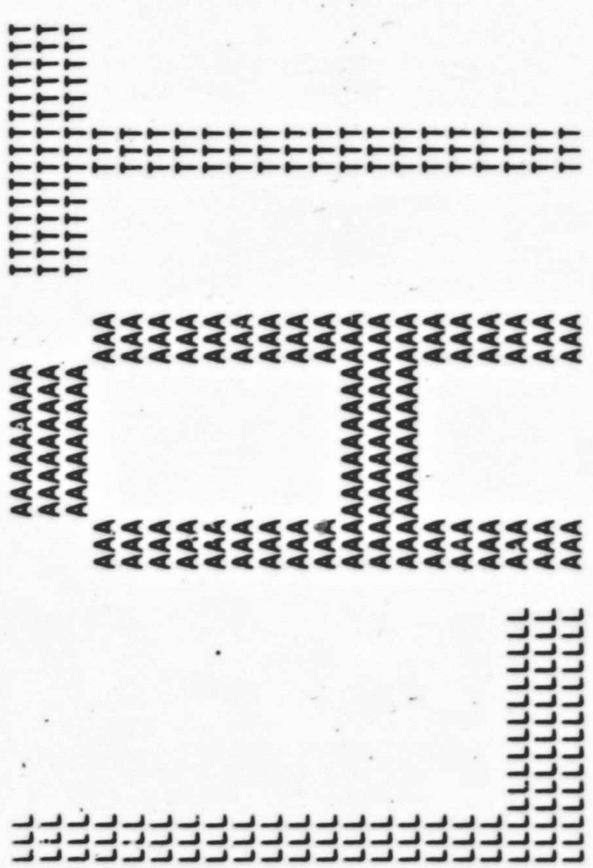
N 12



```
LL          TTTTTTTTTT  DDDDDDDD    RRRRRRRR     IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
LL          TTTTTTTTTT  DDDDDDDD    RRRRRRRR     IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
LL              TT      DD      DD  RR      RR     II      VV      VV  EE          RR      RR
LL              TT      DD      DD  RR      RR     II      VV      VV  EE          RR      RR
LL              TT      DD      DD  RR      RR     II      VV      VV  EE          RR      RR
LL              TT      DD      DD  RR      RR     II      VV      VV  EE          RR      RR
LL              TT      DD      DD  RRRRRRRR       II      VV      VV  EEEEEEEE    RRRRRRRR
LL              TT      DD      DD  RRRRRRRR       II      VV      VV  EEEEEEEE    RRRRRRRR
LL              TT      DD      DD  RR   RR        II      VV      VV  EE          RR   RR
LL              TT      DD      DD  RR   RR        II      VV      VV  EE          RR   RR
LL              TT      DD      DD  RR    RR       II        VV  VV    EE          RR    RR        ....
LL              TT      DD      DD  RR    RR       II        VV  VV    EE          RR    RR        ....
LLLLLLLLLL      TT      DDDDDDDD    RR      RR     IIIIII      VV      EEEEEEEEEE  RR      RR      ....
LLLLLLLLLL      TT      DDDDDDDD    RR      RR     IIIIII      VV      EEEEEEEEEE  RR      RR      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
                        .TITLE  LTDRIVER - Local Area Terminal Port Driver
;JAY0002                .IDENT  'V04-002'
-1
                        ;*****************************************************************************
                        ;*                                                                           *
                        ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
                        ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
                        ;*  ALL RIGHTS RESERVED.                                                     *
                        ;*                                                                           *
                        ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
                        ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
                        ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
                        ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                        ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
                        ;*  TRANSFERRED.                                                             *
                        ;*                                                                           *
                        ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
                        ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
                        ;*  CORPORATION.                                                             *
                        ;*                                                                           *
                        ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
                        ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        ;*                                                                           *
                        ;*****************************************************************************

                        ; Do not remove items from this list.  Add X when item is done.
                        ; to be done:
                        ;
                        ; - LATCP should be allowed to specify the Ethernet device.
                        ; - fix slot stop and circuit stop  code for correct message format
                        ;       including reason code and text.
                        ; - transmit two slots/message from host transmit loop.
                        ; - allocate and be able to send additional buffers based about the servers
                        ;       value of XTRABFRs (in START message).
                        ; - send credits in DATA_B slots.
                        ;
```

```
0000     39
0000     40 ;
0000     41 ;*****************************************************************************
0000     42 ;*                                                                           *
0000     43 ;*  Copyright (c) 1982, 1984                                                 *
0000     44 ;*  by digital equipment corporation, maynard, mass.                         *
0000     45 ;*                                                                           *
0000     46 ;*  this software is furnished under a license and may be used and  copied   *
0000     47 ;*  only  in  accordance  with  the  terms  of  such  license and with the   *
0000     48 ;*  inclusion of the above copyright notice.  this software or  any  other   *
0000     49 ;*  copies  thereof may not be provided or otherwise made available to any   *
0000     50 ;*  other person.  no title to and ownership of  the  software  is  hereby   *
0000     51 ;*  transferred.                                                             *
0000     52 ;*                                                                           *
0000     53 ;*  the information in this software is subject to change  without  notice   *
0000     54 ;*  and  should  not  be  construed  as  a commitment by digital equipment   *
0000     55 ;*  corporation.                                                             *
0000     56 ;*                                                                           *
0000     57 ;*  digital assumes no responsibility for the use or  reliability  of  its   *
0000     58 ;*  software on equipment which is not supplied by digital.                  *
0000     59 ;*                                                                           *
0000     60 ;*****************************************************************************
0000     61 ;
0000     62 ;++
0000     63 ; Facility:
0000     64 ;
0000     65 ;       VAX/VMS Terminal Driver
0000     66 ;
0000     67 ; Abstract:
0000     68 ;
0000     69 ;       Local Area Terminal Port Driver
0000     70 ;
0000     71 ;       This is an alternate port driver for use with the VMS terminal
0000     72 ;       driver.  It forms the host end of the Local Area Terminal (LAT)
0000     73 ;       protocol and communicates with a LAT concentrator to provide
0000     74 ;       terminal communications on a local area network.
0000     75 ;
0000     76 ;       This driver performs IO using a datalink driver via the FFI
0000     77 ;       mechanism.  Its connection with the datalink driver is controlled
0000     78 ;       by a program called LATCP.
0000     79 ;
0000     80 ;
0000     81 ;
0000     82 ; Authors:
0000     83 ;
0000     84 ;       Bruce Mann
0000     85 ;       Darrell Duffy
0000     86 ;       Joe Marchesani
0000     87 ;       Rich Bailey
0000     88 ;
0000     89 ; Revision history:
0000     90 ;
;JAY0002 0000     .1 ;       V04-002 JAY0002         John A. Ywoskus         17-Dec-1984
;JAY0002 0000     .2 ;               Add CPU dispatch table to determine max slots/circuit.
;JAY0002 0000     .3 ;               Fix odd length reject slot format.
;JAY0002 0000     .4 ;
;JAY0001 0000     .5 ;       V04-001 JAY0001         John A. Ywoskus         28-Sep-1984
```

```
;JAY0001                0000     .6 ;
;JAY0001                0000     .7 ;
;JAY0001                0000     .8 ;
;JAY0001                0000     .9 ;
-85                     0000    176 ;
                        0000    177 ;--
```

Add code to not set any flow control characters if terminal is set /PASSALL.
Add MicroVAX II to CPU RATING.
Mark terminals OFFLINE when called at DISCONNECT entry.

```
0000    179          .SBTTL  Global Definitions
0000    180
0000    181          .ENABLE SUPPRESSION
0000    182
0000    183 ;
0000    184 ;        Global definitions
0000    185 ;
0000    186
0000    187          $LATDEF                         ; LAT communication area defs
0000    188          $ACBDEF                         ; Ast control block
0000    189          $ADPDEF                         ; Define adapter
0000    190          $CRBDEF                         ; Define crb
0000    191          $CXBDEF                         ; Complex buffer
0000    192          $DDBDEF                         ; Define ddb
0000    193          $DEVDEF                         ; Define device characteristics
0000    194          $DPTDEF                         ; Define dpt
0000    195          $DYNDEF                         ; Dynamic memory block codes
0000    196          $FFIDEF                         ; Fast Interface block
0000    197          $FKBDEF                         ; Fork block definitions
0000    198          $IPLDEF                         ; IPL levels
0000    199          $MTXDEF                         ; Mutex offsets
0000    200          $ORBDEF                         ; Object's Rights Block offsets
0000    201          $PCBDEF                         ; Process control block
0000    202          $PHDDEF                         ; Process header block
0000    203          $PRDEF                          ; Processor register definitions
0000    204          $RSNDEF                         ; Resouce wait codes
0000    205          $TQEDEF                         ; Timer queue elements
0000    206          $UCBDEF                         ; Define ucb
0000    207          $VECDEF                         ; Define vector for crb
0000    208
0000    209          $TTYDEF                         ; Define terminal driver symbols
0000    210          $TTDEF                          ; Define terminal types
0000    211          $TT2DEF                         ; Define extended characteristics
0000    212          $TTYMACS                        ; Define terminal driver macros
0000    213          $TTYDEFS                        ; Define terminal driver symbols
0000    214
```

```
                        0000   216          .SBTTL  Local Definitions
                        0000   217 ;
                        0000   218 ;       Macros
                        0000   219 ;
                        0000   220          .MACRO  ALIGN_LONG
                        0000   221          .iif ne <.&3>,  .BLKB  <4-<.&3>> ; align to longword boundary, if needed
                        0000   222          .ENDM   ALIGN_LONG
                        0000   223
                        0000   224 ;
                        0000   225 ;       Conditional assembly parameters
                        0000   226 ;
                        0000   227
                        0000   228          LT_HISTORY      = 1             ; keep history of received/transitted ni fra
                        0000   229 ;;&&     LT_CRED_CHECK   = 1             ; validate credit total for UCBs
                        0000   230 ;;&&     LT_XMT_CHECK    = 1             ; validate transmit frame slots dynamically
                        0000   231
                        0000   232 ;
                        0000   233 ;       Constants defined for the VMS LAT host implementation
                        0000   234 ;
                        0000   235          TTY_C_BELL      = 7             ; Control_G (^G) ... bell character
                        0000   236          LAT$C_VMS       = 3             ; VAX/VMS product type code
                        0000   237          LAT$C_VMS_VER   = 4             ; Part of VAX/VMS V4.0
                        0000   238          LAT$C_IPL       = 8             ; Fork IPL to synchronize this driver
                        0000   239          LAT$C_PROGRESS  = 20            ; Must make progress every 20 msgs
                        0000   240          LAT$C_XMT_BUFFERS = 2           ; Number of transmit buffers
                        0000   241          LAT$C_MAX_RCRED = 1             ; Maximum number of credits extended
                        0000   242          LAT$C_UCB_BUFSIZ = 80           ; Size of UCB buffering
                        0000   243          LAT$C_UCB_SLOTSIZ = 4           ; Size of formatted slot buffer in UCB
                        0000   244          LAT$C_UCB_TIMOUT = 2            ; Allow 2 seconds before deleting  UCB's
                        0000   245          LAT$C_MAX_CSBS  = 32            ; Maximum number of circuits handled
                        0000   246          LAT$C_MAX_SLOTS = 64            ; Maximum slots in a message, this also
                        0000   247                                          ;  implies max connections per circuit
                        0000   248          LAT$C_HI_VER    = 5             ; Highest supported LAT protocol version
                        0000   249          LAT$C_LO_VER    = 5             ; Lowest supported LAT protocol version
                        0000   250          LAT$C_CUR_VER   = 5             ; Current LAT protocol version
                        0000   251          LAT$C_CUR_ECO   = 0             ; Current LAT protocol ECO version
                        0000   252          LAT$C_MAX_MSGSIZ = 1500         ; Maximum Data Link message size
                        0000   253          LAT$C_MAX_SLOTSIZ = 255         ; Maximum Slot size
                        0000   254          LAT$C_LOC_LEN   = 64            ; Maximum length of LOCation field
                        0000   255          LAT$C_GRP_LEN   = 32            ; Maximum number of group codes
                        0000   256          LAT$C_SVC_LEN   = 8             ; Maximum number of service classes
                        0000   257          LAT$C_MAX_SERVICES = 6          ; Maximum number of services
                        0000   258          LAT$C_MULTIADD1 = ^X0009        ; LAT Multicast address
                        0000   259          LAT$C_MULTIADD2 = ^X002B        ;       ...
                        0000   260          LAT$C_MULTIADD3 = ^X0F00        ;       ...
                        0000   261          LAT$C_HOST_TIMER = 2            ; Host circuit timeout value (in SECs)
                        0000   262          LAT$C_MULTI_TIMER = 10          ; Multicast message timer
                        0000   263          LAT$C_CIR_TIMER = 80            ; Host preferred circuit timer (in MS)
                        0000   264          LAT$C_SHUTDELAY = 5*1000*1000   ; Delay time = 1/2 second
                        0000   265          LAT$C_KEEP_CSB  = 10            ; Keep last 10 CSBs
                        0000   266
                        0000   267 ;
                        0000   268 ;       Port driver ucb extension to class driver structures
                        0000   269 ;
                        0000   270
                        0000   271          $DEFINI UCB     GLOBAL
                        0000   272          . = UCB$C_TT_LENGTH
```

```
0134    273             ALIGN_LONG                              ; align to longword boundary
0134    274 $DEF        UCB$L_LT_CSB     .BLKL   1              ; CSB address
0138    275 $DEF        UCB$L_LT_DEADLINK .BLKL  1              ; Link for dead ucbs
013C    276 $DEF        UCB$L_LT_INPBUF  .BLKL   1              ; Pointer to LAT type ahead buffer
0140    277 $DEF        UCB$B_LT_REMID   .BLKB   1              ; Remote slot index number
0141    278 $DEF        UCB$B_LT_LOCID   .BLKB   1              ; Local slot index number
0142    279 $DEF        UCB$B_LT_DATAW   .BLKB   1              ; Data waiting flags
0143    280  $VIELD     UCB,0,<-
0143    281             <LT_DATA,1,M>-                          ;* Output data waiting
0143    282             <LT_FLOW,1,M>-                          ;* Send current flow control values
0143    283             <LT_ABORT,1,M>-                         ;* Send abort request
0143    284             >
0143    285
0143    286 $DEF        UCB$B_LT_LATSTS .BLKB    1              ; Status for local terminal connection
0144    287  $VIELD     UCB,0,<-
0144    288             <LT_HANGUP,1,M>-                        ;* We are hanging up the slot
0144    289             <LT_DEAD,1,M>-                          ;* We have linked the UCB on deadlist
0144    290             <LT_OVFLOW,1,M>-                        ;* We had an overflow on input (XOFF)
0144    291             <LT_BOUND,1,M>-                         ;* Terminal is bound, don't delete UCB
0144    292             <LT_INPUT,1,M>-                         ;* Input stream in progress
0144    293             >
0144    294
0144    295 $DEF        UCB$B_LT_STATE   .BLKB   1              ; Slot state
0145    296  $EQU       UCB$C_LT_STATE_START     1              ;* State is starting
0145    297  $EQU       UCB$C_LT_STATE_RUN       2              ;* State is running
0145    298  $EQU       UCB$C_LT_STATE_STOP      3              ;* State is stopping (send stop slot)
0145    299  $EQU       UCB$C_LT_STATE_KILL      4              ;* Kill UCB immediately
0145    300
0145    301 $DEF        UCB$B_LT_REASON .BLKB    1              ; Reason for stopping slot
0146    302 $DEF        UCB$B_LT_TCRED   .BLKB   1              ; Available credits for this slot
0147    303 $DEF        UCB$B_LT_XCRED   .BLKB   1              ; Credits extended to concentrator
0148    304 $DEF        UCB$B_LT_RCRED   .BLKB   1              ; Credits to be returned to concentrator
0149    305 $DEF        UCB$B_LT_SLOTSZ .BLKB    1              ; Slot size (negotiated)
014A    306 $DEF        UCB$B_LT_MAXC    .BLKB   1              ; Size of UCB buffering
014B    307 $DEF        UCB$B_LT_CURC    .BLKB   1              ; Current character count
014C    308 $DEF        UCB$B_LT_CBUF    .BLKB LAT$C_UCB_BUFSIZ ; Startio data buffer
019C    309 $DEF        UCB$B_LT_EBUF                           ; Startio data buffer end
019C    310 $DEF        UCB$C_LT_LENGTH                         ; Length of block
019C    311             $DEFEND UCB
```

```
                     0000   313
                     0000   314 ;+
                     0000   315 ;       Circuit state block
                     0000   316 ;
                     0000   317 ;  For each concentrator, one virtual circuit is maintained.
                     0000   318 ;  Protocol state is maintained in circuit state blocks.
                     0000   319 ;  The CSB addresses are stored in a vector starting
                     0000   320 ;  at CSB_TABLE.
