256  1                NO_EXTENSION : BOOLEAN;
273:D  -256  1                KEYCLASS   : KEYTYPE;
274:D  -256  1                KEYS : ARRAY[BOOLEAN] OF CHAR;
275:D  -256  1              END;
276:D  -256  1   LANGRECORD= RECORD
277:D  -256  1                CAN_NONADV: BOOLEAN;
278:D  -256  1                LANGCODE  : LANGTYPE;
279:D  -256  1                SEMANTICS : PROCEDURE;
280:D  -256  1                KEYTABLE  : ARRAY[0..127] OF LANGKEYREC;
281:D  -256  1              END;
282:D  -256  1   LANGPTR   = ^LANGRECORD;
283:D  -256  1 VAR
284:D  -268  1   LANGCOM   : LANGCOMREC;
285:D  -276  1   LANGTABLE : ARRAY[0..1] OF LANGPTR;
286:D  -278  1   LANGINDEX : 0..1;
287:D  -286  1   KBDTRANSHOOK : KBDHOOKTYPE;
288:D  -288  1   TRANSMODE : KEYTRANSTYPE;
289:D  -291  1   KBDSYSMODE, KBDALTLOCK, KBDCAPSLOCK : BOOLEAN;
290:S
291:D  -291  1 {* HPHIL ************************************************}
292:D  -291  1 {MOVED INTO SYSDEVS 4/6/84 SFB}
293:D  -291  1 const
294:D  -291  1   le_configured = hex('80');
295:D  -291  1   le_error      = hex('81');
296:D  -291  1   le_timeout    = hex('82');
297:D  -291  1   le_loopdown   = hex('84');
298:S
299:D  -291  1   lmaxdevices   = 7;
300:S
```



```
301:D  -291  1 type
302:D  -291  1   loopdvrop   = (datastarting,dataended,resetdevice);
303:D  -291  1   loopdvrproc = procedure(op:loopdvrop);
304:S
305:D  -291  1   {HPHILOP DEFINED AS NEW TYPE 4/6/84 SFB}
306:D  -291  1   HPHILOP     = (RAWSHIFTOP,NORMSHIFTOP,CHECKLOOPOP,CONFIGUREOP,LCOMMANDOP);
307:D  -291  1   {5 PROCEDURES HOOKED AS TYPE HPHILCMDPROC 4/6/84 SFB}
308:D  -291  1   HPHILCMDPROC = PROCEDURE(OP : HPHILOP);
309:S
310:S
311:D  -291  1   descriprec = packed record    { DEVICE DESCRIBE RECORD }
312:D  -291  1                 case boolean of
313:D  -291  1                 true :(id       : byte;
314:D  -291  1                        twosets  : boolean;
315:D  -291  1                        abscoords: boolean;
316:D  -291  1                        size16   : boolean;
317:D  -291  1                        hasprompts:boolean;
318:D  -291  1                        reserved : 0..3;
319:D  -291  1                        numaxes  : 0..3;
320:D  -291  1                        counts   : shortint;
321:D  -291  1                        maxcountx: shortint;
322:D  -291  1                        maxcounty: shortint;
323:D  -291  1                        maxcountz: shortint;
324:D  -291  1                        nprompts : 0..7;
325:D  -291  1                        nbuttons : 0..7);
326:D  -291  1                 false:(darray : array[1..11] of char);
327:D  -291  1               end;
328:S
329:D  -291  1   devicerec = record
330:D  -291  1                devstate : integer;
331:D  -291  1                descrip : descriprec;
332:D  -291  1                opsproc  : loopdvrproc;
333:D  -291  1                dataproc : kbdhooktype;
334:D  -291  1              end;
335:S
336:D  -291  1   loopdvrptr = ^loopdriverrec;
337:D  -291  1   loopdriverrec = record
338:D  -291  1                    lowid,highid,daddr : byte;
339:D  -291  1                    opsproc  : loopdvrproc;
340:D  -291  1                    dataproc : kbdhooktype;
341:D  -291  1                    next     : loopdvrptr;
342:D  -291  1                  end;
343:S
344:D  -291  1   LOOPCONTROLREC = RECORD                    {REDEFINED AS RECORD - 4/6/84 SFB}
345:D  -291  1         rawmode : boolean;
346:D  -291  1         loopdevices : array[1..lmaxdevices] of devicerec;
347:D  -291  1         loopdevice : 1..lmaxdevices;
348:D  -291  1         loopcmd    : byte;    { last loop command sent }
349:D  -291  1         loopdata   : byte;    { data bye in / out }
350:D  -291  1         looperror  : boolean; { error occured on last operation }
351:D  -291  1         loopinconfig:boolean; { now doing reconfigure }
352:D  -291  1         loopcmddone: boolean; { last sent command is done }
353:D  -291  1         loopisok   : boolean; { loop is configured }
354:D  -291  1         loopdevreading: boolean; { reading poll data }  { 3.0 BUG #39 3/17/84 }
355:D  -291  1   END;
356:S
357:D  -291  1 var
358:D  -291  1
359:D  -296  1   loopdriverlist : loopdvrptr;
360:D  -300  1   LOOPCONTROL    : ^LOOPCONTROLREC;     {4/6/84 SFB}
```

```
361:D  -308  1   HPHILCMDHOOK   : HPHILCMDPROC;        {4/6/84 SFB}
362:S
363:D  -308  1 {-------------------------------------------------------------------------------}
364:D  -308  1 PROCEDURE SYSDEV_INIT;
365:D  -308  1 (* BEEPER ************************************************}
366:D  -308  1 PROCEDURE BEEP;
367:D  -308  1 PROCEDURE BEEPER(FREQUENCY,DURATION:BYTE);
368:D  -308  1 (* RPG *************************************************}
369:D  -308  1 PROCEDURE SETRPGRATE(RATE : BYTE);
370:D  -308  1 (* KEYBOARD ********************************************}
371:D  -308  1 PROCEDURE KBDSETUP(CMD,VALUE:BYTE);
372:D        1 PROCEDURE KBDIO(FP: FIBP; REQUEST: AMREQUESTTYPE;
373:D  -308  2                  ANYVAR BUFFER: WINDOW; BUFSIZE,POSITION: INTEGER);
374:D  -308  1 procedure lockedaction(a: action);
375:D  -308  1 (* CRT *************************************************}
376:D        1 PROCEDURE CRTIO(FP: FIBP; REQUEST: AMREQUESTTYPE;
377:D  -308  2                  ANYVAR BUFFER: WINDOW; BUFSIZE,POSITION: INTEGER);
378:D  -308  1 PROCEDURE DUMMYCRTLL(OP:CRTLLOPS; ANYVAR POSITION:INTEGER; C:CHAR);
379:D  -308  1 (* BATTERY *********************************************}
380:D  -308  1 PROCEDURE BATCOMMAND(CMD:BYTE; NUMDATA:INTEGER; B1, B2, B3, B4, B5: BYTE);
381:D  -308  1 FUNCTION  BATBYTERECEIVED:BYTE;
382:D  -308  1 (* CLOCK ***********************************************}
383:D  -308  1 function  sysclock: integer;   {centiseconds from midnight}
384:D  -308  1 procedure sysdate (var thedate: daterec);
385:D  -308  1 procedure systime (var thetime: timerec);
386:D  -308  1 procedure setsysdate ( thedate: daterec);
387:D  -308  1 procedure setsystime ( thetime: timerec);
388:D  -308  1 (* KEYBUFFER *******************************************}
389:D  -308  1 PROCEDURE KEYBUFOPS(OP:KOPTYPE; VAR C: CHAR);
390:D  -308  1 (* STATUSLINE *****************************************}
391:D  -308  1 PROCEDURE SETSTATUS(N:INTEGER; C:CHAR);
392:D  -308  1 FUNCTION  RUNLIGHT:CHAR;
393:D  -308  1 PROCEDURE SETRUNLIGHT(C:CHAR);
394:S
395:D  -308  1 IMPLEMENT
396:S
397:D  -308  1 (* GENERAL PURPOSE DUMMY PROCEDURES ********************}
398:D        1   PROCEDURE DUMMYPROC;
399:C        2   BEGIN END;
400:D        1   PROCEDURE DUMMYOUT2(VALUE1, VALUE2 : BYTE);
401:C        2   BEGIN END;
402:D        1   PROCEDURE DUMMYKBD(VAR STATBYTE,DATABYTE : BYTE;
403:D        2                      VAR DOIT:BOOLEAN);
404:C        2   BEGIN END;
405:D        1   PROCEDURE DUMMYREQ1(CMD : BYTE; VAR VALUE:BYTE);
406:C        2   BEGIN END;
407:D        1   PROCEDURE DUMMYTM(FP: FIBP; REQUEST: AMREQUESTTYPE;
408:D        2                     ANYVAR BUFFER: WINDOW; BUFSIZE,POSITION: INTEGER);
409:C        2   BEGIN END;
410:D        1   PROCEDURE DUMMYBOOLPROC(B:BOOLEAN);
411:C        2   BEGIN END;
412:D  -308  1 (* BEEPER ************************************************}
413:D        1   PROCEDURE BEEP;
414:C        2   BEGIN CALL(BEEPERHOOK,BFREQUENCY,BDURATION);
415:C        2   END;
416:D        1   PROCEDURE BEEPER(FREQUENCY,DURATION:BYTE);
417:C        2   BEGIN CALL(BEEPERHOOK,FREQUENCY,DURATION);
418:C        2   END;
419:S
420:D  -308  1 (* RPG *************************************************}
```



```
421:D        1   PROCEDURE SETRPGRATE(RATE:BYTE);
422:D    -2  2   VAR   IRATE : BYTE;
423:C        2   BEGIN
424:C        2     IRATE := RATE; CALL(RPGREQHOOK,SET_RPG_RATE,IRATE);
425:C        2   END;
426:S
427:D  -308  1 (* KEYBOARD ************************************************}
428:D        1   PROCEDURE KBDSETUP(CMD,VALUE :BYTE);
429:D    -2  2   VAR IVALUE : BYTE;
430:C        2   BEGIN IVALUE := VALUE; CALL(KBDREQHOOK,CMD,IVALUE);
431:C        2   END;
432:D        1   PROCEDURE KBDIO(FP: FIBP; REQUEST: AMREQUESTTYPE;
433:D        2                   ANYVAR BUFFER: WINDOW; BUFSIZE,POSITION: INTEGER);
434:C        2   BEGIN CALL(KBDIOHOOK,FP,REQUEST,BUFFER,BUFSIZE,POSITION); END;
435:S
436:D  -308  1 (* BATTERY *************************************************}
437:D        1   PROCEDURE BATCOMMAND(CMD:BYTE; NUMDATA:INTEGER; B1, B2, B3, B4, B5: BYTE);
438:C        2   BEGIN CALL(BATCMDHOOK, CMD, NUMDATA, B1, B2, B3, B4, B5);
439:C        2   END;
440:D        1   FUNCTION  BATBYTERECEIVED:BYTE;
441:D    -2  2   VAR DATA : BYTE;
442:C        2   BEGIN CALL(BATREADHOOK,DATA); BATBYTERECEIVED := DATA;
443:C        2   END;
444:S
445:D        1   PROCEDURE DUMMYBATCMD(CMD:BYTE; NUMDATA:INTEGER; B1, B2, B3, B4, B5: BYTE);
446:C        2   BEGIN END;
447:D        1   PROCEDURE DUMMYBATREAD(VAR DATA:BYTE);
448:C        2   BEGIN END;
449:S
450:D  -308  1 (* CLOCK *************************************************}
451:D        1   PROCEDURE DUMMYCLOCKSYS(VAR STIME:INTEGER);
452:C        2   BEGIN END;
453:D        1   PROCEDURE DUMMYCLOCKREQ(CMD:CLOCKFUNC; ANYVAR DATA:CLOCKDATA);
454:C        2   BEGIN END;
455:D        1   PROCEDURE DUMMYCLOCKIO(CMD:CLOCKOP  ; VAR DATA:RTCTIME);
456:C        2   BEGIN END;
457:D        1   function sysclock:integer;
458:D    -8  2   var ltime: rtctime;
459:C        2   begin
460:C        2     CALL(CLOCKIOHOOK,CGET,LTIME); sysclock := ltime.packedtime;
461:C        2   end;
462:S
463:D        1   procedure sysdate (var thedate: daterec);
464:C        2   BEGIN CALL(CLOCKREQHOOK,CGETDATE,THEDATE);
465:C        2   END;
466:D        1   procedure systime (var thetime: timerec);
467:C        2   BEGIN CALL(CLOCKREQHOOK,CGETTIME,THETIME);
468:C        2   END;
469:D        1   procedure setsysdate ( thedate: daterec);
470:D    -2  2   VAR D:DATEREC;
471:C        2   BEGIN D:=THEDATE; CALL(CLOCKREQHOOK,CSETDATE,D);
472:C        2   END;
473:D        1   procedure setsystime ( thetime: timerec);
474:D    -4  2   VAR T:TIMEREC;
475:C        2   BEGIN T:=THETIME; CALL(CLOCKREQHOOK,CSETTIME,T);
476:C        2   END;
477:D        1   PROCEDURE DUMMYTIMERIO(TIMER: TIMERTYPES;OP: TIMEROPTYPE;VAR TD: TIMERDATA);
478:C        2   BEGIN IF OP=READT THEN TD.COUNT:=-1
479:C        3         ELSE
480:C        3         IF OP=GETTINFO THEN TD.RESOLUTION:=0;
```