                     0000   321 ;-
                     0000   322
                     0000   323          $DEFINI CSB     GLOBAL
00000004             0000   324                          .BLKL   1
00000008             0004   325                          .BLKL   1
                     0008   326 $DEF     CSB_W_SIZE      .BLKW   1       ; Size of this block in bytes
                     000A   327 $DEF     CSB_B_TYPE      .BLKB   1       ; Type of block
                     000B   328 $DEF     CSB_B_STATE     .BLKB   1       ; Virtual circuit state:
                     000C   329  $EQU    CSB_C_STATE_HALT        0       ;* Circuit halted state
                     000C   330  $EQU    CSB_C_STATE_START       1       ;* Circuit starting state
                     000C   331  $EQU    CSB_C_STATE_RUN         2       ;* Circuit running state
                     000C   332 ;
                     000C   333 ;       Counter area - must follow standard block header!
                     000C   334 ;
                     000C   335          ASSUME  GHB$T_CSBCTR EQ .
                     000C   336 $DEF     CSB_Z_LCB       .BLKB   LCB_C_LENGTH ; Circuit block counter area
                     001C   337 ;
                     001C   338 ;       Server name area - must immediately follow counter area
                     001C   339 ;
                     001C   340 $DEF     CSB_B_SERVER    .BLKB   1       ; Length of server name
                     001D   341 $DEF     CSB_T_SERVER    .BLKB   GHB$K_NAMELEN ; Server name field
                     002D   342 $DEF     CSB_B_REFC      .BLKB   1       ; Number of UCBs attached to this CSB
                     002E   343 ;
                     002E   344 ;       Destination Ethernet address - must immediately follow server name area
                     002E   345 ;
                     002E   346 $DEF     CSB_Z_DST       .BLKW   3       ; Concentrator physical address
                     0034   347
                     0034   348 $DEF     CSB_Q_XBUFQ     .BLKQ   1       ; Current transmit buffer queue
                     003C   349 $DEF     CSB_Q_XWAITQ    .BLKQ   1       ; Transmit buffers waiting queue
                     0044   350 $DEF     CSB_L_XCXB      .BLKL   LAT$C_XMT_BUFFERS ; Transmit CXBs in progress
                     004C   351 $DEF     CSB_L_STOPREASON .BLKL  1       ; Address of the stop reason block
                     0050   352 $DEF     CSB_W_TIMEOUT   .BLKW   1       ; Timeout cell for CSB
                     0052   353 $DEF     CSB_W_XMTTMO    .BLKW   1       ; Timeout count for retransmit
                     0054   354 $DEF     CSB_W_PROGRESS  .BLKW   1       ; Progress counter
                     0056   355 $DEF     CSB_W_PROGSEQ   .BLKW   1       ; Transmit error control for progress
                     0058   356 $DEF     CSB_B_XMTBSY    .BLKB   1       ; Transmit busy and count of waiters
                     0059   357 $DEF     CSB_B_XMTCNT    .BLKB   1       ; Count of buffers to transmit
                     005A   358 $DEF     CSB_B_XCXB_INX  .BLKB   1       ; XCXB index
                     005B   359 $DEF     CSB_B_SPARE     .BLKB   1       ; SPARE
                     005C   360
                     005C   361 ; Start of circuit header as it appears in transmitted messages
                     005C   362
                     005C   363 $DEF     CSB_T_CIRCHDR                   ; Lat circuit header
                     005C   364 $DEF     CSB_B_FLAG      .BLKB   1       ; Message flags byte
                     005D   365 $DEF     CSB_B_NUM_SLOTS .BLKB   1       ; Number of slots
                     005E   366 $DEF     CSB_W_REMID                     ; Remote virtual circuit id
                     005E   367 $DEF     CSB_B_REMIDN    .BLKB   1       ; Remote virtual circuit id number
                     005F   368 $DEF     CSB_B_REMIDS    .BLKB   1       ; Remote virtual circuit id sequence
                     0060   369 $DEF     CSB_W_LOCID                     ; Local virtual circuit id
```

```
0060   370 $DEF    CSB_B_LOCIDN     .BLKB   1        ; Local virtual circuit id number
0061   371 $DEF    CSB_B_LOCIDS     .BLKB   1        ; Local virtual circuit id sequence
0062   372 $DEF    CSB_W_XSEQ                        ; Transmiter error control
0062   373 $DEF    CSB_B_XSEQ       .BLKB   1        ; Sequence number being transmitted
0063   374 $DEF    CSB_B_XACK       .BLKB   1        ; Highest sequence number received
0064   375  $EQU   CSB_S_CIRCHDR    <.-CSB_T_CIRCHDR> ; Size of header
0064   376
0064   377 ; End of circuit header
0064   378
0064   379 $DEF    CSB_W_RSEQ                        ; Sequence number to receive
0064   380 $DEF    CSB_B_RSEQ       .BLKB   1        ; Sequence number to receive
0065   381 $DEF    CSB_B_RACK       .BLKB   1        ; Highest sequence number acknowledged
0066   382 $DEF    CSB_W_TIMRESET   .BLKW   1        ; Timeout reset value
0068   383 $DEF    CSB_B_INX        .BLKB   1        ; CSB table index
0069   384 $DEF    CSB_B_MAX_SLOTS  .BLKB   1        ; Maximum number of slots (negotiated)
006A   385 $DEF    CSB_W_MAX_MSGSIZ .BLKW   1        ; Maximum message size (negotiated)
006C   386 $DEF    CSB_C_FIXLENGTH                   ; Length of fixed block
006C   387 ;
006C   388 ;       UCB list area  (size = 4*min(lat$c_max_slots,server_slot_count) )
006C   389 ;
006C   390        ALIGN_LONG                        ; align to longword boundary
006C   391 $DEF    CSB_L_UCBLST
006C   392
006C   393        $DEFEND CSB
```

```
                        ;+
                        ;       Terminal overflow input buffer definitions
                        ;
                        ;  The terminal overflow buffer is only used when a call is made to
                        ;  the XOFF routine to stop the input flow. The overflow buffer is
                        ;  allocated and the rest of the input data is copied to the overflow
                        ;  buffer. When we are resumed at XON, the data from the input overflow
                        ;  buffer is dumped into the class driver.
                        ;
                        ;-

                                $DEFINI INB     GLOBAL

0000    $DEF    INB_W_BOFF      .BLKW   1       ; Offset to start of data
0002    $DEF    INB_W_BCNT      .BLKW   1       ; Count of data remaining
0004    $DEF    INB_L_SPARE     .BLKL   1       ; Spare longword
0008    $DEF    INB_W_SIZE      .BLKW   1       ; Size of structure
000A    $DEF    INB_B_TYPE      .BLKB   1       ; Type of structure
000B    $DEF    INB_B_SPARE     .BLKB   1       ; Spare byte
000C    $DEF    INB_C_LENGTH                    ; Size of structure header

                                $DEFEND INB
```

```
                     0000   420
                     0000   421 ;+
                     0000   422 ;       Receive buffer offsets and bit definitions
                     0000   423 ;
                     0000   424 ;  The format of receive and transmit buffers is identical for fields
                     0000   425 ;  Starting with and following the destination address field. Ease of
                     0000   426 ;  implemenatation  dictates that the offsets  be defined relative to
                     0000   427 ;  the start of the buffer.
                     0000   428 ;
                     0000   429 ;  These are the offsets into the architecturally defined portion of
                     0000   430 ;  the receive buffer itself.
                     0000   431 ;-
                     0000   432
                     0000   433          $DEFINI RCV     GLOBAL
                     0000   434
00000000             0000   435 .=0
                     0000   436 $DEF     RCV_T_DATA                      ; Start of data is right here
                     0000   437 $DEF     RCV_B_FLAG      .BLKB   1       ; Flags byte and message type
                     0001   438          ;
                     0001   439          ; Flag bits:
                     0001   440          ;
                     0001   441 _VIELD   FLAG,0,<-                       ;* Flags byte bit definitions recv & xmit
                     0001   442          <RRF,1,M>,-                     ;* Response requested flag
                     0001   443          <MASTER,1,M>,-                  ;* Remote station always sets this flag
                     0001   444          <MTYPE,6,M>,-                   ;* Message type bits
                     0001   445          >
                     0001   446          ;
                     0001   447          ; Message types:
                     0001   448          ;
                     0001   449 $EQU     MTYP_C_RUN      0               ;* Run message
                     0001   450 $EQU     MTYP_C_START    1               ;* Start message
                     0001   451 $EQU     MTYP_C_HALT     2               ;* Halt message
                     0001   452 $EQU     MTYP_C_CONF     10              ;* Configuration message
                     0001   453
                     0001   454 $DEF     RCV_B_NUM_SLOTS .BLKB   1       ; Number of slots in buffer
                     0002   455 $DEF     RCV_W_DSTID                     ; Destination virtual circuit id
                     0002   456 $DEF     RCV_B_DSTIDN    .BLKB   1       ; Destination virtual circuit id number
                     0003   457 $DEF     RCV_B_DSTIDS    .BLKB   1       ; Destination virtual circuit id seq
                     0004   458 $DEF     RCV_W_SRCID                     ; Source virtual circuit id
                     0004   459 $DEF     RCV_B_SRCIDN    .BLKB   1       ; Source virtual circuit id number
                     0005   460 $DEF     RCV_B_SRCIDS    .BLKB   1       ; Source virtual circuit id
                     0006   461                                         ;  sequence number
                     0006   462
                     0006   463 $DEF     RCV_B_SEQ       .BLKB   1       ; Buffer sequence number
                     0007   464 $DEF     RCV_B_ACK       .BLKB   1       ; Sequence number acknowledgement
                     0008   465 $DEF     RCV_T_MDATA                     ; Start of type dependent data
                     0008   466
                     0008   467          $DEFEND RCV
```

```
0000    469
0000    470 ;
0000    471 ; Define START message offsets (used for both xmits and recvs)
0000    472 ;
0000    473          $DEFINI STRT     GLOBAL
0000    474 $DEF     STRT_W_MSGSIZ    .BLKW   1          ; Data link frame size
0002    475 $DEF     STRT_B_PVER      .BLKB   1          ; Circuit protocol version
0003    476 $DEF     STRT_B_PECO      .BLKB   1          ; Circuit Eng. change order level
0004    477 $DEF     STRT_B_MAXSLOTS  .BLKB   1          ; Maximum simultaneous slots
0005    478 $DEF     STRT_B_DL_BFRS   .BLKB   1          ; Number of extra rcv bfrs in server
0006    479 $DEF     STRT_B_CIR_TIMR  .BLKB   1          ; Local circuit timer in milliseconds
0007    480 $DEF     STRT_B_KPA_TIMR  .BLKB   1          ; Local keep alive timer in seconds
0008    481 $DEF     STRT_W_FAC_NUM   .BLKW   1          ; Facility number
000A    482 $DEF     STRT_W_PROD_TYP  .BLKW   1          ; Product type code
000C    483 $DEF     STRT_C_LENGTH                       ; Length of fixed portion of message
000C    484 ;
000C    485 ; Start of variable length parameters
000C    486 ;
000C    487 $DEF     STRT_B_VAR                          ; Start of variable portion of start message
000C    488 $DEF     STRT_B_NODE_LEN  .BLKB   1          ; Destination node name length
000D    489 $DEF     STRT_T_NODE      .BLKB   GHB$K_NAMELEN ; Destination node name text
001D    490 $DEF     STRT_B_SYS_LEN   .BLKB   1          ; Source node name length
001E    491 $DEF     STRT_T_SYS       .BLKB   GHB$K_NAMELEN ; Source node name text
002E    492 $DEF     STRT_B_ID_LEN    .BLKB   1          ; Source node descriptor length
002F    493 $DEF     STRT_T_ID        .BLKB   GHB$K_IDLEN ; Source node descriptor text
006F    494 $DEF     STRT_B_LOC_LEN   .BLKB   1          ; Source location length
0070    495 $DEF     STRT_T_LOC       .BLKB   LAT$C_LOC_LEN ; Source location text
00B0    496 $DEF     STRT_B_PARAM     .BLKB   1          ; Parameter code
00B1    497  $EQU    STRT_C_VAR_LEN   <.-STRT_B_VAR>     ; Length of variable portion of start
00B1    498                                              ;   message
00B1    499          $DEFEND STRT
0000    500
0000    501 ;
0000    502 ; Define STOP message offsets
0000    503 ;
0000    504          $DEFINI STOP     GLOBAL
0000    505 $DEF     STOP_B_RCODE     .BLKB              ; Circuit disconnect reason
0001    506 $DEF     STOP_B_RLEN      .BLKB              ; Reason length in bytes
0002    507 $DEF     STOP_T_REAS                         ; Reason ASCII text
0002    508          $DEFEND STOP
0000    509
```

```
                     0000   511
                     0000   512 ;+
                     0000   513 ;       Transmit buffer offsets and bit definitions
                     0000   514 ;
                     0000   515 ;       The buffer header is defined to be the same as that used by
                     0000   516 ;       the Ethernet datalink drivers.
                     0000   517 ;-
                     0000   518
                     0000   519         $DEFINI CXB     GLOBAL          ; Transmit complex buffer
                     0000   520 ;
                     0000   521 ; Define the transmit/receive offsets in the CXB
                     0000   522 ;
0000001C             0000   523 . = CXB$L_END_ACTION
                     001C   524 $DEF     CXB$L_T_ENDADR  .BLKL   1       ; End address of data to transmit
                     0020   525
0000000C             0020   526 . = CXB$W_LENGTH
                     000C   527         ASSUME  CXB$W_OFFSET EQ CXB$W_LENGTH+2
                     000C   528 $DEF     CXB$L_T_CSB                     ; CSB address
                     000C   529
00000014             000C   530 . = CXB$L_IRP
                     0014   531 $DEF     CXB$L_T_SAVE    .BLKL   1       ; BOFF & BCNT save area
                     0018   532
                     0018   533         $DEFEND CXB
                     0000   534
                     0000   535 ;
                     0000   536 ; Start of architecturally defined offsets
                     0000   537 ;
                     0000   538         $DEFINI XMT     GLOBAL
                     0000   539
0000003A             0000   540 . = CXB$C_HEADER - <6+6+2>
                     003A   541 $DEF     XMT_G_DST       .BLKW   3       ; Destination address
                     0040   542
00000048             0040   543 . = CXB$C_HEADER
                     0048   544 $DEF     XMT_T_DATA                      ; Start of transmitted data
                     0048   545 $DEF     XMT_B_FLAG      .BLKB   1       ; Flags byte and message type
                     0049   546
                     0049   547 ;
                     0049   548 ; Run message format:
                     0049   549 ;
                     0049   550 $DEF     XMT_B_NUM_SLOTS .BLKB   1       ; Number of slots in buffer
                     004A   551 $DEF     XMT_W_DSTID                     ; Destination virtual circuit id
                     004A   552 $DEF     XMT_B_DSTIDN    .BLKB   1       ; Destination virtual circuit id number
                     004B   553 $DEF     XMT_B_DSTIDS    .BLKB   1       ; Destination virtual circuit id seq
                     004C   554 $DEF     XMT_W_SRCID                     ; Source virtual circuit id
                     004C   555 $DEF     XMT_B_SRCIDN    .BLKB   1       ; Source virtual circuit id number
                     004D   556 $DEF     XMT_B_SRCIDS    .BLKB   1       ; Source virtual circuit id
                     004E   557                                         ;   sequence number
                     004E   558
                     004E   559 $DEF     XMT_B_SEQ       .BLKB   1       ; Buffer sequence number
                     004F   560 $DEF     XMT_B_ACK       .BLKB   1       ; Sequence number acknowledgement
                     0050   561 $DEF     XMT_T_MDATA                     ; Start of RUN message data
                     0050   562
                     0050   563 ;
                     0050   564 ; Multicast (CONFIGURATION) message format:
                     0050   565 ;
00000049             0050   566         . = XMT_B_FLAG+1
                     0049   567 $DEF     XMT_B_MC_CIR_TIMER      .BLKB   1       ; Circuit timer
```

```
004A   568 $DEF     XMT_B_MC_HI_VER         .BLKB   1       ; Highest supported protocol version
004B   569 $DEF     XMT_B_MC_LO_VER         .BLKB   1       ; Lowest supported protocol version
004C   570 $DEF     XMT_B_MC_CUR_VER        .BLKB   1       ; version of this message
004D   571 $DEF     XMT_B_MC_CUR_ECO        .BLKB   1       ; eco of this message
004E   572 $DEF     XMT_B_MC_INCARN         .BLKB   1       ; Message incarnation
004F   573 $DEF     XMT_B_MC_CHG_FLAG       .BLKB   1       ; Changed fields in this msg
0050   574
0050   575             VIELD  XMT_CHFLG,0,<-
0050   576             <GROUP,1,M>,-                           ;* NODE group codes changed
0050   577             <NDESC,1,M>,-                           ;* NODE descriptor changed
0050   578             <SVCNAM,1,M>,-                          ;* Service name/number changed
0050   579             <RATE,1,M>,-                            ;* Service ratings changed
0050   580             <SDESC,1,M>,-                           ;* Service classes changed
0050   581             <CLASS,1,M>,-                           ;* Service classes changed
0050   582             <,1>,-                                  ;* RESERVED
0050   583             <OTHER,1,M>,-                           ;* One of the OTHER fields changed
0050   584             >
0050   585  $EQU     XMT_CHFLG_M_ALL <^XFF>                  ;* All codes changed
0050   586
0050   587 $DEF     XMT_W_MC_MSG_SIZ        .BLKW   1       ; Maximum frame size
0052   588  $EQU     XMT_C_MC_LENGTH         <.-XMT_T_DATA>  ; Size of fixed part of multicast me
0052   589
0052   590 $DEF     XMT_B_MC_SET            ; Start of LATCP set portion of multicast msg
0052   591 $DEF     XMT_B_MC_MULTIMR        .BLKB   1       ; Multicast message timer
0053   592 $DEF     XMT_B_MC_STATUS         .BLKB   1       ; Node status
0054   593 $DEF     XMT_B_MC_GRP_LEN        .BLKB   1       ; Number of group codes
0055   594 $DEF     XMT_T_MC_GROUP          .BLKB   LAT$C_GRP_LEN ; Group code field
0075   595 $DEF     XMT_B_MC_NODE_LEN       .BLKB   1       ; Node name length
0076   596 $DEF     XMT_T_MC_NODE           .BLKB   GHB$K_NAMELEN ; NODE name
0086   597 $DEF     XMT_B_MC_NUM_SVCS       .BLKB   1       ; Number of services
0087   598
0087   599 $DEF     XMT_B_MC_RATE           .BLKB   1       ; Service rating
0088   600 $DEF     XMT_B_MC_NAME_LEN       .BLKB   1       ; Service name length
0089   601 $DEF     XMT_T_MC_NAME           .BLKB   GHB$K_NAMELEN ; Service name
0099   602 $DEF     XMT_B_MC_ID_LEN         .BLKB   1       ; Length of Identification field
009A   603 $DEF     XMT_T_MC_ID             .BLKB   GHB$K_IDLEN ; Identification field
00DA   604  $EQU     XMT_C_MC_SVC_SIZE       <.-XMT_B_MC_RATE> ; Size of each service field
00DA   605
00DA   606 $DEF     XMT_T_MC_MORE_SVC       .BLKB   <<LAT$C_MAX_SERVICES-1>*XMT_C_MC_SVC_SIZE>
0279   607
0279   608 $DEF     XMT_B_MC_SVC_LEN        .BLKB   1       ; Number of service classes
027A   609 $DEF     XMT_B_MC_SVC            .BLKB   LAT$C_SVC_LEN ; Service classes
0282   610  $EQU     XMT_C_MC_TOTAL  <.-XMT_T_DATA>          ; Length of total message
0282   611             ASSUME  XMT_C_MC_TOTAL LE LAT$C_MAX_MSGSIZ
0282   612
0282   613  $EQU     XMT_C_HDRLEN    <XMT_T_MDATA - XMT_T_DATA>
0282   614                                                  ; Length of header
0282   615  $EQU     XMT_C_MAXDATA   <LAT$C_MAX_MSGSIZ - XMT_C_HDRLEN>
0282   616                                                  ; Data in slots
0282   617  $EQU     XMT_C_MAXLEN    <LAT$C_MAX_MSGSIZ + XMT_T_DATA>
0282   618                                                  ; Total size of a buffer
0282   619             $DEFEND XMT
```

```
                        0000   621 ;+
                        0000   622 ;       Buffer slot offsets
                        0000   623 ;
                        0000   624 ;       Each slot is formatted identically in both
                        0000   625 ;       transmit and receive buffers as defined below.
                        0000   626 ;-
                        0000   627
                        0000   628          $DEFINI SLT     GLOBAL
                        0000   629
                        0000   630 $DEF     SLT_T_START                     ; Start of slot
                        0000   631 $DEF     SLT_B_DSTID     .BLKB   1       ; Destination slot ID
                        0001   632 $DEF     SLT_B_SRCID     .BLKB   1       ; Source slot ID
                        0002   633 $DEF     SLT_B_COUNT     .BLKB   1       ; Character count in slot
                        0003   634 $DEF     SLT_B_REAS                      ; Stop/reject slot reason (low nibble)
                        0003   635 $DEF     SLT_B_CRED                      ; Credits extended (low order nibble)
                        0003   636 $DEF     SLT_B_TYPE      .BLKB   1       ; Slot type (high order nibble)
                        0004   637          ; STOP/REJECT reason codes:
                        0004   638  $EQU    SLT_C_USD       1               ; * User requested disconnect
                        0004   639  $EQU    SLT_C_SYS       2               ; * System shutdown
                        0004   640  $EQU    SLT_C_INVSLOT   3               ; * Invalid slot received
                        0004   641  $EQU    SLT_C_INVSVC    4               ; * Invalid service class received
                        0004   642  $EQU    SLT_C_RESOURCE  5               ; * Insufficient resources
                        0004   643          ; Slot type codes:
                        0004   644  $EQU    SLT_C_DA_SLOT   <^X00>          ; * DATA_A slot
                        0004   645  $EQU    SLT_C_STR_SLOT  <^X09>          ; * START slot
                        0004   646  $EQU    SLT_C_DB_SLOT   <^X0A>          ; * DATA_B slot
                        0004   647  $EQU    SLT_C_ATT_SLOT  <^X0B>          ; * ATTENTION slot
                        0004   648  $EQU    SLT_C_REJ_SLOT  <^X0C>          ; * REJECT slot
                        0004   649  $EQU    SLT_C_STP_SLOT  <^X0D>          ; * STOP slot
                        0004   650 $DEF     SLT_T_DATA                      ; Start of slot data
                        0004   651
                        0004   652 ;
                        0004   653 ; Start slot data definitions
                        0004   654 ;
00000004                0004   655 . = SLT_T_DATA
                        0004   656 $DEF     SLT_B_SVCC      .BLKB   1       ; Service class
                        0005   657 $EQU     SLT_C_SVC_INTT  1               ; * Interactive Terminals
                        0005   658 $DEF     SLT_B_AT_SLTSIZ .BLKB   1       ; Maximum length of attention slot
                        0006   659 $DEF     SLT_B_DT_SLTSIZ .BLKB   1       ; Maximum length of Data slot
                        0007   660 $DEF     SLT_B_DST_NAMLEN .BLKB  1       ; Length of destination slot name
                        0008   661 $DEF     SLT_T_DST_NAME  .BLKB   GHB$K_NAMELEN
                        0018   662 $DEF     SLT_B_SRC_NAMLEN .BLKB  1       ; Length of source slot name
                        0019   663 $DEF     SLT_B_SRC_NAME  .BLKB   GHB$K_NAMELEN
                        0029   664  $EQU    SLT_S_START_SLOT <.-SLT_T_DATA> ; Length of Start slot
                        0029   665 $DEF     SLT_B_PARAMS                    ; Start of parameter area
                        0029   666  $EQU    SLT_C_START_FLAG        1       ;* Start slot parameter flag is code 1
                        0029   667          _VIELD  SLT_FLAG,0,<-
                        0029   668                  <REMOTE,1,M>,-          ;* Remote line
                        0029   669                  <LOGIN,1,M>,-           ;* Disable auto-login
                        0029   670                  <BIND,1,M>,-            ;* Terminal UCB is bound to server
                        0029   671                  >
                        0029   672
                        0029   673 ;
                        0029   674 ; DATA_B slot data definitions
                        0029   675 ;
00000004                0029   676 . = SLT_T_DATA
                        0004   677 $DEF     SLT_B_DB_CONTROL        .BLKB   1       ; DATA_B slot control flag
```

```
                     0005   678          _VIELD  SLT_DB,0,<-
                     0005   679                  <IFENA,1,M>,-                   ; Turn on flow control of keyboard i
                     0005   680                  <IFDIS,1,M>,-                   ; Turn off flow control of keyb inpu
                     0005   681                  <OFENA,1,M>,-                   ; Output flow on
                     0005   682                  <OFDIS,1,M>,-                   ; Output flow off
                     0005   683                  <BREAK,1,M>,-                   ; Break condition detected (recv slo
                     0005   684                  >
                     0005   685 $DEF     SLT_B_DB_OFOF   .BLKB   1               ; Output-off flow char (from keyboar
                     0006   686 $DEF     SLT_B_DB_OFON   .BLKB   1               ; Output-on flow char (from keyboard
                     0007   687 $DEF     SLT_B_DB_IFOF   .BLKB   1               ; Input-off flow character (from hos
                     0008   688 $DEF     SLT_B_DB_IFON   .BLKB   1               ; Input-on flow character (from host
                     0009   689  $EQU    SLT_S_DB_LEN    <.-SLT_T_DATA>          ; Length of DATA_B data
                     0009   690  $EQU    SLT_S_DB_SLOT   <.-SLT_T_START>         ; Length of DATA_B slot
                     0009   691
                     0009   692 ;
                     0009   693 ; ATTention slot data definitions
                     0009   694 ;
00000004             0009   695 . = SLT_T_DATA
                     0004   696 $DEF     SLT_B_ATT_CONTROL       .BLKB   1       ; ATTENTION slot control flag
                     0005   697          _VIELD  SLT_ATT,0,<-
                     0005   698                  <,5>,-                          ; RESERVED
                     0005   699                  <ABORT,1,M>,-                   ; Abort (flush pipe)
                     0005   700                  >
                     0005   701  $EQU    SLT_S_ATT_LEN   <.-SLT_T_DATA>          ; Length of Attention data
                     0005   702  $EQU    SLT_S_ATT_SLOT  <.-SLT_T_START>         ; Length of Attention slot
                     0005   703
                     0005   704 ;
                     0005   705 ; STOP slot data definitions
                     0005   706 ;
00000004             0005   707 . = SLT_T_DATA
                     0004   708 $DEF     SLT_B_STP_RLEN  .BLKB   1               ; STOP slot reason length
                     0005   709  $EQU    SLT_S_STP_LEN   <.-SLT_T_DATA>          ; Length of fixed STOP slot data
                     0005   710  $EQU    SLT_S_STP_SLOT  <.-SLT_T_START>         ; Length of fixed STOP slot
                     0005   711 $DEF     SLT_T_STP_REAS                          ; STOP slot reason (variable)
                     0005   712
                     0005   713 ;
                     0005   714 ; REJect slot data definitions
                     0005   715 ;
00000004             0005   716 . = SLT_T_DATA
;JAY0002                    0004    .1 $DEF     SLT_B_REJ_RLEN                  ; REJ slot reason length
-1                   0004   718  $EQU    SLT_S_REJ_LEN   <.-SLT_T_DATA>          ; Length of fixed REJ slot data
                     0004   719  $EQU    SLT_S_REJ_SLOT  <.-SLT_T_START>         ; Length of fixed REJ slot
                     0004   720 $DEF     SLT_T_REJ_REAS                          ; REJ slot reason (variable)
                     0004   721
                     0004   722          $DEFEND SLT
```

```
0000    724 ;
0000    725 ;       Local macros
0000    726 ;
0000    727
0000    728 .MACRO  INC_CTR,        BASE, TYPE=L, ?L
0000    729         INC'TYPE        BASE                        ; Increment counter
0000    730         BNEQ    L                                   ; Br if no overflow
0000    731         DEC'TYPE        BASE                        ; Else, latch at maximum
0000    732 L:
0000    733 .END    INC_CTR
0000    734
0000    735 .MACRO  ADD_CTR,        BASE1, BASE2, TYPE=L, ?L
0000    736         ADD'TYPE        BASE1,BASE2                 ; Add counters
0000    737         BCC     L                                   ; Br if no overflow
0000    738         MNEG'TYPE       #1,BASE2                    ; Else, latch at maximum
0000    739 L:
0000    740 .END    ADD_CTR
0000    741
0000    742 .MACRO  $DISPATCH,      INDX,VECTOR,TYPE=W,NMODE=S^#,?MN,?MX,?S,?SS,?ZZ
0000    743 SS:
0000    744
0000    745         .MACRO  $DSP1,$DSP1_1
0000    746                 .IRP    $DSP1_2,$DSP1_1
0000    747                         $DSP2   $DSP1_2
0000    748                 .ENDR
0000    749         .ENDM   $DSP1
0000    750
0000    751         .MACRO  $DSP2,$DSP2_1,$DSP2_2
0000    752                 .=<$DSP2_1-MN>*2 + S
0000    753                 .WORD   $DSP2_2-S
0000    754         .ENDM   $DSP2
0000    755
0000    756
0000    757         .MACRO  $BND1,$BND1_1,$BND1_2,$BND1_3
0000    758                 $BND2   $BND1_1,$BND1_2
0000    759         .ENDM   $BND1
0000    760
0000    761         .MACRO  $BND2,$BND2_1,$BND2_2
0000    762                 .IIF    $BND2_1,$BND2_2-.,      .=$BND2_2
0000    763         .ENDM   $BND2
0000    764
0000    765         .MACRO  $BND    $BND_1,$BND_2
0000    766                 .IRP    $BND_3,<$BND_2>
0000    767                         $BNDT   $BND_1,$BND_3
0000    768                 .ENDR
0000    769         .ENDM   $BND
0000    770
0000    771         .=0
0000    772 ZZ:
0000    773         $BND    GT,<VECTOR>
0000    774 MX:
0000    775         $BND    LT,<VECTOR>
0000    776 MN:
0000    777         .=SS
0000    778
0000    779 CASE'TYPE       INDX,#<MN-ZZ>,NMODE'<MX-MN>
0000    780 S:
```

```
0000   781          .REPT   MX-MN+1
0000   782          .WORD   <MX-MN>*2 + 2
0000   783          .ENDR
0000   784
0000   785          .=S
0000   786
0000   787          $DSP1   <<VECTOR>>
0000   788
0000   789          .=<MX-MN>*2 + S + 2
0000   790
0000   791 .ENDM    $DISPATCH
0000   792
0000   793
0000   794          .MACRO  SETBIT BIT, BASE, ?L    ; Set specified bit
0000   795          BBSS    BIT, BASE, L
0000   796 L:
0000   797          .ENDM
0000   798
0000   799          .MACRO  CLRBIT BIT, BASE, ?L    ; Clear specified bit
0000   800          BBCC    BIT, BASE, L
0000   801 L:
0000   802          .ENDM
0000   803
0000   804          .MACRO  BREAK
0000   805          JSB     G^INI$BRK
0000   806          .ENDM
0000   807
0000   808          .MACRO  DEBUG
0000   809          TSTL    @#-1
0000   810          .ENDM   DEBUG
0000   811
```

```
                        .SBTTL  Driver Prologue Table
                        .PSECT  $$$105_PROLOGUE

;
; Driver prologue table:
;

LT$DPT::                                        ; Driver start
        DPTAB   -                               ; Driver prologue table
                END=LT$END,-                    ; Address of end of module
                UCBSIZE=UCB$C_LT_LENGTH,-       ; Size of each ucb of port driver
                ADAPTER=NULL,-                  ; Adapter type
                NAME=LTDRIVER,-                 ; Name of driver
                VECTOR=LT$VECTOR                ; Port driver vector table
        DPT_STORE INIT
        DPT_STORE UCB,UCB$B_FIPL,B,LAT$C_IPL ; Fork ipl
        DPT_STORE UCB,UCB$L_DEVCHAR,L,<-        ; Characteristics
                        DEV$M_REC!-             ;
                        DEV$M_AVL!-             ;
                        DEV$M_IDV!-             ;
                        DEV$M_ODV!-             ;
                        DEV$M_TRM!-             ;
                        DEV$M_CCL>
        DPT_STORE UCB,UCB$L_DEVCHAR2,L,<-
                        DEV$M_NNM>              ; Prefix name with <node$>
        DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_TERM;
        DPT_STORE UCB,UCB$B_TT_DETYPE,B,TT$_UNKNOWN     ; Type
        DPT_STORE UCB,UCB$W_TT_DESIZE,aW,TTY$GW_DEFBUF  ; Buffer size
        DPT_STORE UCB,UCB$L_TT_DECHAR,aL,TTY$GL_DEFCHAR ; Default characters
        DPT_STORE UCB,UCB$L_TT_DECHA1,aL,TTY$GL_DEFCHAR2; Default characters
        DPT_STORE UCB,UCB$W_TT_DESPEE,aB,TTY$GB_DEFSPEED; Default speed
        DPT_STORE UCB,UCB$W_TT_DESPEE+1,aB,TTY$GB_RSPEED; Default rspeed
        DPT_STORE UCB,UCB$B_TT_DEPARI,aB,TTY$GB_PARITY  ; Default parity
;**     DPT_STORE UCB,UCB$B_TT_PARITY,aB,TTY$GB_PARITY  ; Default parity
        DPT_STORE UCB,UCB$B_DEVTYPE,B,TT$_UNKNOWN       ; Type
        DPT_STORE UCB,UCB$W_DEVBUFSIZ,aW,TTY$GW_DEFBUF  ; Buffer size
        DPT_STORE UCB,UCB$L_DEVDEPEND,aL,TTY$GL_DEFCHAR ; Default characters
        DPT_STORE UCB,UCB$L_TT_DEVDP1,aL,TTY$GL_DEFCHAR2; Default characters
        DPT_STORE UCB,UCB$W_TT_SPEED,aB,TTY$GB_DEFSPEED ; Default speed
        DPT_STORE UCB,UCB$W_TT_SPEED+1,aB,TTY$GB_RSPEED ; Default rspeed
        DPT_STORE UCB,UCB$B_DIPL,B,LAT$C_IPL            ; Device ipl
        DPT_STORE UCB,UCB$L_TT_WFLINK,L,0               ; Zero write queue.