```
481:C        2   END;
482:S
483:D  -308  1 (* CRT ************************************************)
484:D        1   PROCEDURE CRTIO(FP: FIBP; REQUEST: AMREQUESTTYPE;
485:D        2                   ANYVAR BUFFER: WINDOW; BUFSIZE,POSITION: INTEGER);
486:C        2   BEGIN CALL(CRTIOHOOK,FP,REQUEST,BUFFER,BUFSIZE,POSITION); END;
487:D        1   PROCEDURE DUMMYCRTLL(OP:CRTLLOPS; ANYVAR POSITION:INTEGER; C:CHAR);
488:C        2   BEGIN END;
489:D        1   PROCEDURE DUMMYDBCRT(OP:DBCRTOPS; VAR DBCRT:DBCINFO);
490:C        2   BEGIN END;
491:S
492:D        1   procedure lockedaction(a: action);
493:D        2   label 1;
494:D   -4   2   var i: integer;
495:C        2   begin
496:C        2    if locklevel = 0 then call(a)
497:C        3    else
498:C        3    begin
499:C        3      i := actionspending;
500:C        3      while i>0 do if deferredaction[i]=a then goto 1 else i := i - 1;
501:C        3      if actionspending = 10 then beep
502:C        4      else
503:C        4      begin
504:C        4        actionspending := actionspending + 1;
505:C        4        deferredaction[actionspending] := a;
506:C        4      end;
507:C        3    end;
508:C        2   1:
509:C        2   end;
510:S
511:D  -308  1 (* KEYBUFFER ******************************************)
512:D        1   PROCEDURE KEYBUFOPS(OP:KOPTYPE; VAR C: CHAR);
513:D        2   VAR
514:D    -8  2     POSITION,TEMP1 : INTEGER;
515:S
516:D        2     PROCEDURE ADVANCE(VAR P:INTEGER);
517:C        3     BEGIN P:=(P+1) MOD KEYBUFFER^.MAXSIZE; END;
518:S
519:D        2     PROCEDURE BACKUP(VAR P:INTEGER);
520:C        3     BEGIN IF P=0 THEN P:=KEYBUFFER^.MAXSIZE-1 ELSE P:=P-1;
521:C        3     END;
522:S
523:C        2   BEGIN { KEYBUFOPS }
524:C        2     WITH KEYBUFFER^ DO
525:C        3     CASE OP OF
526:C        4     KGETCHAR:
527:C        4       BEGIN     { MUST NOT CALL IF SIZE=0 }
528:C        4         IF ECHO THEN CALL(CRTLLHOOK,CLLSHIFTL,POSITION,' ');
529:C        4         C:=BUFFER^[OUTP]; SIZE:=SIZE-1; ADVANCE(OUTP);
530:C        4       END;
531:C        4     KAPPEND,KNONADVANCE:
532:C        4       BEGIN     { MUST NOT CALL IF SIZE>=MAXSIZE }
533:C        4         IF ECHO THEN CALL(CRTLLHOOK,CLLPUT,SIZE,C);
534:C        4         IF OP=KAPPEND THEN
535:C        5         BEGIN
536:C        5           SIZE:=SIZE+1; BUFFER^[INP]:=C ;ADVANCE(INP); NON_CHAR:=' ';
537:C        5         END
538:C        5         ELSE NON_CHAR:=C;
539:C        4       END;
540:C        4     KCLEAR:
```



```
541:C        4       BEGIN
542:C        4         IF ECHO THEN CALL(CRTLLHOOK,CLLCLEAR,POSITION,' ');
543:C        4         SIZE:=0; INP:=OUTP; NON_CHAR:=' ';
544:C        4       END;
545:C        4     KDISPLAY:
546:C        4       IF ECHO THEN
547:C        5       BEGIN
548:C        5         CALL(CRTLLHOOK,CLLCLEAR,POSITION,' ');
549:C        5         POSITION:= 0; TEMP1:=OUTP;
550:C        5         WHILE POSITION<SIZE DO
551:C        6         BEGIN
552:C        6           CALL(CRTLLHOOK,CLLPUT,POSITION,BUFFER^[TEMP1]);
553:C        6           ADVANCE(TEMP1); POSITION:=POSITION+1;
554:C        6         END;
555:C        5         IF NON_CHAR<>' ' THEN
556:C        6           CALL(CRTLLHOOK,CLLPUT,POSITION,NON_CHAR);
557:C        5       END;
558:C        4     KGETLAST:
559:C        4       BEGIN
560:C        4         IF ECHO AND (NON_CHAR<>' ') THEN CALL(CRTLLHOOK,CLLPUT,SIZE,' ');
561:C        4         IF SIZE>0 THEN
562:C        5         BEGIN
563:C        5           BACKUP(INP); C:=BUFFER^[INP]; SIZE:=SIZE-1; NON_CHAR:=' ';
564:C        5           IF ECHO THEN CALL(CRTLLHOOK,CLLPUT,SIZE,' ');
565:C        5         END;
566:C        4       END;
567:C        4     KPUTFIRST:
568:C        4       BEGIN     { CALL ONLY IF SIZE<=MAXSIZE }
569:C        4         IF ECHO THEN
570:C        5         BEGIN
571:C        5           POSITION:=0;
572:C        5           CALL(CRTLLHOOK,CLLSHIFTR,POSITION,' ');
573:C        5           CALL(CRTLLHOOK,CLLPUT,POSITION,C);
574:C        5         END;
575:C        4         BACKUP(OUTP); BUFFER^[OUTP]:=C; SIZE:=SIZE+1;
576:C        4       END;
577:C        4     END; { CASE }
578:C        2   END; { KEYBUFOPS }
579:S
580:D        1   PROCEDURE SETSTATUS(N:INTEGER; C:CHAR);
581:D    -1  2   VAR C1:CHAR;
582:C        2   BEGIN STATUSLINE[N]:=C; C1:=C; CALL(CRTLLHOOK,PUTSTATUS,N,C1);
583:C        2   END;
584:D        1   FUNCTION  RUNLIGHT:CHAR;
585:C        2   BEGIN RUNLIGHT:=STATUSLINE[7];
586:C        2   END;
587:D        1   PROCEDURE SETRUNLIGHT(C:CHAR);
588:C        2   BEGIN SETSTATUS(7,C);
589:C        2   END;
590:S
591:D  -308  1 (* HPHIL ************************************************)
592:D  -308  1 {DUMMY ADDED 4/6/84 SFB}
593:D        1   PROCEDURE DUMMYHPHILCMD(OP : HPHILOP);
594:C        2   BEGIN  END;
595:S
596:D  -308  1 (* INITIALIZE ALL HOOKS ********************************************************)
597:D        1   PROCEDURE SYSDEV_INIT;
598:D    -2  2   VAR I : SHORTINT;
599:C        2   BEGIN (SYSDEV_INIT)
600:C        2     MASKOPSHOOK:= DUMMYOUT2;
```

```
   601:C        2     BEEPERHOOK := DUMMYOUT2;
   602:S
   603:C        2     RPGREQHOOK := DUMMYREQ1;
   604:C        2     RPGISRHOOK := DUMMYKBD;
   605:S
   606:C        2     BATREADHOOK:= DUMMYBATREAD;
   607:C        2     BATCMDHOOK := DUMMYBATCMD;
   608:S
   609:C        2     CLOCKREQHOOK := DUMMYCLOCKREQ;
   610:C        2     CLOCKIOHOOK  := DUMMYCLOCKIO;
   611:C        2     TIMERIOHOOK  := DUMMYTIMERIO;
   612:C        2     TIMERISRHOOK := DUMMYKBD;
   613:S
   614:C        2     KBDREQHOOK := DUMMYREQ1;
   615:C        2     KBDISRHOOK := DUMMYKBD;
   616:C        2     KBDTYPE    := NOKBD;
   617:C        2     KBDLANG    := NO_KBD;
   618:C        2     KBDIOHOOK  := DUMMYTM;
   619:C        2     KBDPOLLHOOK:= DUMMYBOOLPROC;
   620:C        2     MENUSTATE  := M_NONE;
   621:S
   622:C        2     TOGGLEALPHAHOOK       := DUMMYPROC;
   623:C        2     TOGGLEGRAPHICSHOOK    := DUMMYPROC;
   624:C        2     DUMPALPHAHOOK         := DUMMYPROC;
   625:C        2     DUMPGRAPHICSHOOK      := DUMMYPROC;
   626:C        2     UPDATECURSORHOOK      := DUMMYPROC;
   627:C        2     CRTINITHOOK           := DUMMYPROC;
   628:C        2     CRTIOHOOK  := DUMMYTM;
   629:C        2     CRTLLHOOK  := DUMMYCRTLL;
   630:C        2     DBCRTHOOK  := DUMMYDBCRT;
   631:C        2     CURRENTCRT           := NOCRT;
   632:C        2     NEW(SYSCOM);
   633:S
   634:C        2     NEW(KEYBUFFER);
   635:C        2     WITH KEYBUFFER^ DO
   636:C        3     BEGIN
   637:C        3       NEW(BUFFER);
   638:C        3       ECHO      :=TRUE;
   639:C        3       MAXSIZE   :=KMAXBUFSIZE;
   640:C        3       INP:=0; OUTP:=0; SIZE:=0;
   641:C        3       NON_CHAR  :=' ';
   642:C        3     END;
   643:S
   644:C        2     FOR I:=0 TO 7 DO STATUSLINE[I]:=' ';
   645:C        2     KBDWAITHOOK    := DUMMYPROC;
   646:C        2     KBDRELEASEHOOK := DUMMYPROC;
   647:S
   648:C        2     FOR I:=0 TO 1 DO LANGTABLE[I]:=NIL;
   649:C        2     LANGINDEX:=0;
   650:C        2     KBDSYSMODE:=TRUE; KBDALTLOCK:=FALSE; KBDCAPSLOCK:=FALSE;
   651:C        2     KBDTRANSHOOK := DUMMYKBD; TRANSMODE :=KPASSTHRU;
   652:S
   653:C        2     HPHILCMDHOOK    := DUMMYHPHILCMD;    {4/6/84 SF8}
   654:C        2     LOOPCONTROL     := NIL;              {4/9/84 SF8}
   655:C        2     LOOPDRIVERLIST  := NIL;              {4/9/84 SF8}
   656:S
   657:C        2   END; {SYSDEV_INIT}
   658:S
   659:C        1 END; {MODULE}
   660:S
```



```
   661:D        1 IMPORT SYSDEVS;
   662:C        1 BEGIN
   663:C        1   SYSDEV_INIT;
   664:C        1   {MARKUSER}
   665:C        1 END. {PROGRAM}

No errors. No warnings.
            ***** Nonstandard language features enabled *****
```

Description

TAIL invokes the loader to execute TABLE and STARTUP.

Usage

TAIL is placed at the end of INITLIB, since it terminates the boot-up process.

```
Pascal [Rev 3.0M  6/ 4/84] TAIL.TEXT              26-Dec-84 21:11:28 Page 1   *** File name: TAIL ***

     1:S                                                    (*
     2:S
     3:S           (c) Copyright Hewlett-Packard Company, 1984.
     4:S          All rights are reserved.  Copying or other
     5:S          reproduction of this program except for archival
     6:S          purposes is prohibited without the prior
     7:S          written consent of Hewlett-Packard Company.
     8:S
     9:S
    10:S                        RESTRICTED RIGHTS LEGEND
    11:S
    12:S          Use, duplication, or disclosure by the Government
    13:S          is subject to restrictions as set forth in
    14:S          paragraph (b) (3) (B) of the Rights in Technical
    15:S          Data and Computer Software clause in
    16:S          DAR 7-104.9(a).
    17:S
    18:S          HEWLETT-PACKARD COMPANY
    19:D        0 Fort Collins, Colorado                        *)
    20:S
    21:S
    22:D        0 $modcal$
    23:D        0 $heap_dispose off$
    24:D        0 $iocheck off$
    25:D        0 $range off$ $ovflcheck off$
    26:D        0 $debug off$
    27:D        0 $stackcheck off$
    28:D        0 $search  'INITLOAD', 'ASM', 'INIT', 'SYSDEVS'$
    29:S
    30:D        0 program tail(output);
    31:S
    32:D        1 import sysglobals, loader, ldr, misc, asm, bootdammodule, sysdevs;
    33:S
    34:D   -66  1 var start, table, systemname: string[20];
    35:S
    36:C        1 begin
    37:S
    38:C        1 ci_switch;              {move onto supervisor stack}
    39:S
    40:C        1 markuser;               {retain memory allocation from this point}
    41:S
    42:S          {*****************************************************************************
    43:C        1  INTRODUCTORY MESSAGE }
    44:S
    45:C        1    writeln('Copyright 1984 Hewlett-Packard Company.');
    46:S
    47:S          {*****************************************************************************
    48:C        1  COMPUTE NAMES OF STANDARD FILES}
    49:C        1
    50:C        1    start := 'STARTUP'; table := 'TABLE';
    51:C        1    setstrlen(systemname, 0); strmove(10, sysname, 1, systemname, 1);
    52:C        1    systemname := strrtrim(systemname);
    53:C        1    if systemname<>'SYSTEM_P' then
    54:C        2     begin
    55:C        2     setstrlen(start, 5);
***WARNING: (line   56): STRPOS does not conform to HP standard, see $SWITCH_STRPOS$
    56:C        2     if strpos('SYS', systemname)=1 then
    57:C        3       begin
    58:C        3       strdelete(systemname, 1, 3);
***WARNING: (line   59): STRPOS does not conform to HP standard, see $SWITCH_STRPOS$
```

```
Pascal [Rev 3.0M  6/ 4/84] TAIL.TEXT              26-Dec-84 21:11:28 Page 2   *** File name: TAIL ***

    59:C        3       if strpos('TEM_', systemname)=1 then strdelete(systemname, 1, 4);
    60:C        3       end;
    61:C        2     start := start + systemname;
    62:C        2     table := table + systemname;
    63:C        2     end;
    64:S
    65:S          {*****************************************************************************
    66:C        1  LOAD MAIN PROGRAM }
    67:S
    68:C        1    try load(nodestr+start, true);
    69:C        2    recover if escapecode<>-1 then escape(escapecode);
    70:C        1    markuser;
    71:S
    72:S          {*****************************************************************************
    73:C        1  ALLOW RECONFIGURATION OF UNIT TABLE}
    74:S
    75:C        1    try load(nodestr+table, false);      go;     {load new unit table}
    76:C        2    recover if escapecode<>-1 then escape(escapecode);
    77:C        1    releaseuser;
    78:S
    79:S
    80:S          {*****************************************************************************
    81:C        1  END OF SYSTEM INITIALIZATION}
    82:C        1
    83:S
    84:C        1    try
    85:C        2    loadrom(start);
    86:C        2    if entrypoint = nil then load('*'+start, false);
    87:C        2    CALL(MASKOPSHOOK,KBDMASK+RESETMASK,0); {enable the keyboard}
    88:C        2    go;                          {run the program}
    89:C        2    recover printerror(escapecode, ioresult);
    90:C        1
    91:C        1  writeln(output, 'SYSTEM FINISHED');
    92:C        1  CALL(MASKOPSHOOK,KBDMASK+RESETMASK,0); {enable the keyboard}
    93:C        1  while true do try while true do; recover;              {TRY TO TRAP STOP KEY}
    94:S
    95:C        1 end.
    96:S

No errors, 2 warnings.
             ***** Nonstandard language features enabled *****
```

TAPEBKUP

Description

TAPEBKUP is the Command Set '80 disc/tape backup utility program.

Usage

TAPEBKUP is used for complete CS80 media backup and restoration, certification of CS80 tapes and complete media verification of both discs and tapes.

```
Pascal [Rev 3.0M  6/ 4/84] TAPEBKUP.TEXT          26-Dec-84 23:02:22 Page 1   *** File name: TAPEBKUP ***

     1:D        0 $modcal, switch_strpos, debug off, range off, ovflcheck off$
     2:D        0
     3:D        0 $search 'IOLIB:KERNEL', 'INIT:DRVASM', 'INIT:DISCHPIB',
     4:D        0         'INIT:CS80'$
     5:S
     6:S
     7:S
     8:D        0 program tapebkup(keyboard,input,output);  {CS80 streaming backup tape utility}
     9:S
    10:S
    11:D        1 module CS80tbr; {tape backup routines}
    12:S
    13:D        1 import
    14:D        1   sysglobals, bkgnd, discHPIB, CS80;
    15:S
    16:D        1 export
    17:D        1
    18:S            {
    19:S              NOTE: the following functions each perform a COMPLETE transaction. They:
    20:S                    .  issue a (device or transparent) command          (Command message)
    21:S                    .  transfer data if applicable                      (Execution message)
    22:S                    .  return the resulting QSTAT                       (Reporting message)
    23:D        1   }
    24:D        1   function set_specified_unitvol      (uep: uep_type; unit, vol: unsgn4): unsgn8;
    25:D        1   function initialize_media           (uep: uep_type; options, interleave: unsgn8): unsgn8;
    26:D        1   function set_address_and_return_mode(uep: uep_type; address: integer): unsgn8;
    27:D        1   function write_file_mark            (uep: uep_type): unsgn8;
    28:D        1   function copy_start_address         (uep: uep_type; var csa_bytes: sva_type): unsgn8;
    29:D        1   function unload                     (uep: uep_type): unsgn8;
    30:D        1   function locate_and_verify         (uep: uep_type; length: integer): unsgn8;
    31:D        1   function volume_copy_data          (source_uep: uep_type;
    32:D        2                                      source_blk_addr: integer;
    33:D        2                                      destination_uep: uep_type;
    34:D        2                                      destination_blk_addr: integer): unsgn8;
    35:S
    36:S
    37:D        1 implement {CS80tbr}
    38:S
    39:S
    40:D        1 type
    41:D        1   setunitvol_type = {SET_UNIT/SET_VOLUME command pair}
    42:D        1     packed record
    43:D        1       setunit: CMD_type;
    44:D        1       setvol: CMD_type;
    45:D        1     end;
    46:S
    47:S
    48:D        1 function suv_CMD_pair(uep: uep_type): setunitvol_type;
    49:C        2   begin {suv_CMD_pair}
    50:C        2     suv_CMD_pair.setunit := CMD_type(signed16(CMDset_unit_0)+uep^.du);
    51:C        2     suv_CMD_pair.setvol  := CMD_type(signed16(CMDset_vol_0)+uep^.dv);
    52:C        2   end; {suv_CMD_pair}
```

```
Pascal [Rev 3.0M  6/ 4/84] TAPEBKUP.TEXT          26-Dec-84 23:02:22 Page 2   *** File name: TAPEBKUP ***

    53:D        1 $page$
    54:S
    55:D        1 function set_specified_unitvol(uep: uep_type; unit, vol: unsgn4): unsgn8;
    56:S            {
    57:S              issue the following command sequence:
    58:S                .  SET_UNIT
    59:S                .  SET_VOLUME
    60:S              return the QSTAT byte
    61:D        2   }
    62:D        2   var
    63:D    -2  2     suv: setunitvol_type;
    64:C        2   begin {set_specified_unitvol}
    65:C        2     suv.setunit := CMD_type(signed16(CMDset_unit_0)+unit);
    66:C        2     suv.setvol  := CMD_type(signed16(CMDset_vol_0)+vol);
    67:C        2     HPIBshort_msge_out(uep, command_sec, addr(suv), sizeof(suv));
    68:C        2     HPIBwait_for_ppol(uep);
    69:C        2     set_specified_unitvol := qstat(uep);
    70:C        2   end; {set_specified_unitvol}
    71:S
    72:S
    73:D        1 function initialize_media(uep: uep_type; options, interleave: unsgn8): unsgn8;
    74:S            {
    75:S              issue the following command sequence:
    76:S                .  SET_UNIT
    77:S                .  SET_VOLUME
    78:S                .  INITIALIZE_MEDIA
    79:S              return the QSTAT byte
    80:D        2   }
    81:D        2   var
    82:D        2     im: {the 5 bytes in the command message}
    83:D        2       packed record
    84:D        2         setunitvol: setunitvol_type;
    85:D        2         initmedia: CMD_type;
    86:D        2         options: unsgn8;
    87:D        2         interleave: unsgn8;
    88:D    -6  2       end;
    89:C        2   begin {initialize_media}
    90:C        2     im.setunitvol := suv_CMD_pair(uep);
    91:C        2     im.initmedia  := CMDinit_media;
    92:C        2     im.options    := options;
    93:C        2     im.interleave := interleave;
    94:C        2     HPIBshort_msge_out(uep, command_sec, addr(im), sizeof(im));
    95:C        2     HPIBwait_for_ppol(uep);
    96:C        2     initialize_media := qstat(uep);
    97:C        2   end; {initialize_media}
```

```
    98:D        1 $page$
    99:S
   100:D        1 function set_address_and_return_mode(uep: uep_type; address: integer): unsgn8;
   101:S            {
   102:S              issue the following command sequence:
   103:S                .  SET_UNIT
   104:S                .  SET_VOLUME
   105:S                .  SET_ADDRESS_RETURN_MODE
   106:S                .  SET_ADDRESS
   107:S              return the QSTAT byte
   108:D        2   }
   109:D        2   var
   110:D        2     sarm: {the 12 bytes in the command message}
   111:D        2       packed record
   112:D        2         setunitvol: setunitvol_type;
   113:D        2         setretadd: CMD_type;
   114:D        2         retaddmode: unsgn8;
   115:D        2         nop: CMD_type;
   116:D        2         setadd: CMD_type;
   117:D        2         sva: sva_type;
   118:D   -12  2       end;
   119:C        2   begin {set_address_and_return_mode}
   120:C        2     sarm.setunitvol := suv_CMD_pair(uep);
   121:C        2     sarm.setretadd  := CMDset_retadd_mode;
   122:C        2     sarm.retaddmode := 0;  {single vector!}
   123:C        2     sarm.nop        := CMDno_op;
   124:C        2     sarm.setadd     := CMDset_address_1V;
   125:C        2     sarm.sva.utb    := 0;
   126:C        2     sarm.sva.lfb    := address;
   127:C        2     HPIBshort_msge_out(uep, command_sec, addr(sarm), sizeof(sarm));
   128:C        2     HPIBwait_for_ppol(uep);
   129:C        2     set_address_and_return_mode := qstat(uep);
   130:C        2   end; {set_address_and_return_mode}
   131:S
   132:S
   133:D        1 function write_file_mark(uep: uep_type): unsgn8;
   134:S            {
   135:S              issue the following command sequence:
   136:S                .  SET_UNIT
   137:S                .  SET_VOLUME
   138:S                .  WRITE_FILE_MARK
   139:S              return the QSTAT byte
   140:D        2   }
   141:D        2   var
   142:D        2     wfm: {the 3 bytes in the command message}
   143:D        2       packed record
   144:D        2         setunitvol: setunitvol_type;
   145:D        2         writefilemark: CMD_type;
   146:D    -4  2       end;
   147:C        2   begin {write_file_mark}
   148:C        2     wfm.setunitvol    := suv_CMD_pair(uep);
   149:C        2     wfm.writefilemark := CMDwrite_file_mark;
   150:C        2     HPIBshort_msge_out(uep, command_sec, addr(wfm), sizeof(wfm));
   151:C        2     HPIBwait_for_ppol(uep);
   152:C        2     write_file_mark := qstat(uep);
   153:C        2   end; {write_file_mark}
```