        DPT_STORE UCB,UCB$L_TT_WBLINK,L,0               ; Zero write queue.
        DPT_STORE UCB,UCB$L_TT_RTIMOU,L,0               ; Zero read timed out
;**     DPT_STORE UCB,UCB$W_TT_PRTCTL,W,TTY$M_PC_NOTIME ; Disable all timers
        DPT_STORE ORB,ORB$B_FLAGS,B,-                   ; Default flags
                        <ORB$M_PROT_16>                 ; SOGW word protection
        DPT_STORE ORB,ORB$W_PROT,aW,TTY$GW_PROT         ; Default protection
        DPT_STORE ORB,ORB$L_OWNER,aL,TTY$GL_OWNUIC      ; Default owner uic
        DPT_STORE DDB,DDB$L_DDT,D,LT$DDT

        DPT_STORE REINIT
        DPT_STORE CRB,CRB$L_INTD+VEC$L_INITIAL,D,LT$CTRL_INIT ; Controller init
        DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,D,LT$UNIT_INIT ; Unit init
        DPT_STORE END

```

```
0000    870
0000    871
0000    872
0000    873
0000    874          DDTAB   DEVNAM = LT,-   ; Dummy port driver dispatch table
0000    875                  START  = 0,-
0000    876                  FUNCTB = 0
0038    877
```

```
                     0038   879          .SBTTL  Local Storage
                     0038   880
                     0038   881
                00000038    882          .PSECT  $$$115_DRIVER,LONG
                     0038   883
                     0038   884 ;
                     0038   885 ; Local storage
                     0038   886 ;
                     0038   887
                     0038   888 ;+
                     0038   889 ;       Circuit State Block table
                     0038   890 ;
                     0038   891 ;  This table is indexed by RCV_B_DSTIDN in received buffers. A
                     0038   892 ;  concentrator always uses the same value when refering to a
                     0038   893 ;  host for a second time. The RCV_B_DSTIDS byte is a check byte
                     0038   894 ;  which helps prevent random or outdated references to CSBs.
                     0038   895 ;
                     0038   896 ;-
                     0038   897
                     0038   898          .ALIGN  LONG
                     0038   899
                     0038   900 GHB_L_CSBTABLE:
000000B8             0038   901          .BLKL   LAT$C_MAX_CSBS                  ; CSB address table
                     00B8   902 GHB_B_LOC_CCHK:                                  ; Table of local
000000D9             00B8   903          .BLKB   LAT$C_MAX_CSBS+1                ; circuit checks
                     00D9   904
                     00D9   905
                     00D9   906 ;
                     00D9   907 ;       Timer queue element
                     00D9   908 ;
                     00D9   909 ;       Used to timeout CSB's that have had no activity.
                     00D9   910 ;
                     00D9   911
                     00D9   912          .ALIGN  LONG
                     00DC   913 GHB_G_TQE:
0000010C             00DC   914          .BLKB   TQE$C_LENGTH
                     010C   915
;JAY0002              010C    .1 ;
;JAY0002              010C    .2 ; The following is used during controller init to determine what the
;JAY0002              010C    .3 ; maximum number of slots supported by this VAX processor.
;JAY0002              010C    .4 ;
;JAY0002              010C    .5
;JAY0002              010C    .6 GHB_B_MAX_SLOTS:
;JAY0002           00 010C    .7          .BYTE   0                   ; Cell to contain max slot value
;JAY00C2              010D    .8
;JAY0002              010D    .9 GHB_T_MAX_SLOT_TABLE:
;JAY0002           40 010D   .10          .BYTE   LAT$C_MAX_SLOTS ;                       0 (non-existent)
;JAY0002           40 010E   .11          .BYTE   64              ; PR$_SID_TYP780        1
;JAY0002           40 010F   .12          .BYTE   64              ; PR$_SID_TYP750        2
;JAY0002           20 0110   .13          .BYTE   32              ; PR$_SID_TYP730        3
;JAY0002           FF 0111   .14          .BYTE   255             ; PR$_SID_TYP790        4
;JAY0002           40 0112   .15          .BYTE   64              ; PR$_SID_TYP8SS        5
;JAY0002           80 0113   .16          .BYTE   128             ; PR$_SID_TYP8NN        6
;JAY0002           10 0114   .17          .BYTE   16              ; PR$_SID_TYPUV1        7
;JAY0002           30 0115   .18          .BYTE   48              ; PR$_SID_TYPUV2        8
;JAY0002           40 0116   .19          .BYTE   LAT$C_MAX_SLOTS ;                       9
;JAY0002           40 0117   .20          .BYTE   LAT$C_MAX_SLOTS ;  (patch area)         10
```

```
;JAY0002                40  0118  .21         .BYTE   LAT$C_MAX_SLOTS ;                    11
;JAY0002                40  0119  .22         .BYTE   LAT$C_MAX_SLOTS ;                    12
;JAY0002                40  011A  .23         .BYTE   LAT$C_MAX_SLOTS ;                    13
;JAY0002                40  011B  .24         .BYTE   LAT$C_MAX_SLOTS ;                    14
;JAY0002                40  011C  .25         .BYTE   LAT$C_MAX_SLOTS ;                    15
;JAY0002                40  011D  .26         .BYTE   LAT$C_MAX_SLOTS ;                    16
;JAY0002                    011E  .27
;JAY0002          00000010  011E  .28         MAX_PR_SID = 16
```

```
                     011E   917
                     011E   918 ;
                     011E   919 ;       Host counter block
                     011E   920 ;
                     011E   921
                     011E   922          .ALIGN  LONG
                     0120   923 GHB_AREA:                                  ; Communications area
00000180             0120   924          .BLKB   GHB$S_COMM_AREA
                     0180   925
                     0180   926          .ALIGN LONG
                     0180   927
00000184             0180   928 GHB_L_UCB0:      .BLKL   1                ; LT UCB zero address
00000188             0184   929 GHB_L_DEADLINK: .BLKL   1                ; List of dead UCBs
0000018C             0188   930 GHB_L_FFI:      .BLKL   1                ; FFI block address
00000190             018C   931 GHB_L_NULLCPU:  .BLKL   1                ; Pointer to null CPU time
00000194             0190   932 GHB_L_LASTCPU:  .BLKL   1                ; Value of last null CPU sample
00000198             0194   933 GHB_L_NULLSEC:  .BLKL   1                ; Value of latt null CPU sample on secondary
00000199             0198   934 GHB_B_STATUS:   .BLKB   1                ; Host status byte
                     0199   935          _VIELD  GHB_STS,0,<-
                     0199   936                  <MULTI,1,M>,-           ; Multicast buffer in use
                     0199   937                  <ACTIVE,1,M>,-          ; LTDRIVER is active
                     0199   938                  <SHUT,1,M>,-            ; Shutdown is in progress
                     0199   939                  <TQE,1,M>,-             ; Waiting for TQE to fire
                     0199   940                  >
0000019A             0199   941 GHB_B_INCARN:    .BLKB   1                ; Incarnation number for multicast message
0000019B             019A   942 GHB_B_OLD_CSBCNT: .BLKB 1                 ; Count of old CSBs on queue
0000019C             019B   943 GHB_B_RATE:     .BLKB   1                ; Initial rating computed from memory size
                     019C   944 ;
                     019C   945 ; Multicast data set by LATCP
                     019C   946 ;
                     019C   947          .ALIGN  LONG
0000019E             019C   948 GHB_W_MC_SIZE:  .BLKW   1        ; Size of name and ID strings
000003E8             019E   949 GHB_T_MC_DATA:  .BLKB   XMT_C_MC_TOTAL+16 ; Name and ID storage + fudge factor
                     03E8   950
                     03E8   951 ;
                     03E8   952 ; Multicast (Configuration) Message Destination Ethernet address
                     03E8   953 ;
                     03E8   954          .ALIGN  LONG
0009                 03E8   955 GHB_Q_MULTIADD: .WORD   LAT$C_MULTIADD1 ; LAT Multicast destination address
002B                 03EA   956                 .WORD   LAT$C_MULTIADD2 ;       ...
0F00                 03EC   957                 .WORD   LAT$C_MULTIADD3 ;       ...
                     03EE   958                                         ; Picks up next word also (garbage)
                     03EE   959 ;
                     03EE   960 ; Data to control multicast buffer
                     03EE   961 ;
000003F0             03EE   962 GHB_W_MULTIMR:  .BLKW   1                ; Timer for multicast transmits
000003F4             03F0   963 GHB_L_MULTIBFR: .BLKL   1                ; Address of the multicast buffer
                     03F4   964
                     03F4   965 ;
                     03F4   966 ; Host Multicast (Configuration) Message fixed portion template
                     03F4   967 ;       (Fields are order-dependent and adjacent)
                     03F4   968 ;
                     03F4   969          ASSUME  XMT_B_FLAG          EQ  XMT_T_DATA
                     03F4   970          ASSUME  XMT_B_MC_CIR_TIMER  EQ  XMT_B_FLAG+1
                     03F4   971          ASSUME  XMT_B_MC_HI_VER     EQ  XMT_B_MC_CIR_TIMER+1
                     03F4   972          ASSUME  XMT_B_MC_LO_VER     EQ  XMT_B_MC_HI_VER+1
                     03F4   973          ASSUME  XMT_B_MC_CUR_VER    EQ  XMT_B_MC_LO_VER+1
```

```
                     03F4   974          ASSUME  XMT_B_MC_CUR_ECO    EQ  XMT_B_MC_CUR_VER+1
                     03F4   975          ASSUME  XMT_B_MC_INCARN     EQ  XMT_B_MC_CUR_ECO+1
                     03F4   976          ASSUME  XMT_B_MC_CHG_FLAG   EQ  XMT_B_MC_INCARN+1
                     03F4   977          ASSUME  XMT_W_MC_MSG_SIZ    EQ  XMT_B_MC_CHG_FLAG+1
                     03F4   978          ASSUME  XMT_B_MC_SET        EQ  XMT_W_MC_MSG_SIZ+2
                     03F4   979 GHB_T_MC_MSG:                             ; Start of multicast msg fixed data
                  28 03F4   980                  .BYTE   MTYP_C_CONF@FLAG_V_MTYPE ; Message type
                  08 03F5   981                  .BYTE   <LAT$C_CIR_TIMER*95/10  ; Host preferred Circuit timer
                  05 03F6   982                  .BYTE   LAT$C_HI_VER     ; Highest LAT protocol version supported
                  05 03F7   983                  .BYTE   LAT$C_LO_VER     ; Lowest LAT protocol version supported
                  05 03F8   984                  .BYTE   LAT$C_CUR_VER    ; Current LAT protocol version
                  00 03F9   985                  .BYTE   LAT$C_CUR_ECO    ; Current LAT protocol ECO version
                  00 03FA   986                  .BYTE   0                ; Message incarnation (filled in later)
                  00 03FB   987                  .BYTE   0                ; Change flags (filled in later)
               05DC 03FC   988                  .WORD   LAT$C_MAX_MSGSIZ ; Maximum message size
                     03FE   989
                     03FE   990 ;
                     03FE   991 ; Standard Start message (fixed portion) template
                     03FE   992 ;
                     03FE   993          ASSUME  STRT_W_MSGSIZ   EQ  0
                     03FE   994          ASSUME  STRT_B_PVER     EQ  STRT_W_MSGSIZ+2
                     03FE   995          ASSUME  STRT_B_PECO     EQ  STRT_B_PVER+1
                     03FE   996          ASSUME  STRT_B_MAXSLOTS EQ  STRT_B_PECO+1
                     03FE   997          ASSUME  STRT_B_DL_BFRS  EQ  STRT_B_MAXSLOTS+1
                     03FE   998          ASSUME  STRT_B_CIR_TIMR EQ  STRT_B_DL_BFRS+1
                     03FE   999          ASSUME  STRT_B_KPA_TIMR EQ  STRT_B_CIR_TIMR+1
                     03FE  1000          ASSUME  STRT_W_FAC_NUM  EQ  STRT_B_KPA_TIMR+1
                     03FE  1001          ASSUME  STRT_W_PROD_TYP EQ  STRT_W_FAC_NUM+2
00000400             03FE  1002 GHB_W_STRT_LEN: .BLKW   1                  ; Length of start message
                     0400  1003 GHB_T_STRT_MSG:                           ; Start of start message fixed data
               05DC  0400  1004                  .WORD   LAT$C_MAX_MSGSIZ ; Maximum Ethernet frame size
                  05 0402  1005                  .BYTE   LAT$C_CUR_VER    ; Current LAT protocol version
                  00 0403  1006                  .BYTE   LAT$C_CUR_ECO    ; Current LAT ECO version
;JAY0002           00 0404    .1                 .BYTE   0                ; Maximum number of Slots per circuit
-1                00 0405  1008                  .BYTE   0                ; Number of additional buffers (remote)
00000408             0406  1009                  .BLKB   2                ; Remote parameters only
               0000  0408  1010                  .WORD   0                ; Facility number (not yet implemented)
               0003  040A  1011                  .WORD   LAT$C_VMS        ; Product type code
                     040C  1012
                     040C  1013          ASSUME  STRT_B_VAR   EQ  STRT_W_PROD_TYP+2
000004B1             040C  1014 GHB_T_STRT_VAR: .BLKB   STRT_C_VAR_LEN  ; Variable portion of start message
                     04B1  1015 ;
                     04B1  1016 ; Standard Start slot template
                     04B1  1017 ;
                     04B1  1018 GHB_T_STRT_SLOT:                         ; Start of the start slot data
                  01 04B1  1019                  .BYTE   SLT_C_SVC_INIT   ; Service class
                  FF 04B2  1020                  .BYTE   LAT$C_MAX_SLOTSIZ ; Attention slot size
                  FF 04B3  1021                  .BYTE   LAT$C_MAX_SLOTSIZ ; Data slot size
000004B6             04B4  1022                  .BLKB   2                ; Null source and destination slot names
00000005             04B6  1023          GHB_C_STRT_SLOTL = <.-GHB_T_STRT_SLOT>
```

```
                        04B6  1025          .SBTTL  Port Vector
                        04B6  1026
                        04B6  1027 ;
                        04B6  1028 ; The associated class driver uses this table to command the port driver.
                        04B6  1029 ; The address of this table is contained in the terminal ucb extension area.
                        04B6  1030 ; The offset definitons are defined by ttydefs.
                        04B6  1031 ;
                        04B6  1032
                        04B6  1033 ;
                        04B6  1034 ;       LAT specific dispatch table
                        04B6  1035 ;
                        04B6  1036 ; ORDER DEPENDENCY
                        04B6  1037 ;  The following symbol must immediately precede the vector
                        04B6  1038 ;
                        04B6  1039
                        04B6  1040          .ALIGN  LONG
                        04B8  1041 GHB_L_AREA:
000004BC                04B8  1042          .BLKL   1                                   ; Address of counter vector
                        04BC  1043 LT$VECTOR:
                        04BC  1044          $VECINI LT,LT$NULL
                        04F4  1045          $VEC    STARTIO,LT$STARTIO                  ; Start new output
                        04C0  1046          $VEC    DISCONNECT,LT$DISCONNECT            ; Hangup port
                        04C4  1047          $VEC    ABORT,LT$ABORT                      ; Abort output
                        04E0  1048          $VEC    SET_LINE,LT$FLOW_CHANGE             ; Change flow control
                        04C8  1049          $VEC    XON,LT$XON                          ; Send xon sequence
                        04D0  1050          $VEC    XOFF,LT$XOFF                        ; Send xoff sequence
                        04D4  1051 ;        $VEC    DS_SET,LT$DS_SET                    ; Dataset transitions
                        04D4  1052          $VECEND
                        04F8  1053
                        04F8  1054 LT$NULL:                                             ; Null port routine
                     05 04F8  1055          RSB
```

```
                     04F9  1057
                     04F9  1058 ;
                     04F9  1059 ; Stop reasons
                     04F9  1060 ;
                     04F9  1061
                     04F9  1062 .MACRO  $REASON RSNCD, RSNBIT, RSNSTR, ?L
                     04F9  1063         .ALIGN WORD
                     04F9  1064         .SIGNED_WORD L-.