```
   154:D        1 $page$
   155:S
   156:D        1 function unload(uep: uep_type): unsgn8;
   157:S            {
   158:S              issue the following command sequence:
   159:S                .  SET_UNIT
   160:S                .  SET_VOLUME
   161:S                .  UNLOAD
   162:S              return the QSTAT byte
   163:D        2   }
   164:D        2   var
   165:D        2     unl: {the 3 bytes in the command message}
   166:D        2       packed record
   167:D        2         setunitvol: setunitvol_type;
   168:D        2         unloadcmd: CMD_type;
   169:D    -4  2       end;
   170:C        2   begin {unload}
   171:C        2     unl.setunitvol := suv_CMD_pair(uep);
   172:C        2     unl.unloadcmd  := CMDunload;
   173:C        2     HPIBshort_msge_out(uep, command_sec, addr(unl), sizeof(unl));
   174:C        2     HPIBwait_for_ppol(uep);
   175:C        2     unload := qstat(uep);
   176:C        2   end; {unload}
   177:S
   178:S
   179:D        1 function locate_and_verify(uep: uep_type; length: integer): unsgn8;
   180:S            {
   181:S              issue the following command sequence:
   182:S                .  SET_UNIT
   183:S                .  SET_VOLUME
   184:S                .  SET_LENGTH
   185:S                .  LOCATE_AND_VERIFY
   186:S              return the QSTAT byte
   187:D        2   }
   188:D        2   var
   189:D        2     lv: {the 9 bytes in the command message}
   190:D        2       packed record
   191:D        2         setunitvol: setunitvol_type;
   192:D        2         nop: CMD_type;
   193:D        2         setlen: CMD_type;
   194:D        2         len: integer;
   195:D        2         locateandverify: CMD_type;
   196:D   -10  2       end;
   197:C        2   begin {locate_and_verify}
   198:C        2     lv.setunitvol      := suv_CMD_pair(uep);
   199:C        2     lv.nop             := CMDno_op;
   200:C        2     lv.setlen          := CMDset_length;
   201:C        2     lv.len             := length;
   202:C        2     lv.locateandverify := CMDlocate_and_ver;
   203:C        2     HPIBshort_msge_out(uep, command_sec, addr(lv), sizeof(lv));
   204:C        2     HPIBwait_for_ppol(uep);
   205:C        2     locate_and_verify := qstat(uep);
   206:C        2   end; {locate_and_verify}
```

```
207:D        1 $page$
208:S
209:D        1 function volume_copy_data(source_uep: uep_type; source_blk_addr: integer;
210:D        2              destination_uep: uep_type; destination_blk_addr: integer): unsgn8;
211:S           {
212:S             issue the following command sequence:
213:S               .  SET_UNIT_15
214:S               .  SET_LENGTH
215:S               .  COPY_DATA
216:S             return the QSTAT byte
217:D        2    }
218:D        2    var
219:D        2      cd: {the 24 bytes in the command message}
220:D        2        packed record
221:D        2          setunit15: CMD_type;
222:D        2          setlen: CMD_type;
223:D        2          length: integer;
224:D        2          nop: CMD_type;
225:D        2          copydata: CMD_type;
226:D        2          s_volunit: evu_type;
227:D        2          s_setadd: CMD_type;
228:D        2          s_address: sva_type;
229:D        2          d_volunit: evu_type;
230:D        2          d_setadd: CMD_type;
231:D        2          d_address: sva_type;
232:D   -24  2        end;
233:C        2    begin {volume_copy_data}
234:C        2      cd.setunit15      := CMDset_unit_15;
235:C        2      cd.setlen         := CMDset_length;
236:C        2      cd.length         := -1;  {full volume}
237:C        2      cd.nop            := CMDno_op;
238:C        2      cd.copydata       := CMDcopy_data;
239:C        2      cd.s_volunit.vvvv := source_uep^.dv;
240:C        2      cd.s_volunit.uuuu := source_uep^.du;
241:C        2      cd.s_setadd       := CMDset_address_1v;
242:C        2      cd.s_address.utb  := 0;
243:C        2      cd.s_address.lfb  := source_blk_addr;
244:C        2      cd.d_volunit.vvvv := destination_uep^.dv;
245:C        2      cd.d_volunit.uuuu := destination_uep^.du;
246:C        2      cd.d_setadd       := CMDset_address_1v;
247:C        2      cd.d_address.utb  := 0;
248:C        2      cd.d_address.lfb  := destination_blk_addr;
249:C        2      HPIBshort_msge_out(source_uep, command_sec, addr(cd), sizeof(cd));
250:C        2      HPIBwait_for_ppol(source_uep);
251:C        2      volume_copy_data := qstat(source_uep);
252:C        2    end; {volume_copy_data}
```



```
253:D        1 $page$
254:S
255:D        1 function copy_start_address(uep: uep_type; var csa_bytes: sva_type): unsgn8;
256:S           {
257:S             issue the following command sequence:
258:S               .  SET_UNIT
259:S               .  SET_VOLUME
260:S               .  READ DRIVE TABLES: COPY START ADDRESS (7914 ONLY!!!)
261:S             return the QSTAT byte
262:D        2    }
263:D        2    var
264:D        2      csa: {the 5 bytes in the command message}
265:D        2        packed record
266:D        2          setunitvol: setunitvol_type;
267:D        2          initutil: CMD_type;
268:D        2          readdrivetables: unsgn8;
269:D        2          tablenumber: unsgn8;
270:D    -6  2        end;
271:D    -7  2      premature_eoi: boolean;
272:D   -10  2      qs: unsgn8;
273:C        2    begin {copy_start_address}
274:C        2      csa.setunitvol      := suv_CMD_pair(uep);
275:C        2      csa.initutil        := CMDinit_util_SEM;  {send execution message}
276:C        2      csa.readdrivetables := 196;
277:C        2      csa.tablenumber     := 12;
278:C        2      HPIBshort_msge_out(uep, command_sec, addr(csa), sizeof(csa));
279:C        2      HPIBwait_for_ppol(uep);
280:C        2      try
281:C        3        HPIBshort_msge_in(uep, execution_sec, addr(csa_bytes), sizeof(csa_bytes));
282:C        3        premature_eoi := false;
283:C        3      recover
284:C        3        with bip_type(uep^.dvrtemp)^ do  {confirm the "premature" eoi}
285:C        4          begin
286:C        4            if (escapecode<>-10) or (iores<>zbadhardware) then
287:C        5              escape(escapecode);
288:C        4            premature_eoi := true;
289:C        4            iores := inoerror;
290:C        4          end; {with}
291:C        2      HPIBwait_for_ppol(uep);
292:C        2      qs := qstat(uep);
293:C        2      if premature_eoi and (qs=0) then
294:C        3        ioresc_bkgnd(uep, zcatchall);
295:C        2      copy_start_address := qs;
296:C        2    end; {copy_start_address}
297:S
298:S
299:C        1 end; {CS80tbr}
```

```
300:D        1 $page$
301:S
302:D        1 module CS80tbdvr;  {tape backup driver}
303:S
304:D        1 import
305:D        1   sysglobals, misc, bkgnd, discHPIB, tapebuf, CS80, CS80dsr, CS80tbr;
306:S
307:D        1 export
308:D        1   type
309:D        1     uep_proc_type = procedure(uep: uep_type);
310:S
311:D        1   var  {"var parameter" set by Qdescribe}
312:D   -38  1     describe_bytes: describe_type;
313:D   -38  1   procedure Qdescribe(uep: uep_type);
314:S
315:D   -38  1   function Qdevicename(uep: uep_type; var saved_ioresult: integer): string80;
316:S
317:D   -38  1   var  {"parameters" for Qverify}
318:D   -42  1     verify_length: integer;
319:D   -46  1     verify_block_size: integer;
320:D   -46  1   procedure Qverify(uep: uep_type);
321:S
322:D   -46  1   procedure Qtapesetup(uep: uep_type);
323:D   -46  1   procedure Qconfigure(uep: uep_type);
324:D   -46  1   procedure Qwritefilemark(uep: uep_type);
325:D   -46  1   procedure Qunload(uep: uep_type);
326:S
327:D   -46  1   var  {"parameter" for Qcertify}
328:D   -48  1     initialize_options_byte: unsgn8;
329:D   -48  1   procedure Qcertify(uep: uep_type);
330:S
331:D   -48  1   var  {"parameters" for Qcontrollercopy}
332:D   -52  1     destination_uep: uep_type;
333:D   -56  1     source_block_address: integer;
334:D   -60  1     destination_block_address: integer;
335:D   -60  1   procedure Qcontrollercopy(source_uep: uep_type);
336:S
337:D   -60  1   var  {"var parameter" for Qcopy_start_address}
338:D   -66  1     csa_bytes: sva_type;
339:D   -66  1   procedure Qcopy_start_address(uep: uep_type);
340:S
341:S
342:D   -66  1 implement {CS80tbdvr}
```



```
343:D   -66  1 $page$
344:S
345:D        1 procedure with_bkgnd_do(uep_proc: uep_proc_type; uep: uep_type);
346:C        2   begin {with_bkgnd_do}
347:C        2     try
348:C        3       ioresult := ord(inoerror);
349:C        3       allocate_bkgnd_info(uep);
350:C        3       if HPIBamigo_identify(uep) div 256<>2 then
351:C        4         ioresc_bkgnd(uep, znodevice);
352:C        3       call(uep_proc, uep);
353:C        3       deallocate_bkgnd_info(uep);
354:C        3     recover
355:C        3       begin
356:C        3         abort_bkgnd_process(uep);
357:C        3         ioresult := uep^.dvrtemp;
358:C        3         uep^.dvrtemp := ord(inoerror);  {report the error only once}
359:C        3         escape(-10);
360:C        3       end; {recover}
361:C        2   end; {with_bkgnd_do}
362:S
363:S
364:D        1 procedure Qtapesetup(uep: uep_type);
365:S
366:D        2   procedure tapesetup(uep: uep_type);
367:D        3     var
368:D    -1  3       retry_required: boolean;
369:C        3     begin {tapesetup}
370:C        3       repeat
371:C        4         retry_required := false;
372:C        4         if set_unitvol(uep)<>0 then
373:C        5           handle_bad_status(uep, true, retry_required);
374:C        4       until not retry_required;
375:C        3       repeat  {enable auto skip sparing}
376:C        4         retry_required := false;
377:C        4         if set_options(uep, (options byte) 6)<>0 then
378:C        5           handle_bad_status(uep, true, retry_required);
379:C        4       until not retry_required;
380:C        3     end; {tapesetup}
381:S
382:C        2   begin {Qtapesetup}
383:C        2     with_bkgnd_do(tapesetup, uep);
384:C        2   end; {Qtapesetup}
385:S
386:S
387:D        1 procedure Qconfigure(uep: uep_type);
388:C        2   begin {Qconfigure}
389:C        2     with_bkgnd_do(configure, uep);
390:C        2   end; {Qconfigure}
```

```
391:D   -66  1 $page$
392:S
393:D        1 procedure Qwritefilemark(uep: uep_type);
394:S
395:D        2   procedure writefilemark(uep: uep_type);
396:D        3     var
397:D    -1  3       retry_required: boolean;
398:C        3     begin {writefilemark}
399:C        3       repeat
400:C        4         retry_required := false;
401:C        4         if write_file_mark(uep)<>0 then
402:C        5           handle_bad_status(uep, true, retry_required);
403:C        4       until not retry_required;
404:C        3     end; {writefilemark}
405:S
406:C        2   begin {Qwritefilemark}
407:C        2     with_bkgnd_do(writefilemark, uep);
408:C        2   end; {Qwritefilemark}
409:S
410:S
411:D        1 procedure Qcertify(uep: uep_type);
412:S
413:D        2   procedure certify(uep: uep_type);
414:D        3     var
415:D    -1  3       retry_required: boolean;
416:C        3     begin {certify}
417:C        3       repeat  {initialize the medium}
418:C        4         retry_required := false;
419:C        4         if initialize_media(uep, initialize_options_byte, {interleave_factor} 0)<>0 then
420:C        5           handle_bad_status(uep, true, retry_required);
421:C        4       until not retry_required;
422:C        3     end; {certify}
423:S
424:C        2   begin {Qcertify}
425:C        2     with_bkgnd_do(certify, uep);
426:C        2   end; {Qcertify}
427:S
428:S
429:D        1 procedure Qdescribe(uep: uep_type);
430:S
431:D        2   procedure do_describe(uep: uep_type);
432:D        3     var
433:D    -1  3       retry_required: boolean;
434:C        3     begin {do_describe}
435:C        3       repeat
436:C        4         retry_required := false;
437:C        4         if set_unitvol(uep)<>0 then
438:C        5           handle_bad_status(uep, true, retry_required);
439:C        4       until not retry_required;
440:C        3       repeat
441:C        4         retry_required := false;
442:C        4         if describe(uep, describe_bytes)<>0 then
443:C        5           handle_bad_status(uep, true, retry_required);
444:C        4       until not retry_required;
445:C        3     end; {do_describe}
446:S
447:C        2   begin {Qdescribe}
448:C        2     with_bkgnd_do(do_describe, uep);
449:C        2   end; {Qdescribe}
```



```
450:D   -66  1 $page$
451:S
452:D        1 function Qdevicename(uep: uep_type; var saved_ioresult: integer): string80;
453:S
454:D        2   var
455:D   -82  2     devicename: string80;
456:D   -86  2     index: integer;
457:D   -86  2     bcd_product_number: {within the describe bytes}
458:D   -86  2       packed record case integer of
459:D   -86  2         0: (dn:  unsgn24);
460:D   -86  2         1: (bcd: packed array[1..6] of unsgn4);
461:D   -90  2       end;
462:D   -94  2     product_number: integer;
463:D   -94  2     fixed_vol_byte:
464:D   -94  2       packed record case integer of
465:D   -94  2         0: (byte: unsgn8);
466:D   -94  2         1: (bit:  packed array[0..7] of boolean);
467:D   -98  2       end;
468:S
469:C        2   begin {Qdevicename}
470:S
471:C        2     try
472:C        3       Qdescribe(uep);
473:C        3     recover
474:C        3       if escapecode<>-10 then
475:C        4         escape(escapecode);
476:S
477:C        2     if saved_ioresult=ord(inoerror) then  {report any error with describe}
478:C        3       saved_ioresult := ioresult;  {can be inoerror}
479:S
480:C        2     if ioresult=ord(inoerror) then
481:C        3       with describe_bytes do
482:C        4         begin
483:C        4           bcd_product_number.dn := dn;
484:C        4           product_number := 0;
485:C        4           for index := 1 to 5 do
486:C        5             product_number := product_number*10+bcd_product_number.bcd[index];
487:C        4           devicename := 'HP';
488:C        4           strwrite(devicename, 3, index, product_number:1);
489:C        4           case dt of
490:C        5             0:        devicename := devicename+' fixed disc';
491:C        5             1:        begin
492:C        5                         fixed_vol_byte.byte := describe_bytes.fvb;
493:C        5                         if fixed_vol_byte.bit[7-uep^.dv]
494:C        6                           then devicename := devicename+' fixed disc'
495:C        6                           else devicename := devicename+' removeable disc';
496:C        5                       end;
497:C        5             2:        devicename := devicename+' tape';
498:C        5             otherwise {can't further describe the generic device type};
499:C        5           end; {case}
500:C        4         end {with}
501:C        4     else
502:C        3       devicename := '<inaccessible CS80 device>';
503:S
504:C        2     Qdevicename := devicename;
505:S
506:C        2   end; {Qdevicename}
```

```
 507:D   -66  1 $page$
 508:S
 509:D        1 procedure Qverify(uep: uep_type);
 510:S
 511:D        2   var
 512:D    -4  2     working_verify_length: integer;
 513:D    -5  2     bad_blocks_encountered: boolean;
 514:S
 515:D        2   procedure handle_verify_status(var retry_required: boolean);
 516:S
 517:D        3     var
 518:D    -2  3       iorval_to_report: iorsltwd;  {to hold the first reportable error}
 519:D    -4  3       working_iorval: iorsltwd;  {cleared each time status is read}
 520:D   -24  3       status_bytes: status_type;
 521:D   -28  3       eb_scan, parameter_field_owner: errorbit_type;
 522:D   -29  3       reconfiguration_needed: boolean;
 523:S
 524:C        3     begin {handle_verify_status}
 525:S
 526:C        3       iorval_to_report := inoerror;
 527:S
 528:C        3       repeat
 529:S
 530:C        4         if status(uep, status_bytes)<>0 then
 531:C        5           ioresc_bkgnd(uep, zbadhardware);
 532:S
 533:C        4         working_iorval := inoerror;
 534:C        4         parameter_field_owner := channel_parity_error;  {doesn't really own it!}
 535:C        4         reconfiguration_needed := false;
 536:S
 537:C        4         for eb_scan := eb63 downto eb0 do
 538:C        5           if status_bytes.errorbits[eb_scan] then
 539:C        6             begin
 540:S
 541:C        6               if eb_scan in errorbits_owning_parmfield then
 542:C        7                 parameter_field_owner := eb_scan;
 543:S
 544:C        6               case eb_scan of
 545:S
 546:C        7                 {specific fatal errors}
 547:C        7                   channel_parity_error,
 548:C        7                   controller_fault,
 549:C        7                   unit_fault,
 550:C        7                   diagnostic_result:
 551:C        7                       working_iorval := zbadhardware;
 552:C        7                   illegal_opcode,
 553:C        7                   parameter_bounds,
 554:C        7                   illegal_parameter:
 555:C        7                       working_iorval := zbadmode;  {some cmds optional in SS/80}
 556:C        7                   module_addressing:
 557:C        7                       working_iorval := znodevice;
 558:C        7                   address_bounds,
 559:C        7                   end_of_volume:
 560:C        7                       working_iorval := znosuchblk;
 561:C        7                   uninitialized_media:
 562:C        7                       if status_bytes.errorbits[power_fail]
 563:C        8                         then {probably an uncertified tape; allow access anyway}
 564:C        8                         else working_iorval := zuninitialized;
 565:C        7                   no_spares_available:
 566:C        7                       working_iorval := zinitfail;
```



```
 567:C        7                   not_ready:
 568:C        7                       working_iorval := znotready;
 569:C        7                   write_protect:
 570:C        7                       working_iorval := zprotected;
 571:C        7                   no_data_found,
 572:C        7                   end_of_file:
 573:C        7                       working_iorval := znoblock;
 574:C        7                   unrecoverable_data_overflow,
 575:C        7                   unrecoverable_data:
 576:C        7                       working_iorval := zbadblock;
 577:S
 578:C        7                 {power fail}
 579:C        7                   power_fail:
 580:C        7                       begin
 581:C        7                         uep^.umediavalid := false;
 582:C        7                         if uep=tapebuf_uep then
 583:C        8                           tapebuf_state := undefined;
 584:C        7                         if uep^.ureportchange then
 585:C        8                           working_iorval := zmediumchanged;
 586:C        7                         reconfiguration_needed := true;
 587:C        7                         retry_required := true;
 588:C        7                       end;
 589:S
 590:C        7                 {retryable errors}
 591:C        7                   operator_release_required,
 592:C        7                   diagnostic_release_required,
 593:C        7                   internal_maintenance_required,
 594:C        7                   retransmit:
 595:C        7                       retry_required := true;
 596:S
 597:C        7                 {errors indicating release requested}
 598:C        7                   operator_request,
 599:C        7                   diagnostic_request,
 600:C        7                   internal_maintenance_request:
 601:C        7                       {do nothing here; release below if parmeter field owned};
 602:S
 603:C        7                 {errors indicating reconfiguration needed}
 604:C        7                   media_wear,                       {supposed to be masked out}
 605:C        7                   latency_induced,                  {supposed to be masked out}
 606:C        7                   eb53,                             {supposed to be masked out}
 607:C        7                   eb54,                             {supposed to be masked out}
 608:C        7                   auto_sparing_invoked,             {supposed to be masked out}
 609:C        7                   eb56,                             {supposed to be masked out}
 610:C        7                   recoverable_data_overflow,        {supposed to be masked out}
 611:C        7                   marginal_data,                    {supposed to be masked out}
 612:C        7                   recoverable_data,                 {supposed to be masked out}
 613:C        7                   eb60,                             {supposed to be masked out}
 614:C        7                   maintenance_track_overflow,       {supposed to be masked out}
 615:C        7                   eb62:                             {supposed to be masked out}
 616:C        7                   eb63:                             {supposed to be masked out}
 617:C        7                     reconfiguration_needed := true;
 618:S
 619:C        7                 {errors not covered by the above cases}
 620:C        7                   otherwise
 621:S                                ( specifically including:
 622:S                                  message_sequence,
 623:S                                  message_length,
 624:S                                  cross_unit,
 625:C        7                         illegal_parallel_operation )
 626:C        7                       working_iorval := zcatchall;
```

```
627:S
628:C        7              end; {case}
629:S
630:C        6            end; {if}
631:S
632:C        4         if iorval_to_report=inoerror then  {none previously found; report this one}
633:C        5           iorval_to_report := working_iorval;  {it can be inoerror also!}
634:S
635:C        4         if parameter_field_owner=unrecoverable_data then
636:C        5           begin
637:C        5             writeln('  bad block at ',status_bytes.bba.lfb:1);
638:C        5             bad_blocks_encountered := true;
639:C        5             working_verify_length := verify_length-(status_bytes.bba.lfb+1)*verify_block_size;
640:C        5             if working_verify_length>0 then
641:C        6               retry_required := true;
642:C        5           end; {if}
643:S
644:C        4         if parameter_field_owner in errorbits_requesting_release then
645:C        5           if not (status_bytes.urr[1] in [0..15]) then
646:C        6             ioresc_bkgnd(uep, zcatchall)
647:C        6           else if release(uep, status_bytes.urr[1])<>0 then
648:C        7             {handle the bad qstat elsewhere; worry not, the device won't forget!};
649:S
650:C        4         if reconfiguration_needed then
651:C        5           configure(uep);
652:S
653:C        4       until set_unit(uep, status_bytes.current_vu.uuuu)=0;  {restore original command unit}
654:S
655:C        3       if not (iorval_to_report in [inoerror, zbadblock]) then
656:C        4         ioresc_bkgnd(uep, iorval_to_report);
657:S
658:C        3     end; {handle_verify_status}
```



```
659:D    -5  2 $page$
660:S
661:D        2   procedure verify(uep: uep_type);
662:S
663:D        3     var
664:D    -1  3       retry_required: boolean;
665:S
666:C        3     begin {verify}
667:S
668:C        3       repeat  {set address to zero and set return addressing mode}
669:C        4         retry_required := false;
670:C        4         if set_address_and_return_mode(uep, {address} 0)<>0 then
671:C        5           handle_bad_status(uep, true, retry_required);
672:C        4       until not retry_required;
673:S
674:C        3       working_verify_length := verify_length;
675:C        3       bad_blocks_encountered := false;
676:S
677:C        3       repeat  {verify the requested amount}
678:C        4         retry_required := false;
679:C        4         if locate_and_verify(uep, working_verify_length)<>0 then
680:C        5           handle_verify_status(retry_required);
681:C        4       until not retry_required;
682:S
683:C        3       if bad_blocks_encountered then
684:C        4         ioresc_bkgnd(uep, zbadblock);
685:S
686:C        3     end; {verify}
687:S
688:C        2   begin {Qverify}
689:C        2     with_bkgnd_do(verify, uep);
690:C        2   end; {Qverify}
```

```
691:D   -66  1 $page$
692:S
693:D        1 procedure Qcontrollercopy(source_uep: uep_type);
694:S
695:D        2   procedure handle_copydata_status(var retry_required: boolean);
696:S
697:D        3     var
698:D    -2  3       iorval_to_report: iorsltwd;  {to hold the first reportable error}
699:D    -4  3       working_iorval: iorsltwd;  {cleared each time status is read}
700:D   -24  3       status_bytes: status_type;
701:D   -28  3       eb_scan, parameter_field_owner: errorbit_type;
702:D   -29  3       reconfiguration_needed: boolean;
703:D   -30  3       uee_pending: boolean;
704:D   -32  3       index: signed16;
705:S
706:D        3     procedure handle_uee_pending(uee_byte: signed8);
707:D        4       var
708:D        4         evu: {encoded volume/unit byte}
709:D        4           packed record case integer of
710:D        4             0: (sgn8: signed8);
711:D        4             1: (vu: evu_type);
712:D    -2  4           end;
713:D    -6  4         vol, unit: unsgn4;
714:D    -7  4         retry_required: boolean;
715:D   -88  4         message: string80;
716:C        4       begin {handle_uee_pending}
717:C        4         evu.sgn8 := uee_byte;
718:C        4         vol := evu.vu.vvvv;
719:C        4         unit := evu.vu.uuuu;
720:C        4         try
721:C        5           if unit=15 then
722:C        6             repeat
723:C        7               retry_required := false;
724:C        7               if set_unit(source_uep, unit)<>0 then
725:C        8                 handle_bad_status(source_uep, false, retry_required);
726:C        7             until not retry_required
727:C        7           else
728:C        6             repeat
729:C        7               retry_required := false;
730:C        7               if set_specified_unitvol(source_uep, unit, vol)<>0 then
731:C        8                 handle_bad_status(source_uep, false, retry_required);
732:C        7             until not retry_required;
733:C        5         recover
734:C        5           if escapecode<>-10 then escape(escapecode);
735:C        4         with bip_type(source_uep^.dvrtemp)^ do
736:C        5           if iores<>inoerror then  {we need to report it}
737:C        6             begin
738:C        6               if iorval_to_report=inoerror then  {none previously found...}
739:C        7                 iorval_to_report := iores;  {report this one later in a fatal way!}
740:C        6               if (vol=source_uep^.dv) and (unit=source_uep^.du) then
741:C        7                 writeln('  source unit error:')
742:C        7               else if (vol=destination_uep^.dv) and (unit=destination_uep^.du) then
743:C        8                 writeln('  destination unit error:')
744:C        8               else
745:C        8                 writeln('  unit: ', unit:1,', volume: ', vol:1, ' error:');
746:C        6               getioerrmsg(message, ord(iores));
747:C        6               writeln('     ',message);
748:C        6               iores := inoerror;  {now clear it out}
749:C        6             end;  {if}
750:C        4       end; {handle_uee_pending}
```