                     04F9  1065         .BYTE RSNCD, RSNBIT
                     04F9  1066 L:      .ASCIC  /RSNSTR/
                     04F9  1067 .ENDM
                     04F9  1068
00000001             04F9  1069 LAT_C_CRSN_DONE         = 1             ; No slots in VC
00000002             04F9  1070 LAT_C_CRSN_IVMSG        = 2             ; Invalid message
00000003             04F9  1071 LAT_C_CRSN_IVSCI        = 3             ; Invalid slave conn id
00000004             04F9  1072 LAT_C_CRSN_IVMCI        = 4             ; invalid master conn id
00000005             04F9  1073 LAT_C_CRSN_DISC         = 5             ; Disconnect requested
00000006             04F9  1074 LAT_C_CRSN_PROGRESS     = 6             ; No progress being made
00000007             04F9  1075 LAT_C_CRSN_TMO          = 7             ; Timeout of circuit
00000008             04F9  1076 LAT_C_CRSN_RETRANS      = 8             ; Retransmit count exceeded
00000009             04F9  1077 LAT_C_CRSN_RESOURCE     = 9             ; No resources
0000000A             04F9  1078 LAT_C_CRSN_TIMRANG      = 10            ; Timer value out of range
                     04F9  1079
                     04F9  1080
00000000             04F9  1081 LAT_C_RSN_xxx = 0                       ; junk
                     04F9  1082
                     04F9  1083 LT_RSN_START:
                     04F9  1084         $REASON -
                     04F9  1085                 GHB$V_START,-
                     04F9  1086                 LAT_C_RSN_XXX,-
                     04F9  1087         <other than start msg with a zero src index>
                     0529  1088 LT_RSN_CSBZERO:
                     0529  1089         $REASON -
                     0529  1090                 GHB$V_CSBZERO,-
                     0529  1091                 LAT_C_RSN_XXX,-
                     0529  1092         <circuit index zero>
                     0541  1093 LT_RSN_CSBRANGE:
                     0541  1094         $REASON -
                     0541  1095                 GHB$V_CSBRANGE,-
                     0541  1096                 LAT_C_RSN_XXX,-
                     0541  1097         <circuit index out of range>
                     0561  1098 LT_RSN_CSBINVALID:
                     0561  1099         $REASON -
                     0561  1100                 GHB$V_CSBINVALID,-
                     0561  1101                 LAT_C_RSN_XXX,-
                     0561  1102         <circuit index invalid>
                     057C  1103 LT_RSN_CSBSTALE:
                     057C  1104         $REASON -
                     057C  1105                 GHB$V_CSBSTALE,-
                     057C  1106                 LAT_C_RSN_XXX,-
                     057C  1107         <circuit stale reference>
                     0598  1108 LT_RSN_HALT:
                     0598  1109         $REASON -
                     0598  1110                 GHB$V_HALT,-
                     0598  1111                 LAT_C_RSN_XXX,-
                     0598  1112         <circuit forced to halt>
                     05B3  1113 LT_RSN_INVALIDLOCID:
```

```
05B3  1114          $REASON -
05B3  1115                  GHB$V_INVALIDLOCID,-
05B3  1116                  LAT_C_RSN_XXX,-
05B3  1117          <invalid local slot id>
05CE  1118 LT_RSN_INVALIDREMID:
05CE  1119          $REASON -
05CE  1120                  GHB$V_INVALIDREMID,-
05CE  1121                  LAT_C_RSN_XXX,-
05CE  1122          <invalid remote slot id>
05E9  1123 LT_RSN_BADCREDITS:
05E9  1124          $REASON -
05E9  1125                  GHB$V_BADCREDITS,-
05E9  1126                  LAT_C_RSN_XXX,-
05E9  1127          <bad number of credits in slot>
060C  1128 LT_RSN_REPCREATE:
060C  1129          $REASON -
060C  1130                  GHB$V_REPCREATE,-
060C  1131                  LAT_C_RSN_XXX,-
060C  1132          <repeat create of slot by server>
0630  1133 LT_RSN_REPDISC:
0630  1134          $REASON -
0630  1135                  GHB$V_REPDISC,-
0630  1136                  LAT_C_RSN_XXX,-
0630  1137          <repeat disconnect of slot by server>
0658  1138
0658  1139 .ALIGN  LONG
0658  1140
```

```
                     0658  1142          .SBTTL  LT$CTRL_INIT  - Controller Initialization
                     0658  1143
                     0658  1144          .DEFAULT DISPLACEMENT, WORD
                     0658  1145 ;++
                     0658  1146 ; LT$CTRL_INIT - Initialize port driver
                     0658  1147 ;
                     0658  1148 ; Functional description:
                     0658  1149 ;
                     0658  1150 ; This routine is entered at driver load and power recovery.
                     0658  1151 ;
                     0658  1152 ; Inputs:
                     0658  1153 ;
                     0658  1154 ;       R6 = DDB address
                     0658  1155 ;       R8 = CRB address
                     0658  1156 ;
                     0658  1157 ; Outputs:
                     0658  1158 ;
                     0658  1159 ;       R0 through r3 are modified.
                     0658  1160 ;
                     0658  1161 ; Implicit inputs:
                     0658  1162 ;
                     0658  1163 ;       IPL = IPL$_POWER
                     0658  1164 ;
                     0658  1165 ;--
                     0658  1166 LT$CTRL_INIT:                           ; Initialize controller
                     0658  1167 ;
                     0658  1168 ; Save the address of unit zero for cloning ucbs, if its here now.
                     0658  1169 ;
   51    04 A6   D0  0658  1170          MOVL    DDB$L_UCB(R6), R1       ; Its in the device data block
         05      13  065C  1171          BEQL    5$                      ; Do not store a zero
 FB1D CF   51    D0  065E  1172          MOVL    R1, GHB_L_UCB0          ; Save its address
                     0663  1173 5$:
 0000006C'GF     D0  0663  1174          MOVL    G^SCH$GL_NULLPCB+PCB$L_PHD,- ; Get address of NULL PHD
     FB20 CF         0669  1175                    GHB_L_NULLCPU
 FB1B CF   38    C0  066C  1176          ADDL    #PHD$L_CPUTIM,GHB_L_NULLCPU ; Get offset to "idle time"
                     0671  1177
                     0671  1178 ;
                     0671  1179 ; Attempt to relocate terminal class vector table in case we are the
                     0671  1180 ; first port driver to load.
                     0671  1181 ;
                     0671  1182
 00000000'GF     D0  0671  1183          MOVL    G^TTY$GL_DPT,R1         ; Get terminal class driver DPT address
         51          0677
   50    1E A1   3C  0678  1184          MOVZWL  DPT$W_VECTOR(R1),R0     ; Get offset to class vector
50   51    50    C1  067C  1185          ADDL3   R0,R1,R0                ; Calculate virtual address
         60    D5  0680  1186 10$:     TSTL    (R0)                    ; Already relocated or done?
         05    15  0682  1187          BLEQ    20$                     ; Br if yes
      80 51    C0  0684  1188          ADDL    R1,(R0)+                ; Else, add bias
         F7    11  0687  1189          BRB     10$                     ; Loop till done
                     0689  1190 ;
                     0689  1191 ; Relocate port vector table
                     0689  1192 ;
                     0689  1193 20$:
51   0000'CF   DE  0689  1194          MOVAL   LT$DPT,R1               ; Get our DPT address
50   FE2A CF   DE  068E  1195          MOVAL   LT$VECTOR,R0            ; Get address of our dispatch vector
                     0693  1196 30$:
         60    D5  0693  1197          TSTL    (R0)                    ; Already relocated or done?
```

```
                05   15  0695  1198                BLEQ    40$                             ; Br if yes, powerfail
              80 51   C0  0697  1199                ADDL    R1,(R0)+                        ; Else, add bias
                 F7   11  069A  1200                BRB     30$                             ; Loop til done
                      069C  1201 40$:
          075D'CF   9E  069C  1202                MOVAB   LT$STRENTRY,-                   ; Set address of start entry point
             FA8D CF        06A0  1203                        GHB$L_STRENTRY+GHB_AREA ;
          0835'CF   9E  06A3  1204                MOVAB   LT$SHUTENTRY,-                  ; Set address shutdown entry point
             FA82 CF        06A7  1205                        GHB$L_SHUTENTRY+GHB_AREA;
          0965'CF   9E  06AA  1206                MOVAB   LT$SETENTRY,-                   ; Set address of multicast set entry
             FA73 CF        06AE  1207                        GHB$L_SETENTRY+GHB_AREA ;
             FA6B CF   9E  06B1  1208                MOVAB   GHB_AREA,-                      ; Save address of the common area
             FE00 CF        06B5  1209                        GHB_L_AREA                      ;   so we can find them from LATCP
    00000000'GF   90  06B8  1210                MOVB    G^EXE$GQ_SYSTIME,-              ; Get a random number
             FAD8 CF        06BE  1211                        GHB_B_INCARN                    ;  and save for the message seed value
                            06C1  1212
             FAD9 CF   9E  06C1  1213                MOVAB   GHB_T_MC_DATA,-                 ; Address of multicast data area
             FA74 CF        06C5  1214                        GHB$L_NODE+GHB_AREA
                            06C8  1215
                            06C8  1216                ASSUME  GHB$B_LATVERSION  EQ  GHB$W_VMSVERSION+2
                            06C8  1217                ASSUME  GHB$B_LATECO      EQ  GHB$B_LATVERSION+1
    00050004 8F   D0  06C8  1218                MOVL    #<<LAT$C_CUR_ECO>@8! -          ; Store current ECO
             FA4F CF        06CE
                            06D1  1219                         LAT$C_CUR_VER>@16! -           ;           and VERsion
                            06D1  1220                         LAT$C_VMS_VER, -               ;         and VMS VERsion
                            06D1  1221                        GHB$W_VMSVERSION+GHB_AREA
                 0A   90  06D1  1222                MOVB    #LAT$C_KEEP_CSB,-               ; Set number of CSBs to keep around
             FAC4 CF        06D3  1223                        GHB_B_OLD_CSBCNT
             F95E CF   9E  06D6  1224                MOVAB   GHB_L_CSBTABLE,-                ; Store address of CSB table
             FA63 CF        06DA  1225                        GHB$L_CSBLST+GHB_AREA
          50 FA63 CF   9E  06DD  1226                MOVAB   GHB$Q_OLD_CSBS+GHB_AREA,R0 ; Get address of old CSB entries
                    60   D5  06E2  1227                TSTL    (R0)                            ; Already initialized?
                    07   12  06E4  1228                BNEQ    50$                             ; Br if yes, skip next section
                 60 50   D0  06E6  1229                MOVL    R0,(R0)                         ; Initialize listhead
              04 A0 50   D0  06E9  1230                MOVL    R0,4(R0)                        ;       ...
                            06ED  1231
                            06ED  1232 50$:   ;
;JAY0002                                     06ED    .1        ; Set up the maximum circuit session limit value for this processor
;JAY0002                                     06ED    .2        ;
;JAY0002          00000000'GF   9A  06ED    .3                MOVZBL  G^EXE$GB_CPUTYPE,R0             ; Get CPU type code
                            50  06F3
;JAY0002              50   10   91  06F4    .4                CMPB    #MAX_PR_SID,R0                  ; Within range?
;JAY0002                    08   1E  06F7    .5                BGEQU   55$                             ; Branch if so
;JAY0002                 40 8F   90  06F9    .6                MOVB    #LAT$C_MAX_SLOTS, -             ; Else load default maximum
                        FA0D CF     06FC
;JAY0002                             06FF    .7                        W^GHB_B_MAX_SLOTS               ;
;JAY0002                    08   11  06FF    .8                BRB     60$                             ; And branch
;JAY0002            FA07 CF40   90  0701    .9 55$:           MOVB    GHB_T_MAX_SLOT_TABLE[R0], - ; Load maximum session count
                        FA03 CF     0706
;JAY0002                             0709   .10                        W^GHB_B_MAX_SLOTS               ;
;JAY0002                             0709   .11        ;
;JAY0002                             0709   .12        ;  Setup the initial rating value, based on memory size
;JAY0002                             0709   .13        ;
;JAY0002          00000000'GF   D0  0709   .14 60$:           MOVL    G^MMG$GL_MAXPFN,R0      ; Get the max PFN count
                            50  070F
-3                  00002000 8F   C6  0710  1236                DIVL    #8192,R0                ; Weight by 4MB chunks
                            50  0716
                 50 08      50   C3  0717  1237                SUBL3   R0,#8,R0                ; Compute for up to 32 MB systems
```

```
                02   18  071B  1238          BGEQ    70$                     ; Br if result okay
                50   D4  071D  1239          CLRL    R0                      ; Else, more than 32 MB - impressive!
FF 8F           50   83  071F  1240 70$:      SUBB3   R0,#255,GHB_B_RATE      ; Store initial rating
   FA75 CF               0723
                05   0726  1241          RSB
                     0727  1242
                     0727  1243          .IF DEFINED LT_HISTORY
```

```
                     0727  1245          .SBTTL  LT$HISTORY  - Save history in the buffer
                     0727  1246 ;++
                     0727  1247 ; LT$HISTORY - Save history in the buffer
                     0727  1248 ;
                     0727  1249 ;   Save history on transmits and receives.
                     0727  1250 ;   The history buffer is defined as follows:
                     0727  1251 ;
                     0727  1252 ;                   +-------------------------+
                     0727  1253 ;                   |  Pointer to next slot   |
                     0727  1254 ;                   |-------------------------|
                     0727  1255 ;                   | Pointer to end of buffer|
                     0727  1256 ;                   |-------------------------|
                     0727  1257 ;                   |spare!type! buffer size  |
                     0727  1258 ;                   |-------------------------|
                     0727  1259 ;                   |       ... data ...      |
                     0727  1260 ;                   |                         |
                     0727  1261 ;                   |                         |
                     0727  1262 ;                   +-------------------------+
                     0727  1263 ;
                     0727  1264 ; Inputs:
                     0727  1265 ;       R0 -> data to save
                     0727  1266 ;
                     0727  1267 ; Outputs:
                     0727  1268 ;       R1 is clobbered
                     0727  1269 ;--
                     0727  1270
           00000020  0727  1271          SLOT_SIZE = 32                  ; Size of each history slot
                     0727  1272
                     0727  1273 LT$HISTORY:
 51    F9FD CF    D0  0727  1274          MOVL    GHB$L_HISTORY+GHB_AREA,R1 ; Point to history buffer
              2E    13  072C  1275          BEQL    90$                     ; Br if none
                    072E  1276                                            ;
              50    DD  072E  1277          PUSHL   R0                      ; Save R0
              20    C1  0730  1278          ADDL3   #SLOT_SIZE,-            ; Compute end address for this slot
        50    61        0732  1279                    HBF_L_NEXT(R1),R0
  04  A1    50    D1  0734  1280          CMPL    R0,HBF_L_BUFEND(R1)     ; Are we past the end?
              04    1B  0738  1281          BLEQU   30$                     ; Br if no, continue
           0C A1    9E  073A  1282          MOVAB   HBF_Z_DATA(R1),-        ; Else, back to beginning
              61        073D  1283                  HBF_L_NEXT(R1)
        51    61    D0  073E  1284 30$:     MOVL    HBF_L_NEXT(R1),R1       ; Get address of next available slot
 00000000'GF    7D  0741  1285          MOVQ    G^EXE$GQ_SYSTIME,(R1)+  ; Store time in first quadword
              81        0747
        50    6E    D0  0748  1286          MOVL    (SP),R0                 ; Get back buffer address
        81    80    7D  074B  1287          MOVQ    (R0)+, (R1)+            ; Save the data
        81    80    7D  074E  1288          MOVQ    (R0)+, (R1)+            ; Save the data
        81    80    7D  0751  1289          MOVQ    (R0)+, (R1)+            ; Save the data
                    0754  1290          ASSUME  HBF_L_NEXT EQ 0
F9CF DF    51    D0  0754  1291          MOVL    R1,@GHB$L_HISTORY+GHB_AREA ; Save address of next slot
              50 8ED0  0759  1292          POPL    R0                      ; Restore R0
                    075C  1293                                            ;
                 05  075C  1294 90$:     RSB
                    075D  1295          .ENDC   ;; DEFINED LT_HISTORY
```

```
                        075D  1297          .SBTTL  LT$STRENTRY  - Entry to start driver
                        075D  1298 ;++
                        075D  1299 ; LT$STRENTRY - Start LTDRIVER entry point
                        075D  1300 ;
                        075D  1301 ; Functional description:
                        075D  1302 ;
                        075D  1303 ;       This entry is called by the startup program to initialize the
                        075D  1304 ;       driver.  Here we save the datalink UCB address, allocate the CSB's
                        075D  1305 ;       and start the first receive.
                        075D  1306 ;
                        075D  1307 ; Inputs:
                        075D  1308 ;       R5 = UCB address of datalink
                        075D  1309 ;
                        075D  1310 ; Implicit inputs:
                        075D  1311 ;
                        075D  1312 ;       Called in user context.
                        075D  1313 ;
                        075D  1314 ; Outputs:
                        075D  1315 ;       R0 = success or failure
                        075D  1316 ;--
                        075D  1317
                        075D  1318
                        075D  1319 LT$STRENTRY:
                        075D  1320          DSBINT  #LAT$C_IPL                  ; Raise IPL to prevent interruption
F9CC CF    55    D0     0763  1321          MOVL    R5,GHB$L_UCB+GHB_AREA       ; Save the datalink UCB address
                        0768  1322
59   F8CC CF    9E     0768  1323          MOVAB   GHB_L_CSBTABLE,R9           ; Get CSB table base address
     5A   20    9A     076D  1324          MOVZBL  #LAT$C_MAX_CSBS,R10         ; Get table length in longwords
          89    D4     0770  1325 20$:     CLRL    (R9)+                       ; Zero the whole table
        FB 5A    F5     0772  1326          SOBGTR  R10,20$                     ; Do whole table
                        0775  1327          ;
                        0775  1328          ;  Save the last CPU idle value
                        0775  1329          ;
     FA13 DF     D0     0775  1330          MOVL    @GHB_L_NULLCPU,-            ; Save last idle value
     FA14 CF            0779  1331            GHB_L_LASTCPU
00000000'GF     D0     077C  1332          MOVL    G^EXE$GL_MP,R0              ; Get ptr to MP code
          50            0782
          07    13     0783  1333          BEQL    40$                         ; Br if not there
     0000'C0    D0     0785  1334          MOVL    MPS$GL_NULLCPU(R0),-        ; Else, get Secondary null time
     FA08 CF            0789  1335            GHB_L_NULLSEC
                        078C  1336 40$:     ;
                        078C  1337          ;  See if the datalink supports the fast interface and if so
                        078C  1338          ;  then allocate an FFI block and initialize that interface.
                        078C  1339          ;
          50    D4     078C  1340          CLRL    R0                          ; Assume error
  51    24 A5    D0     078E  1341          MOVL    UCB$L_CRB(R5), R1           ; Get address of datalink's CRB
  51    40 A1    D0     0792  1342          MOVL    CRB$L_INTD+VEC$L_START(R1), R1 ; Get address of FFI INIT routine
          36    13     0796  1343          BEQL    70$                         ; Br if not present
        0045    30     0798  1344          BSBW    LT$GETFFI                   ; Else, create and initialize FFI
       30 50    E9     079B  1345          BLBC    R0,70$                      ; Br if error
F9E5 CF    54    D0     079E  1346          MOVL    R4,GHB_L_FFI                ; Else, save FFI block address
                        07A3  1347
                        07A3  1348          CLRBIT  #GHB_STS_V_MULTI,-          ; Clear the multicast busy flag
                        07A3  1349            GHB_B_STATUS
     FC43 CF    D4     07A9  1350          CLRL    GHB_L_MULTIBFR              ; No multicast buffer is possible
        01B5    30     07AD  1351          BSBW    LT$SETENTRY                 ; Setup multicast message
                        07B0  1352
```

```
                                07B0  1353          MOVL    GHB_L_UCB0, R0          ; Set unit seed in the ucb
                             17 13  07B5  1354          BEQL    70$                     ; No ucb0, bad news
                             60 B4  07B7  1355          CLRW    UCB$W_MB_SEED(R0)       ; Start with unit 1
                           105C 30  07B9  1356          BSBW    LT$SET_TIMER            ; Start timer, so we always have a tick
                                07BC  1357
50      F9CC CF                 DO  07BC  1358          MOVL    GHB$L_UCB+GHB_AREA, R5  ; Get datalink UCB address
                             5C A5  B6  07C1  1359          INCW    UCB$W_REFC(R5)          ; One more ref to keep it around for us
                           8000 8F  A8  07C4  1360          BISW    #^x8000,UCB$W_DEVSTS(R5); Allow unit to be re-started
                             68 A5      07C8
                                07CA  1361                                              ;   automatically by datalink
50                00'8F         9A  07CA  1362          MOVZBL  #SS$_NORMAL, R0         ; Return success
                             0B 50  E8  07CE  1363 70$:    BLBS    R0, 80$                 ; Br if okay
                          F95F CF   D4  07D1  1364          CLRL    GHB$L_UCB+GHB_AREA      ; Else, forget about datalink UCB
                                5E  10  07D5  1365          BSBB    LT$SHUTENTRY            ; Clean it all up
50                0000'8F         3C  07D7  1366          MOVZWL  #SS$_INSFMEM, R0        ; Return not enough memory
                                07DC  1367 80$:    ENBINT                          ; Restore IPL
                                05  07DF  1368          RSB
                                07E0  1369
```

```
                        07E0  1371          .SBTTL  LT$GETFFI - Get the FFI block and initialize FFI interface
                        07E0  1372 ;++
                        07E0  1373 ; LT$GETFFI - Get FFI block
                        07E0  1374 ;
                        07E0  1375 ; Functional description:
                        07E0  1376 ;
                        07E0  1377 ;       Initialize the FFI interface.
                        07E0  1378 ;
                        07E0  1379 ; Inputs:
                        07E0  1380 ;
                        07E0  1381 ;       R1 = Address of FFI INIT routine in datalink driver
                        07E0  1382 ;       R5 = UCB address of datalink
                        07E0  1383 ;
                        07E0  1384 ; Outputs:
                        07E0  1385 ;
                        07E0  1386 ;       R0 = Status
                        07E0  1387 ;       R1-R4 are destroyed.
                        07E0  1388 ;
                        07E0  1389 ;--
                        07E0  1390
                        07E0  1391 LT$GETFFI:
            51    DD    07E0  1392          PUSHL   R1                              ; Save init routine address
        51  3E    9A    07E2  1393          MOVZBL  #FFI$C_LENGTH,R1                ; Set length of buffer to allocate
00000000'GF   16    07E5  1394          JSB     G^EXE$ALONONPAGED               ; Try to allocate an FFI block
          31  50    E9    07EB  1395          BLBC    R0,30$                          ; Br if error
          54  52    D0    07EE  1396          MOVL    R2,R4                           ; Copy FFI block address
                        07F1  1397          ASSUME  FFI$L_FL EQ 0
                        07F1  1398          ASSUME  FFI$L_BL EQ FFI$L_FL+4
                        07F1  1399          ASSUME  FFI$W_SIZE EQ FFI$L_BL+4
                        07F1  1400          ASSUME  FFI$B_TYPE EQ FFI$W_SIZE+2
                        07F1  1401          ASSUME  FFI$B_SPARE EQ FFI$B_TYPE+1
            82    7C    07F1  1402          CLRQ    (R2)+                           ; Clear link pointers
001?^^3E    8F    D0    07F3  1403          MOVL    #<DYN$C_BUFIO@16>!FFI$C_LENGTH,(R2)+ ; Store block size & type
            82
                        07F9
                        07FA  1404          ASSUME  FFI$L_CTX_DL EQ FFI$B_SPARE+1
                        07FA  1405          ASSUME  FFI$L_XMIT EQ FFI$L_CTX_DL+4
                        07FA  1406          ASSUME  FFI$L_XMIT_DONE EQ FFI$L_XMIT+4
                        07FA  1407          ASSUME  FFI$L_RECV_DONE EQ FFI$L_XMIT_DONE+4
                        07FA  1408          ASSUME  FFI$L_ERROR EQ FFI$L_RECV_DONE+4
                        07FA  1409          ASSUME  FFI$L_SHUT_DONE EQ FFI$L_ERROR+4
            82    7C    07FA  1410          CLRQ    (R2)+                           ; Zero CTX & XMIT routine address
    62  1B54'CF    9E    07FC  1411          MOVAB   W^LT$XMT_FFIDONE,(R2)+          ; Store XMIT Complete routine address
    82  0D1C'CF    9E    0801  1412          MOVAB   W^LT$FFI_RCV_MSG,(R2)+          ; Store RECV Complete routine address
    82  0964'CF    9E    0806  1413          MOVAB   W^LT$ASYNCERR,(R2)+             ; Store address of ERROR handler
    82  0953'CF    9E    080B  1414          MOVAB   W^LT$SHUT_DONE,(R2)+            ; Store address of SHUTDOWN handler
                        0810  1415          ASSUME  FFI$L_SPARE0 EQ FFI$L_SHUT_DONE+4
                        0810  1416          ASSUME  FFI$L_SPARE1 EQ FFI$L_SPARE0+4
                        0810  1417          ASSUME  FFI$L_SPARE2 EQ FFI$L_SPARE1+4
                        0810  1418          ASSUME  FFI$L_SPARE3 EQ FFI$L_SPARE2+4
            82    7C    0810  1419          CLRQ    (R2)+                           ; Not used
            82    7C    0812  1420          CLRQ    (R2)+
                        0814  1421          ASSUME  FFI$L_DL_UCB EQ FFI$L_SPARE3+4
                        0814  1422          ASSUME  FFI$L_PID EQ FFI$L_DL_UCB+4
                        0814  1423          ASSUME  FFI$W_CHAN EQ FFI$C_PID+4
        82  55    D0    0814  1424          MOVL    R5,(R2)+                        ; Store datalink UCB address
            82    D4    0817  1425          CLRL    (R2)+                           ; Zero PID
            82    B4    0819  1426          CLRW    (R2)+                           ; Zero CHAN
```

```
50   00'8F   9A   081B  1427          MOVZBL  #SS$_NORMAL,R0          ; Return success
       51 8ED0   081F  1428 30$:       POPL    R1                      ; Restore R1
     0F 50     E9   0822  1429          BLBC    R0,90$                  ; Br if allocation error
       53      D4   0825  1430          CLRL    R3                      ; No parameters
       61      16   0827  1431          JSB     (R1)                    ; Else, call datalink init routine
     08 50     E8   0829  1432          BLBS    R0,90$                  ; Br if success
   50   54      D0   082C  1433          MOVL    R4,R0                   ; Else, copy FFI block address
     13A7      30   082F  1434          BSBW    LT$DEALPOOL             ; Deallocate the FFI block
       50      D4   0832  1435          CLRL    R0                      ; Return failure
               05   0834  1436 90$:     RSB
```

```
                     0835  1438          .SBTTL  LT$SHUTENTRY  - Shutdown entry point
                     0835  1439 ;++
                     0835  1440 ; LT$SHUTENTRY - Shutdown LTDRIVER entry point
                     0835  1441 ;
                     0835  1442 ; Functional description:
                     0835  1443 ;
                     0835  1444 ;       Shutdown things when we are done.
                     0835  1445 ;
                     0835  1446 ; Inputs:
                     0835  1447 ;       none
                     0835  1448 ;
                     0835  1449 ; Implicit inputs:
                     0835  1450 ;
                     0835  1451 ;       Called in user context.
                     0835  1452 ;
                     0835  1453 ; Outputs:
                     0835  1454 ;       none
                     0835  1455 ;--
                     0835  1456
                     0835  1457 LT$SHUTENTRY:
                     0835  1458          DSBINT  #LAT$C_IPL              ; Sync with rest of things
                     083B  1459          SETBIT  #GHB_STS_V_SHUT,-       ; Indicate that shutdown has been
                     083B  1460                   GHB_B_STATUS           ;    requested
                     0841  1461
                     0841  1462 ;
                     0841  1463 ;       Deallocate and shut down all the circuits
                     0841  1464 ;
59   F7F3 CF    DE   0841  1465          MOVAL   GHB_L_CSBTABLE, R9      ; Table of CSB addresses
     5A   20    9A   0846  1466          MOVZBL  #LAT$C_MAX_CSBS, R10    ; and its length
     58   89    D0   0849  1467 40$:     MOVL    (R9)+, R8               ; Get a CSB address
          03    13   084C  1468          BEQL    60$                     ; No CSB address here
        0ED9    30   084E  1469          BSBW    LT$CIRCDEAD             ; Kill all processes on this one
        F5 5A   F5   0851  1470 60$:     SOBGTR  R10, 40$                ; Til end of table
                     0854  1471
                     0854  1472 ;
                     0854  1473 ; We can reduce our interest in the datalink UCB here early because
                     0854  1474 ; the LAT control program still has a channel to the UCB and is still
                     0854  1475 ; active.   The UCB will disappear when the LATCP deassigns its last
                     0854  1476 ; channel to it.
                     0854  1477 ;
                     0854  1478
59   F8DC CF    D0   0854  1479          MOVL    GHB$L_UCB+GHB_AREA, R9  ; Get datalink UCB and reduce our interest
          19    13   0859  1480          BEQL    80$                     ; No ucb here
     8000 8F    AA   085B  1481          BICW    #^x8000,UCB$W_DEVSTS(R9); Don't want datalink to re-start
       68 A9         085F
                     0861  1482                                          ;   any more!
       5C A9    B7   0861  1483          DECW    UCB$W_REFC(R9)          ; Let datalink UCB go away.
     F8CC CF    D4   0864  1484          CLRL    GHB$L_UCB+GHB_AREA      ; Forget we ever had it
                     0868  1485                                          ; This prevents more circuits from
                     0868  1486                                          ; being accepted.
                     0868  1487          ;
                     0868  1488          ;  Wait for any datalink re-start timer to expire. Spin on
                     0868  1489          ;  the INTERLOCK bit.
                     0868  1490          ;
                     0868  1491          $SETIPL #IPL$_ASTDEL            ; Prevent us from being deleted,
                     086B  1492                                          ;  but allow scheduling activity
04 68 A9    0E   E1  086B  1493 70$:     BBC     #14,UCB$W_DEVSTS(R9),80$ ; Proceed if UCB can evaporate
```

```
                70   10  0870  1494          BSBB    BUILD_TQE               ; Wait around for a while
                F7   11  0872  1495          BRB     70$                     ; Try again
                         0874  1496
                         0874  1497 80$:
                         0874  1498 ;
                         0874  1499 ;       Now we must wait for all the CSBs to disappear.  We must not leave
                         0874  1500 ;       here until all the processes have run down and the CSBs are gone
                         0874  1501 ;       since latstart can be used again very quickly and that could cause
                         0874  1502 ;       CSBs to be allocated again.  If processes don't run down we will
                         0874  1503 ;       be stuck here until they do.
                         0874  1504 ;
                         0874  1505          SETIPL  #IPL$_ASTDEL            ; Prevent us from being deleted,
                         0877  1506                                          ;  but allow scheduling activity
                02   11  0877  1507          BRB     110$                    ; See if CSB's are all gone
                         0879  1508          ;
                         0879  1509          ;  Now delay while we wait for CSB's to run down
                         0879  1510          ;
                67   10  0879  1511 100$:    BSBB    BUILD_TQE               ; Build TQE to wait on and wait
                         087B  1512
            59  F7B9 CF   DE  087B  1513 110$:    MOVAL   GHB_L_CSBTABLE,R9       ; Table of CSB addresses
                5A   20   9A  0880  1514          MOVZBL  #LAT$C_MAX_CSBS, R10    ;   and its length
                58   89   D0  0883  1515 120$:    MOVL    (R9)+, R8               ; Get a CSB address
                     F1   12  0886  1516          BNEQ    100$                    ; Just spin here looking
                   F8 5A  F5  0888  1517          SOBGTR  R10, 120$               ; Til end of table
                         088B  1518          ;
                         088B  1519          ;  Now wait for all the UCBs to dissappear except unit 0
                         088B  1520          ;
                02   11  088B  1521          BRB     160$                    ; Check if we're all done
                         088D  1522
                53   10  088D  1523 140$:    BSBB    BUILD_TQE               ; Build TQE to wait on and wait
                         088F  1524          ;
                         088F  1525          ;  Now delay while we wait for terminal UCB's to run down
                         088F  1526          ;
            50  F8ED CF   D0  088F  1527 160$:    MOVL    GHB_L_UCB0,R0           ; Get the LT UCB address
                     0A   13  0894  1528          BEQL    180$                    ; ???
                51   24 A0   D0  0896  1529          MOVL    UCB$L_CRB(R0),R1        ; Get the CRB address
                01   0C A1   B1  089A  1530          CMPW    CRB$W_REFC(R1),#1       ; All done ?