```
751:S
752:C        3     begin {handle_copydata_status}
753:S
754:C        3       iorval_to_report := inoerror;
755:S
756:C        3       repeat
757:S
758:C        4         if status(source_uep, status_bytes)<>0 then
759:C        5           ioresc_bkgnd(source_uep, zbadhardware);
760:S
761:C        4         working_iorval := inoerror;
762:C        4         parameter_field_owner := channel_parity_error;  {doesn't really own it!}
763:C        4         reconfiguration_needed := false;
764:S
765:C        4         for eb_scan := eb63 downto eb0 do
766:C        5           if status_bytes.errorbits[eb_scan] then
767:C        6             begin
768:S
769:C        6               if eb_scan in errorbits_owning_parmfield then
770:C        7                 parameter_field_owner := eb_scan;
771:S
772:C        6               case eb_scan of
773:S
774:C        7                 {the expected error}
775:C        7                   cross_unit:
776:C        7                       {do nothing here; handle below if parmeter field owned};
777:S
778:C        7                 {specific fatal errors}
779:C        7                   channel_parity_error,
780:C        7                   controller_fault,
781:C        7                   unit_fault,
782:C        7                   diagnostic_result:
783:C        7                       working_iorval := zbadhardware;
784:C        7                   illegal_opcode,
785:C        7                   parameter_bounds,
786:C        7                   illegal_parameter:
787:C        7                       working_iorval := zbadmode;  {some cmds optional in SS/80}
788:C        7                   module_addressing:
789:C        7                       working_iorval := znodevice;
790:C        7                   address_bounds,
791:C        7                   end_of_volume:
792:C        7                       working_iorval := znosuchblk;
793:C        7                   uninitialized_media:
794:C        7                       if status_bytes.errorbits[power_fail]
795:C        8                         then {probably an uncertified tape; allow access anyway}
796:C        8                         else working_iorval := zuninitialized;
797:C        7                   no_spares_available:
798:C        7                       working_iorval := zinitfail;
799:C        7                   not_ready:
800:C        7                       working_iorval := znotready;
801:C        7                   write_protect:
802:C        7                       working_iorval := zprotected;
803:C        7                   no_data_found,
804:C        7                   end_of_file:
805:C        7                       working_iorval := znoblock;
806:C        7                   unrecoverable_data_overflow,
807:C        7                   unrecoverable_data:
808:C        7                       working_iorval := zbadblock;
809:S
810:C        7                 {power fail}
```

```
811:C        7                    power_fail:
812:C        7                        begin
813:C        7                          source_uep^.umediavalid := false;
814:C        7                          if source_uep=tapebuf_uep then
815:C        8                            tapebuf_state := undefined;
816:C        7                          if source_uep^.ureportchange then
817:C        8                            working_iorval := zmediumchanged;
818:C        7                          reconfiguration_needed := true;
819:C        7                          retry_required := true;
820:C        7                        end;
821:S
822:C        7                  {retryable errors}
823:C        7                    operator_release_required,
824:C        7                    diagnostic_release_required,
825:C        7                    internal_maintenance_required,
826:C        7                    retransmit:
827:C        7                        retry_required := true;
828:S
829:C        7                  {errors indicating release requested}
830:C        7                    operator_request,
831:C        7                    diagnostic_request,
832:C        7                    internal_maintenance_request:
833:C        7                        {do nothing here; release below if parmeter field owned};
834:S
835:C        7                  {errors indicating reconfiguration needed}
836:C        7                    media_wear,                        {supposed to be masked out}
837:C        7                    latency_induced,                   {supposed to be masked out}
838:C        7                    eb53,                              {supposed to be masked out}
839:C        7                    eb54,                              {supposed to be masked out}
840:C        7                    auto_sparing_invoked,              {supposed to be masked out}
841:C        7                    eb56,                              {supposed to be masked out}
842:C        7                    recoverable_data_overflow,         {supposed to be masked out}
843:C        7                    marginal_data,                     {supposed to be masked out}
844:C        7                    recoverable_data,                  {supposed to be masked out}
845:C        7                    eb60,                              {supposed to be masked out}
846:C        7                    maintenance_track_overflow,        {supposed to be masked out}
847:C        7                    eb62,                              {supposed to be masked out}
848:C        7                    eb63:                              {supposed to be masked out}
849:C        7                        reconfiguration_needed := true;
850:S
851:C        7                  {errors not covered by the above cases}
852:C        7                    otherwise
853:S                                   ( specifically including:
854:S                                     message_sequence,
855:S                                     message_length,
856:C        7                            illegal_parallel_operation )
857:C        7                        working_iorval := zcatchall;
858:S
859:C        7                end; (case)
860:S
861:C        6              end; (if)
862:S
863:C        4          if iorval_to_report=inoerror then  {none previously found; report this one)
864:C        5            iorval_to_report := working_iorval;  {it can be inoerror also!)
865:S
866:C        4          index := 0;
867:C        4          if parameter_field_owner=cross_unit then
868:C        5            repeat
869:C        6              index := index+1;
870:C        6              if index<=6
```



```
871:C        7                then uee_pending := status_bytes.uee[index]>=0
872:C        7                else uee_pending := false;
873:C        6              if uee_pending then
874:C        7                handle_uee_pending(status_bytes.uee[index]);
875:C        6            until not uee_pending;
876:S
877:C        4          if parameter_field_owner in errorbits_requesting_release then
878:C        5            if not (status_bytes.urr[1] in [0..15]) then
879:C        6              ioresc_bkgnd(source_uep, zcatchall)
880:C        6            else if release(source_uep, status_bytes.urr[1])<>0 then
881:C        7              {handle the bad qstat elsewhere; worry not, the device won't forget!);
882:S
883:C        4          if reconfiguration_needed then
884:C        5            configure(source_uep);
885:S
886:C        4        until set_unit(source_uep, status_bytes.current_vu.uuuu)=0;  {restore original command unit)
887:S
888:C        3        if iorval_to_report<>inoerror then
889:C        4          ioresc_bkgnd(source_uep, iorval_to_report);
890:S
891:C        3      end; {handle_copydata_status)
892:S
893:S
894:D        2    procedure controllercopy(uep: uep_type);
895:D        3      var
896:D    -1  3        retry_required: boolean;
897:C        3      begin {controllercopy)
898:C        3        repeat  {copy the entire volume)
899:C        4          retry_required := false;
900:C        4          if volume_copy_data(source_uep, source_block_address,
901:C        5                              destination_uep, destination_block_address)<>0 then
902:C        5            handle_copydata_status(retry_required);
903:C        4        until not retry_required;
904:C        3      end; {controllercopy)
905:S
906:C        2    begin {Qcontrollercopy)
907:C        2      with bkgnd_do(controllercopy, source_uep);
908:C        2    end; {Qcontrollercopy)
```

```
 909:D   -66  1 $page$
 910:S
 911:D        1 procedure Qunload(uep: uep_type);
 912:S
 913:D        2   procedure do_unload(uep: uep_type);
 914:D        3     var
 915:D    -1  3       retry_required: boolean;
 916:C        3     begin {do_unload}
 917:C        3       repeat
 918:C        4         retry_required := false;
 919:C        4         if unload(uep)<>0 then
 920:C        5           handle_bad_status(uep, true, retry_required);
 921:C        4       until not retry_required;
 922:C        3     end; {do_unload}
 923:S
 924:C        2   begin {Qunload}
 925:C        2     with_bkgnd_do(do_unload, uep);
 926:C        2   end; {Qunload}
 927:S
 928:S
 929:D        1 procedure Qcopy_start_address(uep: uep_type);
 930:S
 931:D        2   procedure do_copy_start_address(uep: uep_type);
 932:D        3     var
 933:D    -1  3       retry_required: boolean;
 934:C        3     begin {do_copy_start_address}
 935:C        3       repeat
 936:C        4         retry_required := false;
 937:C        4         if copy_start_address(uep, csa_bytes)<>0 then
 938:C        5           handle_bad_status(uep, true, retry_required);
 939:C        4       until not retry_required;
 940:C        3     end; {do_copy_start_address}
 941:S
 942:C        2   begin {Qcopy_start_address}
 943:C        2     with_bkgnd_do(do_copy_start_address, uep);
 944:C        2   end; {Qcopy_start_address}
 945:S
 946:S
 947:C        1 end; {CS80tbdvr}
```



```
 948:D        1 $page$
 949:S
 950:D        1 {program tapebkup;}
 951:S
 952:D        1 import
 953:D        1   sysglobals, misc, sysdevs, fs, bkgnd, tapebuf, CS80tbr, CS80tbdvr;
 954:S
 955:D        1 var
 956:D        1   keyboard: text;
 957:S
 958:D        1 type
 959:D        1   proc_type = procedure;
 960:S
 961:D        1 const  {for under_lock_do procedure calls}
 962:D        1   verbosely = true;
 963:D        1   silently = false;  {if desired, set true to debug}
 964:S
 965:S
 966:D        1 procedure upcchar(var ch : char);
 967:C        2   begin
 968:C        2     if (ch>='a') and (ch<='z') then ch := chr(ord(ch)-32);
 969:C        2   end;
 970:S
 971:S
 972:D   -82  1 function yes(prompt: string80): boolean;
 973:D   -82  2   var
 974:D   -83  2     answer: char;
 975:C        2   begin {yes}
 976:C        2     writeln;
 977:C        2     write(prompt, ' (Y/N) ');
 978:C        2     repeat
 979:C        3       read(keyboard, answer);
 980:C        3       upcchar(answer);
 981:C        3     until (answer='Y') or (answer='N');
 982:C        2     writeln(answer);
 983:C        2     yes := answer='Y';
 984:C        2   end; {yes}
 985:S
 986:S
 987:D   -82  1 procedure fatal_ioresult(message: string80; iores: integer);
 988:C        2   begin {fatal_ioresult}
 989:C        2     writeln(#7+message);
 990:C        2     getioerrmsg(message, iores);
 991:C        2     writeln('  ',message);
 992:C        2     escape(-1);
 993:C        2   end; {fatal_ioresult}
 994:S
 995:S
 996:D   -82  1 procedure fatal_message(message: string80);
 997:C        2   begin {fatal_message}
 998:C        2     writeln(#7+message);
 999:C        2     escape(-1);
1000:C        2   end; {fatal_message}
```

```
1001:D        1 $page$
1002:S
1003:D        1 function on_same_medium(lun1, lun2: unitnum): boolean;
1004:D        2   var
1005:D    -8  2     uep1, uep2: uep_type;
1006:C        2   begin {on_same_medium}
1007:C        2     uep1 := addr(unitable^[lun1]);
1008:C        2     uep2 := addr(unitable^[lun2]);
1009:C        2     on_same_medium := (uep1^.sc=uep2^.sc) and (uep1^.ba=uep2^.ba) and
1010:C        2                       (uep1^.du=uep2^.du) and (uep1^.dv=uep2^.dv) and
1011:C        2                       (uep1^.letter=uep2^.letter);
1012:C        2   end; {on_same_medium}
1013:S
1014:S
1015:D        1 function on_same_controller(uep1, uep2: uep_type): boolean;
1016:C        2   begin {on_same_controller}
1017:C        2     on_same_controller := (uep1^.sc=uep2^.sc) and (uep1^.ba=uep2^.ba);
1018:C        2   end; {on_same_controller}
1019:S
1020:S
1021:D        1 procedure enable_report_changes(uep: uep_type);
1022:C        2   begin {enable_report_changes}
1023:C        2     with uep^ do
1024:C        3       begin
1025:C        3         umediavalid := false;
1026:C        3         ureportchange := true;
1027:C        3       end; {with}
1028:C        2   end; {enable_report_changes}
1029:S
1030:S
1031:D        1 procedure disable_report_changes(uep: uep_type);
1032:C        2   begin {disable_report_changes}
1033:C        2     with uep^ do
1034:C        3       begin
1035:C        3         umediavalid := false;
1036:C        3         ureportchange := false;
1037:C        3       end; {with}
1038:C        2   end; {disable_report_changes}
1039:S
1040:S
1041:D        1 procedure user_confirmation;
1042:C        2   begin {user_confirmation}
1043:C        2     if not yes('Are you SURE you want to proceed?') then
1044:C        3       escape(-1);
1045:C        2   end; {user_confirmation}
```