                     ED   12  089F  1531          BNEQ    140$                    ; Br if no
                         08A0  1532 180$:    ;
                         08A0  1533          ;  Reset completion routines for datalink's FFI interface
                         08A0  1534          ;
                         08A0  1535          SETIPL  #LAT$C_IPL              ; Sync access to rest of driver
            55  F8E1 CF   D0  08A3  1536          MOVL    GHB_L_FFI, R5           ; Get FFI block address
                     0C   13  08A8  1537          BEQL    200$                    ; Br if none
                1BD8'CF   9E  08AA  1538          MOVAB   W^LT$DROPCXB,-          ; Drop all RECEIVE buffers
                  18 A5       08AE  1539                  FFI$L_RECV_DONE(R5)
                1BCF'CF   9E  08B0  1540          MOVAB   W^LT$FFIPOSTDONE,-      ; Deallocate all TRANSMIT CXBs
                  14 A5       08B4  1541                  FFI$L_XMIT_DONE(R5)
                         08B6  1542 200$:    ;
                         08B6  1543          ;  Deallocate the multicast buffer
                         08B6  1544          ;
            50  FB36 CF   D0  08B6  1545          MOVL    GHB_L_MULTIBFR,R0       ; Get the multicast buffer
                     08   13  08BB  1546          BEQL    220$                    ; Br if none
                         08BD  1547          ASSUME  GHB_STS_V_MULTI EQ 0
                03 F8D7 CF   E8  08BD  1548          BLBS    GHB_B_STATUS,220$       ; Br if multicast buffer is active
                   1314      30  08C2  1549          BSBW    LT$DEALPOOL             ; Else, deallocate the multicast buffer
                FB27 CF      D4  08C5  1550 220$:    CLRL    GHB_L_MULTIBFR          ; Forget we had a multicast buffer
```

```
                     08C9  1551          ;
                     08C9  1552          ;  Flush all CSBs on the old CSB display queue
                     08C9  1553          ;
50   F877 DF   0F    08C9  1554 240$:    REMQUE  @GHB$Q_OLD_CSBS+GHB_AREA,R0 ; Get CSB address
          05   1D    08CE  1555          BVS     300$                        ; Br if none
        1306   30    08D0  1556          BSBW    LT$DEALPOOL                 ; Else, deallocate the CSB
          F4   11    08D3  1557          BRB     240$                        ; Loop til done
          0A   90    08D5  1558 300$:    MOVB    #LAT$C_KEEP_CSB,-           ; Reset number of CSBs to keep around
        F8C0 CF      08D7  1559                  GHB_B_OLD_CSBCNT
                     08DA  1560          ;
                     08DA  1561          ;  Mark LTDRIVER as inactive
                     08DA  1562          ;
        F8BA CF   94 08DA  1563          CLRB    GHB_B_STATUS                ; Clear ACTIVE flag to stop timer
                     08DE  1564                                              ;   clear SHUTdown flag
                     08DE  1565                                              ;   clear MULTIcast buffer flag
                     08DE  1566          ENBINT
                  05 08E1  1567          RSB
                     08E2  1568
```

```
                        08E2  1570          .SBTTL  TQE TIMER BUILDING AND WAITING ROUTINES
                        08E2  1571 ;++
                        08E2  1572 ; BUILD_TQE - Build a TQE
                        08E2  1573 ;
                        08E2  1574 ; Inputs:
                        08E2  1575 ;       IPL = ASTDEL
                        08E2  1576 ;
                        08E2  1577 ; Implicit inputs:
                        08E2  1578 ;
                        08E2  1579 ;       Called in user context
                        08E2  1580 ;
                        08E2  1581 ; Outputs:
                        08E2  1582 ;       R0-R5 are destroyed
                        08E2  1583 ;--
                        08E2  1584
                        08E2  1585 BUILD_TQE:
00000000'GF   16        08E2  1586          JSB     G^EXE$ALLOCTQE          ; Allocate a TQE (returns IPL = ASTDEL)
      4A 50   E9        08E8  1587          BLBC    R0,90$                  ; Br if allocation failure
   0B A2  01  90        08EB  1588          MOVB    #TQE$C_SSSNGL,TQE$B_RQTYPE(R2) ; Indicate single shot request
0C A2  36'AF  9E        08EF  1589          MOVAB   B^SHUT_TIMER,TQE$L_FPC(R2) ; Set wake up routine address
         7E   DC        08F4  1590          MOVPSL  -(SP)                   ; Push current PSL to resume process
                        08F6  1591          ASSUME  IPL$_SYNCH EQ IPL$_TIMER
                        08F6  1592          SETIPL  #IPL$_SYNCH             ; Sync access to system data base
00000000'GF   7D        08F9  1593          MOVQ    G^EXE$GQ_SYSTIME,R0     ; Get current time
         50             08FF
004C4B40 8F   C0        0900  1594          ADDL    #LAT$C_SHUTDELAY,R0     ; Calculate expiration time
         50             0906
      51  00   D8        0907  1595          ADWC    #0,R1                   ; Add carry to top half
      55  52   D0        090A  1596          MOVL    R2,R5                   ; Copy TQE address
00000000'GF   16        090D  1597          JSB     G^EXE$INSTIMQ           ; Insert TQE on timer queue
                        0913  1598          CLRBIT  #GHB_STS_V_TQE,-        ; Indicate TQE not complete - yet
                        0913  1599                   GHB_B_STATUS
         08   11        0919  1600          BRB     60$                     ; Wait for TQE
         03   E0        091B  1601 40$:     BBS     #GHB_STS_V_TQE,-        ; Br if TQE is complete
14 F878 CF              091D  1602                   GHB_B_STATUS,90$
         7E   DC        0921  1603          MOVPSL  -(SP)                   ; Push current PSL to resume process
00000000'GF   D0        0923  1604 60$:     MOVL    G^SCH$GL_CURPCB,R4      ; Get current PCB address
         54             0929
      50  01   9A        092A  1605          MOVZBL  #RSN$_ASTWAIT,R0        ; Set waiting resource
00000000'GF   16        092D  1606          JSB     G^SCH$RWAIT             ; Make process wait for resource
         E6   11        0933  1607          BRB     40$                     ; Check if TQE expired
              05        0935  1608 90$:     RSB                             ; Return to caller
                        0936  1609
                        0936  1610 ;++
                        0936  1611 ; SHUT_TIMER
                        0936  1612 ;
                        0936  1613 ;       This routine wakes up the waiting process
                        0936  1614 ;
                        0936  1615 ; Inputs:
                        0936  1616 ;       R4 = PCB address
                        0936  1617 ;       R5 = TQE address
                        0936  1618 ;
                        0936  1619 ;       IPL = IPL$_TIMER
                        0936  1620 ;
                        0936  1621 ; Outputs:
                        0936  1622 ;       R0 is destroyed
                        0936  1623 ;       R5 = TQE address of system non-repeating TQE
```

```
                     0936  1624 ;--
                     0936  1625
                     0936  1626 SHUT_TIMER:
                     0936  1627          SETBIT  #GHB_STS_V_TQE,-          ; Indicate TQE has completed
                     0936  1628                  GHB_B_STATUS
          50    01   9A 093C  1629          MOVZBL  #RSN$_ASTWAIT,R0          ; Set waiting resource
00000000'GF   16   9F 093F  1630          JSB     G^SCH$RAVAIL              ; Mark resource available
          50    55   D0 0945  1631          MOVL    R5,R0                     ; Copy TQE address
                128E 30 0948  1632          BSBW    LT$DEALPOOL               ; Deallocate the TQE
00000000'GF   DE   9F 094B  1633          MOVAL   G^EXE$AL_TQENOREPT,R5     ; Setup SYSTEM no repeat TQE address
          55        0951
                     05 0952  1634          RSB                               ; Return to caller
```

```
                     0953  1636          .SBTTL  LT$SHUT_DONE - Datalink shutdown complete entry
                     0953  1637 ;++
                     0953  1638 ; LT$SHUT_DONE - Datalink shutdown complete entry point
                     0953  1639 ;
                     0953  1640 ; Functional description:
                     0953  1641 ;
                     0953  1642 ;       Deallocate the FFI block when datalink is all done.
                     0953  1643 ;
                     0953  1644 ; inputs:
                     0953  1645 ;       R4 = FFI block address
                     0953  1646 ;
                     0953  1647 ; outputs:
                     0953  1648 ;       R0,R1,R4 are destroyed
                     0953  1649 ;       All other registers must be preserved
                     0953  1650 ;--
                     0953  1651
                     0953  1652 LT$SHUT_DONE::
          2C   BB    0953  1653          PUSHR   #^M<R2,R3,R5>           ; Save registers
50  F82F CF   D0    0955  1654          MOVL    GHB_L_FFI,R0            ; Get FFI block address
     F82A CF   D4    095A  1655          CLRL    GHB_L_FFI               ; Zero pointer to FFI block
        1278   30    095E  1656          BSBW    LT$DEALPOOL             ; Deallocate the FFI block
          2C   BA    0961  1657          POPR    #^M<R2,R3,R5>           ; Restore registers
               05    0963  1658          RSB
```

```
                        0964  1660          .SBTTL  LT$ASYNCERR - Process async errors from datalink
                        0964  1661 ;++
                        0964  1662 ; LT$ASYNCERR - Process async errors from datalink
                        0964  1663 ;
                        0964  1664 ; Functional description:
                        0964  1665 ;
                        0964  1666 ;       Do Nothing.
                        0964  1667 ;
                        0964  1668 ;--
                        0964  1669 LT$ASYNCERR:
05                      0964  1670          RSB
```

```
                                                J 1
                     0965  1672                .SBTTL  LT$SETENTRY  - Set a new multicast message
                     0965  1673 ;++
                     0965  1674 ; LT$SETENTRY - Set new multicast message
                     0965  1675 ;
                     0965  1676 ; Functional description:
                     0965  1677 ;
                     0965  1678 ; Take the data from the multicast data area and store it in a new message
                     0965  1679 ; or bump the incarnation so we know to do it next time.
                     0965  1680 ;
                     0965  1681 ; Inputs:
                     0965  1682 ;       None.
                     0965  1683 ;
                     0965  1684 ; Outputs:
                     0965  1685 ;       R0 = success or failure
                     0965  1686 ;       All other registers are preserved.
                     0965  1687 ;
                     0965  1688 ; Implicit outputs:
                     0965  1689 ;       Multicast timer set to 1 second interval.
                     0965  1690 ;
                     0965  1691 ;--
                     0965  1692
                     0965  1693 LT$SETENTRY:
                     0965  1694         DSBINT  #LAT$C_IPL
        007E 8F   BB 096B  1695         PUSHR   #^M<R1,R2,R3,R4,R5,R6>  ; Save registers
             56   D4 096F  1696         CLRL    R6                      ; Assume no old CXB change flags
  52 FA7B CF   D0 0971  1697         MOVL    GHB_L_MULTIBFR, R2      ; Get the Multicast CXB
             1A   13 0976  1698         BEQL    10$                     ; Br if none, go make one
        F81D CF   96 0978  1699         INCB    GHB_B_INCARN            ; Bump the incarnation count
                     097C  1700         ASSUME  GHB_STS_V_MULTI EQ 0
  03 F818 CF   E9 097C  1701         BLBC    GHB_B_STATUS, 5$        ; Br if multicast buffer is free
             0099 31 0981  1702         BRW     70$                     ; Else, do this operation later
                     0984  1703         ;
                     0984  1704         ;  Multicast buffer is available, deallocate it!
                     0984  1705         ;
  56   4F A2   90 0984  1706 5$:      MOVB    XMT_B_MC_CHG_FLAG(R2),R6; Save previous change flags
       50   52 D0 0988  1707         MOVL    R2, R0                  ; Deallocate the old CXB buffer
       FA61 CF D4 098B  1708         CLRL    GHB_L_MULTIBFR          ; NO more mutlicast CXB buffer
            1247 30 098F  1709         BSBW    LT$DEALPOOL             ; Dump the buffer
                     0992  1710
  51   F806 CF   3C 0992  1711 10$:    MOVZWL  GHB_W_MC_SIZE, R1       ; Get size of data set by LATCP
 00000052 8F   C0 0997  1712         ADDL    #XMT_B_MC_SET, R1       ;    plus message overhead
             51   099D
 00000000'GF   16 099E  1713         JSB     G^EXE$ALONONPAGED       ; Now get the buffer
          7B 50   E9 09A4  1714         BLBC    R0, 80$                 ; No pool for buffer
    08 A2   51   B0 09A7  1715         MOVW    R1, CXB$W_SIZE(R2)      ; Size of buffer
                     09AB  1716         ASSUME  CXB$B_CODE EQ CXB$B_TYPE+1
        0119 8F   B0 09AB  1717         MOVW    #<1@8!DYN$C_CXB,-       ; Set type and indicate multicast
          0A A2      09AF  1718                 CXB$B_TYPE(R2)          ; buffer
        0048 8F   B0 09B1  1719         MOVW    #XMT_T_DATA,CXB$W_BOFF(R2) ; Set offset to start of data
          18 A2      09B5
  FA34 CF   52   D0 09B7  1720         MOVL    R2, GHB_L_MULTIBFR      ; Save address of the buffer
                     09BC  1721         ;
                     09BC  1722         ; Copy fixed portion of multicast message
                     09BC  1723         ;
  51   FA34 CF   9E 09BC  1724         MOVAB   GHB_T_MC_MSG,R1         ; Point to msg template
       61   0A   28 09C1  1725         MOVC3   #XMT_C_MC_LENGTH,(R1),- ; Copy the fixed portion of message
              48 A2      09C4  1726                 XMT_T_DATA(R2)
```

```
                        09C6  1727          ;
                        09C6  1728          ; Copy the variable data set by LATCP
                        09C6  1729          ;
51   F7D4 CF    9E      09C6  1730          MOVAB   GHB_T_MC_DATA,R1            ; Get address of variable data
61   F7CD CF    28      09CB  1731          MOVC3   GHB_W_MC_SIZE,(R1),(R3) ; Copy the variable data
           63            09D0
                        09D1  1732          ;
                        09D1  1733          ; Set change flags, and incarnation number
                        09D1  1734          ;
52   FA1B CF    D0      09D1  1735          MOVL    GHB_L_MULTIBFR,R2           ; Get back the multicast buffer
  56   FF 8F    8D      09D6  1736          XORB3   #XMT_CHFLG_M_ALL,R6,-   ; Set change flags
         4F A2          09DA
                        09DC  1737                    XMT_B_MC_CHG_FLAG(R2)
     F7B9 CF    90      09DC  1738          MOVB    GHB_B_INCARN,               ; Set incarnation number
         4E A2          09E0  1739                    XMT_B_MC_INCARN(R2)
  50   48 A2    9E      09E2  1740          MOVAB   XMT_T_DATA(R2), R0          ; Address of start of data
       53 50    C2      09E6  1741          SUBL    R0, R3                      ; Calculate the length
  1A   A2 53    B0      09E9  1742          MOVW    R3, CXB$W_BCNT(R2)          ; Set length of message
                        09ED  1743          ;
                        09ED  1744          ; Copy the node name and descriptor for the START message
                        09ED  1745          ;
51   F7AF CF    9E      09ED  1746          MOVAB   GHB_T_MC_DATA+2,R1          ; Get address of multicast variable data
53   FA16 CF    9E      09F2  1747          MOVAB   GHB_T_STRT_VAR,R3           ; Get address of start msg variable data
     56   53    D0      09F7  1748          MOVL    R3, R6                      ; Save address of start of variable data
     50   81    9A      09FA  1749          MOVZBL  (R1)+, R0                   ; Get size of group codes
     51   50    C0      09FD  1750          ADDL    R0, R1                      ; Calculate address of node name
     50   61    9B      0A00  1751          MOVZBW  (R1), R0                    ; Get size of node name
          50    B6      0A03  1752          INCW    R0                          ; Include the byte count
63   61   50    28      0A05  1753          MOVC3   R0,(R1),(R3)                ; Copy the node name
     50   61    9B      0A09  1754          MOVZBW  (R1), R0                    ; Get size of node descriptor
          50    B6      0A0C  1755          INCW    R0                          ; Include the byte count
63   61   50    28      0A0E  1756          MOVC3   R0,(R1),(R3)                ; Copy the node descriptor
          83    B4      0A12  1757          CLRW    (R3)+                       ; No location text, no parameters
     53   56    C2      0A14  1758          SUBL    R6, R3                      ; Calculate length of start message
     53   0C    A1      0A17  1759          ADDW3   #STRT_C_LENGTH, R3, -       ; Store length in holding cell
     F9E1 CF            0A1A
                        0A1D  1760                    GHB_W_STRT_LEN
50   0000'8F    3C      0A1D  1761  70$:     MOVZWL  #SS$_NORMAL, R0             ; Return success
F9C7 CF    01    B0      0A22  1762  80$:     MOVW    #1, GHB_W_MULTIMR           ; Send multicast msg on next tick
     007E 8F    BA      0A27  1763  90$:     POPR    #^M<R1,R2,R3,R4,R5,R6>      ; Restore registers
                        0A2B  1764          ENBINT
                  05    0A2E  1765          RSB
                        0A2F  1766
```

```
                     0A2F  1768          .SBTTL  LT$UNIT_INIT  - Unit Initialization
                     0A2F  1769 ;++
                     0A2F  1770 ; LT$UNIT_INIT - Unit Initialization
                     0A2F  1771 ;
                     0A2F  1772 ; Functional description:
                     0A2F  1773 ;
                     0A2F  1774 ; This routine performs a simple unit initialization.