```
1046:D        1 $page$
1047:S
1048:D   -82  1 procedure request_vol(prompt: string80; var lun: unitnum);
1049:S
1050:D   -82  2   const
1051:D   -82  2     nonabortive_ioresult_set = [ord(inoerror), ord(zbadblock), ord(znoblock),
1052:D   -82  2                                 ord(zuninitialized), ord(inodirectory)];
1053:D   -82  2   var
1054:D  -164  2     response: string80;
1055:D  -182  2     fvid: vid;
1056:D  -304  2     ftitle: fid;
1057:D  -308  2     fsegs: integer;
1058:D  -310  2     fkind: filekind;
1059:D  -311  2     scantitle_ok: boolean;
1060:D  -316  2     saved_ioresult: integer;
1061:D  -320  2     uep: uep_type;
1062:S
1063:C        2   begin {request_vol}
1064:S
1065:C        2     repeat
1066:C        3       write(prompt,cteol);
1067:C        3       readln(response);
1068:C        3       if (strlen(response)=0) or (strpos(response,#27)<>0) then escape(-1);
1069:C        3       scantitle_ok := scantitle(response, fvid, ftitle, fsegs, fkind);
1070:C        3       if scantitle_ok and (ftitle='') then
1071:C        4         begin
1072:C        4           lun := findvolume(fvid, true);
1073:C        4           saved_ioresult := ioresult;          {for decoding below}
1074:C        4           getioerrmsg(response, ord(inounit));  {for the case lun=0}
1075:C        4         end {then}
1076:C        4       else
1077:C        4         begin
1078:C        4           lun := 0;
1079:C        4           response := 'Illegal syntax: volume ID required';
1080:C        4         end; {else}
1081:C        3       if lun=0 then
1082:C        4         writeln('  ',response);
1083:C        3     until lun<>0;
1084:S
1085:C        2     uep := addr(unitable^[lun]);
1086:C        2     with uep^ do
1087:C        3       begin
1088:C        3         if not (letter in ['Q', 'K']) then
1089:C        4           fatal_message('Specified volume is not on a CS80 device');
1090:S
1091:C        3         write('Device: ', Qdevicename(uep, saved_ioresult));
1092:C        3         writeln(', ',100*sc+ba:1,', ',du:1,', ',dv:1);
1093:C        3       end; {with uep^}
1094:S
1095:C        2     write('Logical unit #',lun:1,' - ');
1096:C        2     if fvid[1]<>'#' then
1097:C        3       writeln('''',fvid,':''')
1098:C        3     else if saved_ioresult in nonabortive_ioresult_set then
1099:C        4       writeln('<no directory>')
1100:C        4     else
1101:C        4       fatal_ioresult('', saved_ioresult);
1102:S
1103:C        2   end; {request_vol}
```

```
1104:D        1 $page$
1105:S
1106:D        1 procedure other_volume_check(lun: unitnum);
1107:D        2   var
1108:D    -1  2     warning_issued: boolean;
1109:D    -4  2     flun: unitnum;
1110:D    -6  2     line: shortint;
1111:D   -18  2     CRTline, CRTcolumn, stopline: integer;
1112:D   -36  2     fvid: vid;
1113:D   -36  2   const
1114:D   -36  2     maxlines = 10;
1115:D   -36  2     fieldwidth = 16;
1116:C        2   begin {other_volume_check}
1117:S
1118:C        2     warning_issued := false;
1119:S
1120:C        2     for flun := 1 to maxunit do
1121:C        3       if on_same_medium(flun, lun) then
1122:C        4         with unitable^[flun] do
1123:C        5           begin
1124:C        5             if flun<>lun then
1125:C        6               begin
1126:C        6                 if not warning_issued then
1127:C        7                   begin
1128:C        7                     writeln('NOTICE: this will also affect:');
1129:C        7                     warning_issued := true;
1130:C        7                     line := 1;
1131:C        7                   end; {if}
1132:S
1133:C        6                 fgetxy(output,CRTcolumn, CRTline);
1134:C        6                 write(' #',flun:1);
1135:C        6                 fgotoxy(output,CRTcolumn+5, CRTline);
1136:C        6                 call(dam, fvid, flun, getvolumename);
1137:C        6                 if strlen(fvid)>0
1138:C        7                   then write(fvid,':')
1139:C        7                   else write('<no dir>');
1140:C        6                 if CRTcolumn=0
1141:C        7                   then writeln              (scrolls the screen if necessary)
1142:C        7                   else fgotoxy(output,CRTcolumn, CRTline+1);
1143:S
1144:C        6                 line := line+1;
1145:C        6                 if line>maxlines then
1146:C        7                   begin
1147:C        7                     fgetxy(output,CRTcolumn, stopline);
1148:C        7                     fgotoxy(output,CRTcolumn+fieldwidth, stopline-maxlines);
1149:C        7                     line := 1;
1150:C        7                   end; {if}
1151:S
1152:C        6               end; {if flun<>lun}
1153:C        5             umediavalid := false;
1154:C        5           end; {with}
1155:S
1156:C        2     if warning_issued and (CRTcolumn<>0) then
1157:C        3       fgotoxy(output,0, stopline);
1158:S
1159:C        2   end; {other_volume_check}
```



```
1160:D        1 $page$
1161:S
1162:D        1 procedure preserving_status_do(proc: proc_type);
1163:D        2   var
1164:D    -2  2     saved_escapecode: shortint;
1165:D    -6  2     saved_ioresult: integer;
1166:C        2   begin {preserving_status_do}
1167:C        2     saved_escapecode := escapecode;
1168:C        2     saved_ioresult := ioresult;
1169:C        2     call(proc);
1170:C        2     sysescapecode := saved_escapecode;
1171:C        2     ioresult := saved_ioresult;
1172:C        2   end; {preserving_status_do}
1173:S
1174:S
1175:D        1 procedure writeln_proc;
1176:C        2   begin {writeln_proc}
1177:C        2     writeln;
1178:C        2   end; {writeln_proc}
1179:S
1180:S
1181:D        1 procedure under_lock_do(verbose: boolean; uep_proc: uep_proc_type;
1182:D   -82  2                                           uep: uep_type; op_description: string80);
1183:C        2   begin {under_lock_do}
1184:S
1185:C        2     lockup;
1186:C        2     try
1187:C        3       if verbose then
1188:C        4         begin
1189:C        4           writeln;
1190:C        4           writeln(op_description, ' in progress');
1191:C        4         end; {if}
1192:C        3       call(uep_proc, uep);
1193:C        3       if verbose then
1194:C        4         writeln(op_description, ' completed');
1195:C        3       escape(0);  {to do the lockdown}
1196:C        3     recover
1197:C        3       begin
1198:C        3         preserving_status_do(lockdown);
1199:C        3         if escapecode=0 then
1200:C        4           {do nothing}
1201:C        4         else if escapecode=-10 then
1202:C        5           begin
1203:C        5             if not verbose then
1204:C        6               preserving_status_do(writeln_proc);
1205:C        5             fatal_ioresult(op_description+' errored:', ioresult);
1206:C        5           end
1207:C        5         else
1208:C        5           escape(escapecode);
1209:C        3       end; {recover}
1210:S
1211:C        2   end; {under_lock_do}
```

```
1212:D        1 $page$
1213:S
1214:D        1 procedure medium_copy;
1215:S
1216:D        2   var
1217:D     -4 2     source_lun, destination_lun: unitnum;
1218:D    -12 2     source_uep, tape_uep: uep_type;
1219:D    -14 2     source_is_a_tape, destination_is_a_tape: boolean;
1220:D    -16 2     source_is_a_7914, destination_is_a_7914: boolean;
1221:D    -24 2     source_block_size, destination_block_size: integer;
1222:D    -32 2     source_size, destination_size: integer;
1223:D    -33 2     verify_destination: boolean;
1224:D    -35 2     copy_attempt_completed, loop_condition: boolean;
1225:D    -38 2     pass: shortint;
1226:S
1227:D        2   procedure attempt_copy;
1228:S
1229:D        3     procedure cleanup;
1230:C        4       begin {cleanup}
1231:C        4         if destination_is_a_tape then  {restore its configuration}
1232:C        5           Qconfigure(destination_uep);
1233:C        4       end; {cleanup}
1234:S
1235:C        3     begin {attempt_copy}
1236:S
1237:C        3       if source_is_a_tape and destination_is_a_7914 then  {see where to restore}
1238:C        4         begin
1239:C        4           under_lock_do(silently, Qcopy_start_address, source_uep, 'Copy start address request');
1240:C        4           destination_block_address := csa_bytes.lfb;
1241:C        4         end; {if}
1242:S
1243:C        3       if destination_is_a_tape then  {ensure it's certified; update size afterward!!!}
1244:C        4         begin
1245:C        4           tapebuf_state := undefined;
1246:C        4           initialize_options_byte := 0;  {certify only if currently uncertified}
1247:C        4           under_lock_do(verbosely, Qcertify, destination_uep, 'Destination tape certification');
1248:C        4           under_lock_do(silently, Qdescribe, destination_uep, 'Destination tape describe request');
1249:C        4           with describe_bytes do
1250:C        5             destination_size := (maxsvadd.lfb+1)*nbpb;
1251:C        4         end; {if}
1252:S
1253:C        3       lockup;
1254:C        3       try
1255:C        4         if destination_is_a_tape then  {enable auto skip-sparing}
1256:C        5           Qtapesetup(destination_uep);
1257:S
1258:C        4         if source_is_a_7914 or destination_is_a_7914 then
1259:C        5           begin
1260:C        5             writeln;
1261:C        5             writeln('Copy parameters for 7914 save/restore -');
1262:C        5             writeln('  source starting block address: ', source_block_address:0);
1263:C        5             writeln('  destination starting block address: ', destination_block_address:0);
1264:C        5           end; {if}
1265:S
1266:C        4         under_lock_do(verbosely, Qcontrollercopy, source_uep, 'Medium copy');
1267:S
1268:C        4         if destination_is_a_tape and (destination_size>source_size) then
1269:C        5           under_lock_do(silently, Qwritefilemark, destination_uep, 'Destination tape filemark append')
;
1270:S
```



```
1271:C        4         escape(0);  {to do the cleanup}
1272:C        4       recover
1273:C        4         begin
1274:C        4           preserving_status_do(cleanup);
1275:C        4           preserving_status_do(lockdown);
1276:C        4           if escapecode<>0 then
1277:C        5             escape(escapecode);
1278:C        4         end; {recover}
1279:S
1280:C        3       if destination_is_a_tape and verify_destination then  {verify it}
1281:C        4         begin
1282:C        4           if destination_size>source_size
1283:C        5             then verify_length := source_size
1284:C        5             else verify_length := destination_size;
1285:C        4           verify_block_size := destination_block_size;
1286:C        4           under_lock_do(verbosely, Qverify, destination_uep, 'Destination tape verification');
1287:C        4         end; {if}
1288:S
1289:C        3     end; {attempt_copy}
1290:S
1291:C        2   begin {medium_copy}
1292:S
1293:C        2     writeln(clearscr);
1294:C        2     request_vol('Source medium for copy? ', source_lun);
1295:C        2     source_uep := addr(unitable^[source_lun]);
1296:C        2     enable_report_changes(source_uep);
1297:C        2     other_volume_check(source_lun);
1298:C        2     source_is_a_7914 := source_uep^.devid=7914;
1299:C        2     with describe_bytes do
1300:C        3       begin
1301:C        3         source_is_a_tape := dt=2;
1302:C        3         source_block_size := nbpb;
1303:C        3         source_size := (maxsvadd.lfb+1)*nbpb;
1304:C        3       end; {with}
1305:S
1306:C        2     writeln;
1307:C        2     request_vol('Destination medium for copy? ', destination_lun);
1308:C        2     destination_uep := addr(unitable^[destination_lun]);
1309:C        2     enable_report_changes(destination_uep);
1310:C        2     if on_same_medium(source_lun, destination_lun) then
1311:C        3       fatal_message('Specified volumes are on the same medium');
1312:C        2     if not on_same_controller(source_uep, destination_uep) then
1313:C        3       fatal_message('Source and destination not on the same controller');
1314:C        2     other_volume_check(destination_lun);
1315:C        2     destination_is_a_7914 := destination_uep^.devid=7914;
1316:C        2     with describe_bytes do
1317:C        3       begin
1318:C        3         destination_is_a_tape := dt=2;
1319:C        3         destination_block_size := nbpb;
1320:C        3         destination_size := (maxsvadd.lfb+1)*nbpb;
1321:C        3       end; {with}
1322:S
1323:C        2     if destination_is_a_tape and (destination_size<source_size) then
1324:C        3       if source_is_a_7914 then
1325:C        4         begin
1326:C        4           writeln;
1327:C        4           writeln('REMINDER: two long tapes are required for');
1328:C        4           writeln('  complete 7914 backup.');
1329:C        4         end {then}
1330:C        4       else
```

```
1331:C  4          begin
1332:C  4            writeln;
1333:C  4            writeln('NOTICE: the destination tape is too small for a');
1334:C  4            writeln('  complete source backup!');
1335:C  4          end; {else}
1336:S
1337:C  2      user_confirmation;
1338:S
1339:C  2      if destination_is_a_tape
1340:C  3        then verify_destination := yes('Verify destination tape after the copy?')
1341:C  3        else verify_destination := true;  {always verify discs}
1342:S
1343:C  2      pass := 1;
1344:C  2      source_block_address := 0;
1345:C  2      destination_block_address := 0;
1346:S
1347:C  2      repeat  {loop for swapping tapes}
1348:S
1349:C  3        repeat  {copy attempts}
1350:C  4          try
1351:C  5            attempt_copy;
1352:C  5            copy_attempt_completed := true;
1353:C  5          recover
1354:C  5            if escapecode<>-1 then
1355:C  6              escape(escapecode)
1356:C  6            else
1357:C  6              if yes('Retry the current copy segment?') then
1358:C  7                copy_attempt_completed := false
1359:C  7              else if yes('Abort the entire medium copy sequence?') then
1360:C  8                fatal_message('Medium copy sequence aborted')
1361:C  8              else
1362:C  8                begin
1363:C  8                  writeln('Medium copy sequence continued');
1364:C  8                  copy_attempt_completed := true;
1365:C  8                end;
1366:C  4        until copy_attempt_completed;
1367:S
1368:C  3        if source_is_a_tape then
1369:C  4          under_lock_do(verbosely, Qunload, source_uep, 'Source tape unload request');
1370:S
1371:C  3        if destination_is_a_tape then
1372:C  4          under_lock_do(verbosely, Qunload, destination_uep, 'Destination tape unload request');
1373:S
1374:C  3        loop_condition := (source_is_a_7914 or destination_is_a_7914) and (pass<2);
1375:S
1376:C  3        if loop_condition then  {prepare for the next pass through the loop}
1377:C  4          begin
1378:S
1379:C  4            if destination_is_a_tape then  {update the source disc parms}
1380:C  5              begin
1381:C  5                source_size := source_size - destination_size;
1382:C  5                source_block_address := source_block_address + destination_size div source_block_size;
1383:C  5              end; {if}
1384:S
1385:C  4            if source_is_a_tape
1386:C  5              then tape_uep := source_uep
1387:C  5              else tape_uep := destination_uep;
1388:S
1389:C  4            writeln;
1390:C  4            writeln('Waiting for a new tape to be loaded:');
```



```
1391:C  4            writeln('Press <stop> to abort');
1392:S
1393:C  4            disable_report_changes(tape_uep);
1394:S
1395:C  4            repeat
1396:C  5              try
1397:C  6                Qcopy_start_address(tape_uep);
1398:C  6              recover
1399:C  6                if escapecode<>-10 then
1400:C  7                  escape(escapecode)
1401:C  7                else if ioresult<>ord(znotready) then
1402:C  8                  fatal_ioresult('Error while awaiting new tape load:', ioresult)
1403:C  8            until ioresult<>ord(znotready);
1404:S
1405:C  4            enable_report_changes(tape_uep);
1406:S
1407:C  4            pass := pass+1;
1408:S
1409:C  4          end; {if}
1410:S
1411:C  3      until not loop_condition;
1412:S
1413:C  2      if not destination_is_a_tape then
1414:C  3        begin
1415:C  3          verify_length := destination_size;
1416:C  3          verify_block_size := destination_block_size;
1417:C  3          under_lock_do(verbosely, Qverify, destination_uep, 'Destination disc verification');
1418:C  3        end; {if}
1419:S
1420:C  2    end; {medium_copy}
```

```
1421:D        1 $page$
1422:S
1423:D        1 procedure verify;
1424:S
1425:D        2   var
1426:D    -2  2     lun: unitnum;
1427:D    -6  2     uep: uep_type;
1428:D    -7  2     auto_unload: boolean;
1429:S
1430:C        2   begin {verify}
1431:S
1432:C        2     writeln(clearscr);
1433:C        2     request_vol('Verify what medium? ', lun);
1434:C        2     uep := addr(unitable^[lun]);
1435:C        2     enable_report_changes(uep);
1436:C        2     other_volume_check(lun);
1437:S
1438:C        2     user_confirmation;
1439:S
1440:C        2     with describe_bytes do
1441:C        3       begin
1442:C        3         if dt=2
1443:C        4           then auto_unload := yes('Unload tape after verification?')
1444:C        4           else auto_unload := false;
1445:C        3         verify_length := (maxsvadd.lfb+1)*nbpb;
1446:C        3         verify_block_size := nbpb;
1447:C        3       end; {with}
1448:S
1449:C        2     under_lock_do(verbosely, Qverify, uep, 'Verification');
1450:S
1451:C        2     if auto_unload then
1452:C        3       under_lock_do(verbosely, Qunload, uep, 'Tape unload request');
1453:S
1454:C        2   end; {verify}
```



```
1455:D        1 $page$
1456:S
1457:D        1 procedure certify;
1458:S
1459:D        2   var
1460:D    -2  2     lun: unitnum;
1461:D    -6  2     uep: uep_type;
1462:D    -7  2     auto_unload: boolean;
1463:S
1464:C        2   begin {certify}
1465:S
1466:C        2     writeln(clearscr);
1467:C        2     request_vol('Certify what tape? ', lun);
1468:C        2     uep := addr(unitable^[lun]);
1469:C        2     enable_report_changes(uep);
1470:C        2     if describe_bytes.dt<>2 then
1471:C        3       fatal_message('Specified volume is not on a tape');
1472:C        2     other_volume_check(lun);
1473:S
1474:C        2     user_confirmation;
1475:S
1476:C        2     if yes('Re-certify if already certified?')
1477:C        3       then initialize_options_byte := 1   {force complete certification}
1478:C        3       else initialize_options_byte := 0;  {certify only if currently uncertified}
1479:S
1480:C        2     auto_unload := yes('Unload tape after certification?');
1481:S
1482:C        2     tapebuf_state := undefined;
1483:C        2     under_lock_do(verbosely, Qcertify, uep, 'Tape certification');
1484:S
1485:C        2     if auto_unload then
1486:C        3       under_lock_do(verbosely, Qunload, uep, 'Tape unload request');
1487:S
1488:C        2   end; {certify}
```

```
1489:D        1 $page$
1490:S
1491:D        1 procedure commandlevel;
1492:S
1493:D        2   var
1494:D    -1  2     ch: char;
1495:S
1496:C        2   begin {commandlevel}
1497:S
1498:C        2     repeat
1499:S
1500:C        3       try
1501:C        4         write(homechar,
1502:C        4               'Tapebkup: Medium-copy Verify Certify-tape Quit  ?',
1503:C        4               cteol);
1504:C        4         read(keyboard, ch);
1505:C        4         upcchar(ch);
1506:C        4         writeln;
1507:C        4         case ch of
1508:C        5           'C': certify;
1509:C        5           'M': medium_copy;
1510:C        5           'Q': {quit};
1511:C        5           'V': verify;
1512:C        5           otherwise
1513:C        5             write(clearscr);
1514:C        5         end; {case}
1515:S
1516:C        4       recover
1517:C        4         if escapecode=-20 then
1518:C        5           ch := 'Q'  {stop key means "quit"}
1519:C        5         else if escapecode=-1 then
1520:C        6           {do nothing; remain in command level}
1521:C        6         else
1522:C        6           escape(escapecode);
1523:S
1524:C        3     until ch='Q';
1525:S
1526:C        2   end; {commandlevel}
1527:S
1528:S
1529:S
1530:C        1 begin {tapebkup}
1531:C        1   writeln(clearscr);
1532:C        1   writeln('              [Version 3.0]');
1533:C        1   writeln;
1534:C        1   writeln;
1535:C        1   writeln;
1536:C        1   writeln('  Copyright 1984 Hewlett-Packard Company');
1537:C        1   writeln('       All rights are reserved.');
1538:C        1   commandlevel;
1539:C        1 end. {tapebkup}
1540:S
1541:S

No errors. No warnings.
             ***** Nonstandard language features enabled *****
```

TYPES

Description

TYPES contains declarations used by the DGL graphics routines.

```
Pascal [Rev 3.0M  6/ 4/84] TYPES.TEXT              26-Dec-84 22:38:26 Page 1   *** File name: TYPES ***

     1:S
     2:D         0 {                                                                          }
     3:D         0 { Pascal work station graphics library                                     }
     4:D         0 {                                                                          }
     5:D         0 { Module    = DGL_TYPES                                                    }
     6:D         0 { Programer = BJS                                                          }
     7:D         0 { Date      = 2/1/82                                                       }
     8:S
     9:S           { Purpose:  Holds declarations which must be imported by graphics library
    10:D         0              users when interfacing with some library routines.           }
    11:S
    12:D         0 { Rev history                                                              }
    13:D         0 {   6-15-82  BJS Added data type gchar_list                                }
    14:S
    15:S           {     (c) Copyright Hewlett-Packard Company, 1983.
    16:S                 All rights are reserved.  Copying or other
    17:S                 reproduction of this program except for archival
    18:S                 purposes is prohibited without the prior
    19:S                 written consent of Hewlett-Packard Company.
    20:S
    21:S
    22:S                             RESTRICTED RIGHTS LEGEND
    23:S
    24:S                 Use, duplication, or disclosure by the Government
    25:S                 is subject to restrictions as set forth in
    26:S                 paragraph (b) (3) (B) of the Rights in Technical
    27:S                 Data and Computer Software clause in
    28:S                 DAR 7-104.9(a).
    29:S
    30:S                 HEWLETT-PACKARD COMPANY
    31:D         0       Fort Collins, Colorado                                 }
    32:D         0
    33:D         0 $TABLES$
    34:D         0 $modcal$
    35:D         0 $include 'OPTIONS'$
    36:S
    37:D           { This include file specifies range checking, debug and other compiler
    38:D         0   options for the graphics library                                        }
    39:S
    40:D         0 $debug OFF$
    41:D         0 $range OFF$
    42:D         0 $copyright 'COPYRIGHT 1984 BY HEWLETT-PACKARD COMPANY'$
    43:D         0 $FLOAT_HDW TEST$
    44:S
    45:S
    46:S
    47:D         0 $include 'OPTIONS'$
  1000:D         0 $linenum 1000$
  1001:S
  1002:D         0 module DGL_TYPES;
  1003:S
  1004:D         1 export
  1005:S
  1006:D         1 type
  1007:D         1   gbyte = -128 .. 127;
  1008:D         1   gshortint = -32768 .. 32767;
  1009:D         1   gstring255 = string[255];
  1010:S
  1011:D         1   gshortint_list  = array [1..maxint] of gshortint;
```

```
Pascal [Rev 3.0M  6/ 4/84] TYPES.TEXT              26-Dec-84 22:38:26 Page 2   *** File name: TYPES ***

  1012:D         1   gint_list  = array [1..maxint] of integer;
  1013:D         1   greal_list = array [1..maxint] of real;
  1014:D         1   gchar_list = packed array [1..maxint] of char;
  1015:S
  1016:D         1 implement
  1017:S

Dump of DGL_TYPES
Imported:

Exported:
 DGL_TYPES        module lev=1
 GBYTE            type
  subrange min=-128 max=127 unpacksize=2 align=2 bitsize=8 signed
 GCHAR_LIST       type
  array elbitsize=8 unpacksize=2147483647 align=2
 GINT_LIST        type
  array elsize=4 unpacksize=4 (OFLO) align=2
 GREAL_LIST       type
  array elsize=8 unpacksize=4 (OFLO) align=2
 GSHORTINT        type
  subrange min=-32768 max=32767 unpacksize=2 align=2 bitsize=16 signed
 GSHORTINT_LIST   type
  array elsize=2 unpacksize=4 (OFLO) align=2
 GSTRING255       type
  array elbitsize=8 unpacksize=256 align=2


DGL_TYPES dump complete

  1018:D         1 end.  { DGL_TYPES }
  1019:D         1 $LIST ON$

No errors. No warnings.
              ***** Nonstandard language features enabled *****
```

Part No. 98615-90074
E 0285
Microfiche No. 98615-99074

Printed in U.S.A.
Edition 1
February 1985