                     0A2F  1775 ;
                     0A2F  1776 ; Inputs:
                     0A2F  1777 ;
                     0A2F  1778 ;       R5 = UCB address
                     0A2F  1779 ;
                     0A2F  1780 ; Outputs:
                     0A2F  1781 ;
                     0A2F  1782 ;       R0-R3 are modified
                     0A2F  1783 ;--
                     0A2F  1784
                     0A2F  1785 LT$UNIT_INIT:
          05   E0    0A2F  1786          BBS     #UCB$V_POWER,-          ; Skip if power fail recovery
       5A 64   A5    0A31  1787                  UCB$L_STS(R5), 70$      ;
                     0A34  1788
 00000000'GF   D0    0A34  1789          MOVL    G^TTY$GL_DPT,R1         ; Address of class dpt
           51        0A3A
    50   1E   A1    3C    0A3B  1790          MOVZWL  DPT$W_VECTOR(R1),R0     ; Locate class driver vector table
       51   50    C0    0A3F  1791          ADDL    R0,R1                   ; Relocate base address
  0114   C5   51    D0    0A42  1792          MOVL    R1,UCB$L_TT_CLASS(R5)   ; Set terminal class driver vector
                     0A47  1793
  010C   C5   61    D0    0A47  1794          MOVL    CLASS_GETNXT(R1),UCB$L_TT_GETNXT(R5)
          04   A1    D0    0A4C  1795          MOVL    CLASS_PUTNXT(R1),UCB$L_TT_PUTNXT(R5)
        0110   C5          0A4F
    50    28   A5    D0    0A52  1796          MOVL    UCB$L_DDB(R5),R0        ; Get DDB address
OC   A0   10   A1    D0    0A56  1797          MOVL    CLASS_DDT(R1),DDB$L_DDT(R0)
          10   A1    D0    0A5B  1798          MOVL    CLASS_DDT(R1),UCB$L_DDT(R5) ; Set DDT address in UCB
        0088   C5          0A5E
                     0A61  1799
          54   A5    B5    0A61  1800          TSTW    UCB$W_UNIT(R5)          ; Special action for ucb 0
               07    13    0A64  1801          BEQL    20$                     ; Its is unit zero
                     0A66  1802          CLRBIT  #UCB$V_TEMPLATE,-       ; All others lose the template
                     0A66  1803                  UCB$L_STS(R5)           ;  bit so they won't clone on assign
               1D    11    0A6B  1804          BRB     40$                     ; Continue
 F70E   CF     55    D0    0A6D  1805 20$:     MOVL    R5, GHB_L_UCB0          ; Save address of ucb zero
                     0A72  1806 ;
                     0A72  1807 ; We are setting or clearing some bits in the LTA0 device here so that all
                     0A72  1808 ; later lta devices will have them set/clear the same way.  These bits are
                     0A72  1809 ; different from the defaults that are set in the sysgen parameters for
                     0A72  1810 ; the other types of terminals.
                     0A72  1811 ;
                     0A72  1812 ;       SETBIT  #TT$V_MODEM, -          ; Set modem so we do modem
                     0A72  1813 ;               UCB$L_TT_DECHAR(R5)     ;  processing for this terminal
                     0A72  1814          SETBIT  #TT$V_REMOTE,-          ; Set remote so we know its a remote
                     0A72  1815                  UCB$L_TT_DECHAR(R5)     ;  terminal forever.
                     0A78  1816                                          ; See setup_ucb routine for reasons
                     0A78  1817          CLRBIT  #TT2$V_AUTOBAUD,-       ; We do not autobaud here.
                     0A78  1818                  UCB$L_TT_DECHA1(R5)     ;
                     0A7E  1819          SETBIT  #TT2$V_HANGUP,-         ; Set default for terminal to hangup
                     0A7E  1820                  UCB$L_TT_DECHA1(R5)     ; on logout.  this can be cleared by
                     0A84  1821                                          ; the user from his terminal.
```

```
                     0A84  1822          SETBIT  #TTY$V_PC_NOTIME,-      ; no timer service please
                     0A84  1823                    UCB$W_TT_PRTCTL(R5)   ; ...
                     0A8A  1824  40$:     ;
                     0A8A  1825          ; Finish initializing UCB.
                     0A8A  1826          ;
           0F   10   0A8A  1827          BSBB    LT$UCB_INIT             ; Initialize the terminal UCB
           0C   11   0A8C  1828          BRB     90$                     ; And leave
                     0A8E  1829
     50 0134 C5   D0   0A8E  1830  70$:  MOVL    UCB$L_LT_CSB(R5), R0    ; On power fail, reset the timeout
           05   13   0A93  1831          BEQL    90$                     ; No CSB, just forget it
        66 A0   B0   0A95  1832          MOVW    CSB_W_TIMRESET(R0),-    ; And just wait on some thing
        50 A0        0A98  1833                  CSB_W_TIMEOUT(R0)       ; To happen.
           05   GA9A  1834  90$:  RSB
```

```
                        0A9B  1836          .SBTTL  LT$UCB_INIT  - Initialize a terminal UCB
                        0A9B  1837 ;++
                        0A9B  1838 ; LT$UCB_INIT - Initialize a terminal UCB
                        0A9B  1839 ;
                        0A9B  1840 ; Functional description:
                        0A9B  1841 ;
                        0A9B  1842 ;       Init the terminal UCB before starting a process to use it.
                        0A9B  1843 ;
                        0A9B  1844 ; Inputs:
                        0A9B  1845 ;       R5 = UCB address
                        0A9B  1846 ;
                        0A9B  1847 ; Outputs:
                        0A9B  1848 ;       R5 = UCB address
                        0A9B  1849 ;       R0, R1 clobbered
                        0A9B  1850 ;--
                        0A9B  1851
                        0A9B  1852 LT$UCB_INIT:
          54 A5   B5    0A9B  1853          TSTW    UCB$W_UNIT(R5)          ; Is this unit #0?
             3D   13    0A9E  1854          BEQL    90$                     ; Br if yes, skip it
          5C A5   B5    0AA0  1855          TSTW    UCB$W_REFC(R5)          ; Any channels attached?
             21   12    0AA3  1856          BNEQ    40$                     ; Br if yes, don't reinit UCB
       01    54 A5   78    0AA5  1857          ASHL    UCB$W_UNIT(R5),#1,R1    ; Build unit's bit mask
             51          0AA9
  0080 C5    51   B0    0AAA  1858          MOVW    R1,UCB$B_ERTCNT(R5)     ; Save it
       0141  C5   94    0AAF  1859          CLRB    UCB$B_LT_LOCID(R5)      ; Flag as available (no local slot indx)
       FA05  CF   DE    0AB3  1860          MOVAL   LT$VECTOR,-             ; Set the port address in ucb
       0118  C5          0AB7  1861                  UCB$L_TT_PORT(R5)       ;  in case port driver reloaded
       00CC  C5   D4    0ABA  1862          CLRL    UCB$L_TT_WFLINK(R5)     ; Zero to force reinit of queue head
 51    0114  C5   D0    0ABE  1863          MOVL    UCB$L_TT_CLASS(R5),R1   ; Address class vector table
          08 B1   16    0AC3  1864          JSB     @CLASS_SETUP_UCB(R1)    ; Init UCB fields
                        0AC6  1865 40$:
             01   90    0AC6  1866          MOVB    #UCB$M_LT_DATA,-        ; Look for data first time
       0142  C5          0AC8  1867                  UCB$B_LT_DATAW(R5)
            50  8F      90    0ACB  1868          MOVB    #LAT$C_UCB_BUFSIZ,-     ; Init buffer
       014A  C5          0ACE  1869                  UCB$B_LT_MAXC(R5)       ;       ...
       014B  C5   94    0AD1  1870          CLRB    UCB$B_LT_CURC(R5)       ;       ...
       0146  C5   94    0AD5  1871          CLRB    UCB$B_LT_TCRED(R5)      ;       ...
       0120  C5   B4    0AD9  1872          CLRW    UCB$W_TT_OUTLEN(R5)     ; No data waiting
                   05   0ADD  1873 90$:     RSB
```

```
                                0ADE  1875          .SBTTL  LT$STARTIO  - Start I/O routine
                                0ADE  1876 ;++
                                0ADE  1877 ; LT$STARTIO - Start I/O Operation
                                0ADE  1878 ;
                                0ADE  1879 ; Functional description:
                                0ADE  1880 ;
                                0ADE  1881 ; This routine is entered from the device independent terminal startio
                                0ADE  1882 ; Routine to enable output interrupts on an idle unit.
                                0ADE  1883 ;
                                0ADE  1884 ; Inputs:
                                0ADE  1885 ;
                                0ADE  1886 ; R3 =  character        and     cc = plus
                                0ADE  1887 ; R3 =  address          and     cc = minus      and     R2 = count
                                0ADE  1888 ; R5 = ucb address
                                0ADE  1889 ;
                                0ADE  1890 ; Outputs:
                                0ADE  1891 ;
                                0ADE  1892 ;       R5 = UCB address
                                0ADE  1893 ;       R0,R1,R2,R3 and R4 are modified
                                0ADE  1894 ;--
                                0ADE  1895
                                0ADE  1896 LT$STARTIO:                                  ; Start I/O on unit
          0760 8F    BB         0ADE  1897          PUSHR   #^M<R5,R6,R8,R9,R10>
    58    0134 C5    D0         0AE2  1898          MOVL    UCB$L_LT_CSB(R5), R8        ; Get Circuit State Block
               59    13         0AE7  1899          BEQL    85$                         ; Br if UCB is hanging up
                                0AE9  1900          ;
                                0AE9  1901          ;  Dispatch on our terminal UCB state
                                0AE9  1902          ;
                                0AE9  1903          $DISPATCH   UCB$B_LT_STATE(R5),TYPE=B,-
                                0AE9  1904          <-      ; state             ; action
                                0AE9  1905            <UCB$C_LT_STATE_KILL  168$>,- ; KILL state, flush data
                                0AE9  1906          >
                                0AF1  1907
                                0AF1  1908          ;  All other states -- okay to send data
                                0AF1  1909
            01    88            0AF1  1910          BISB    #UCB$M_LT_DATA,-            ; Assume output data available
          0142 C5               0AF3  1911                   UCB$B_LT_DATAW(R5)
                                0AF6  1912          ;
                                0AF6  1913          ;   Buffer data locally
                                0AF6  1914          ;
                                0AF6  1915 5$:      $DISPATCH   UCB$B_TT_OUTYPE(R5),TYPE=B,-
                                0AF6  1916          <-      ; type              ; action
                                0AF6  1917                  < 0     170$>,-         ; No data, allow more calls
                                0AF6  1918                  < 1     10$>,-          ; Single character
                                0AF6  1919          >
                                0B00  1920          ;
                                0B00  1921          ;  Default to burst data
                                0B00  1922          ;
    53    011C C5    D0         0B00  1923          MOVL    UCB$L_TT_OUTADR(R5),R3  ; Get output buffer address
    52    0120 C5    3C         0B05  1924          MOVZWL  UCB$W_TT_OUTLEN(R5),R2  ;  and charcter count
               21    11         0B0A  1925          BRB     30$                     ; Continue in common path
                                0B0C  1926
    50    014A C5    9E         0B0C  1927 10$:     MOVAB   UCB$B_LT_MAXC(R5),R0    ; Get address of maxc
          60    80    91         0B11  1928          CMPB    (R0)+,(R0)              ; Any room left ?
                02    14         0B14  1929          BGTR    20$                     ; Br if yes
                60    97         0B16  1930          DECB    (R0)                    ; Make room for it
                60    96         0B18  1931 20$:     INCB    (R0)                    ; Update count
```

```
                51    00    9A  0B1A  1932          MOVZBL  (R0),R1                 ; Get count
                50    51    C0  0B1D  1933          ADDL    R1,R0                  ; Form destination address
                60    53    90  0B20  1934          MOVB    R3,(R0)                ; Load character
                                0B23  1935 ;
                                0B23  1936 ;       We are setting up the address and count of buffered characters in
                                0B23  1937 ;       case we are balanced and we should send these characters in a msg
                                0B23  1938 ;       to come out of balanced mode
                                0B23  1939 ;
            52  014B C5    9A  0B23  1940          MOVZBL  UCB$B_LT_CURC(R5), R2  ; Get count of buffered characters
            53  014C C5    9E  0B28  1941          MOVAB   UCB$B_LT_CBUF(R5), R3  ; Get address of buffer
                      00    E1  0B2D  1942 30$:     BBC     #FLAG_V_RRF,-          ; Br if we can we send an
                03 5C A8        0B2F  1943                  CSB_B_FLAG(R8),70$     ;  unsolicited message
                   00E0       31  0B32  1944 60$:     BRW     190$
                                0B35  1945 ;
                                0B35  1946 ; Transmit data to concentrator
                                0B35  1947 ;
                0146 C5    97  0B35  1948 70$:     DECB    UCB$B_LT_TCRED(R5)     ; Use a credit
                      0A    18  0B39  1949          BGEQ    90$                    ; Br if we have one to use
                0146 C5    96  0B3B  1950 80$:     INCB    UCB$B_LT_TCRED(R5)     ; Else, no credits
                                0B3F  1951 ;
                                0B3F  1952 ;       Woops.  We find we have data and we can't send it anywhere since
                                0B3F  1953 ;       we have no credit.  Never mind we can just leave here.  We will
                                0B3F  1954 ;       receive a credit message eventually.  We are set non-balanced here
                                0B3F  1955 ;       and if we leave so that all the int bits are turned off in the
                                0B3F  1956 ;       other ucbs, and this one, then we won't get anymore embarrassing
                                0B3F  1957 ;       startio calls.  When we get the credit message we will start things
                                0B3F  1958 ;       up again.
                                0B3F  1959 ;
                   00CE       31  0B3F  1960          BRW     180$                   ; Leave quietly
                                0B42  1961
                                0B42  1962 ;
                                0B42  1963 ;       We have nowhere to put the data and the circuit is down, so just
                                0B42  1964 ;       clear interrupt expected, discard the data and return
                                0B42  1965 ;
                   00C4       31  0B42  1966 85$:     BRW     168$                   ; Leave, flush data & clear INT expected
                                0B45  1967
                56    34 A8    7E  0B45  1968 90$:     MOVAQ   CSB_Q_XBUFQ(R8),R6     ; Get transmit buffer queue address
                      56 66    D1  0B49  1969          CMPL    (R6), R6               ; Is there a transmit buffer?
                         ED    13  0B4C  1970          BEQL    80$                    ; Br if no transmit buffer, ignore it
                5D A8    01    90  0B4E  1971          MOVB    #1, CSB_B_NUM_SLOTS(R8) ; We have data for the server now
                                0B52  1972          SETBIT  #FLAG_V_RRF,-          ; No more unsolicited messages
                                0B52  1973                  CSB_B_FLAG(R8)         ;
                52 A8    02    B0  0B57  1974          MOVW    #LAT$C_HOST_TIMER, -   ; Set retransmit timer
                                0B5B  1975                  CSB_W_XMTTMO(R8)       ;
                      56 66    D0  0B5B  1976          MOVL    (R6), R6               ; Get the buffer address
            50  014C C5    9E  0B5E  1977          MOVAB   UCB$B_LT_CBUF(R5), R0  ; Address the UCB extension
                      50 53    D1  0B63  1978          CMPL    R3, R0                 ; Is that our buffer?
                         04    12  0B66  1979          BNEQ    100$                   ; Br if no
                                0B68  1980 ; ***
                                0B68  1981 ; Reading the code I noticed the following line(s) would not work if
                                0B68  1982 ; the startio entry were to discover that it were balanced after
                                0B68  1983 ; previous previous calls stored data in the UCB extensions because
                                0B68  1984 ; we were not balanced. I guess that never happens, because UCB$B_LT_CURC
                                0B68  1985 ; would not be equal to one, as assumed below, but some higher value. BEM
                                0B68  1986 ; ***
                0148 C5    97  0B68  1987          DECB    UCB$B_LT_CURC(R5)      ; Yes. Don't save the character there
                                0B6C  1988 100$:
```

```
                59   50 A6   9E   0B6C  1989          MOVAB   XMT_T_MDATA(R6), R9      ; Address of the slot to use (first)
5A              0149 C5      9A   0B70  1990          MOVZBL  UCB$B_LT_SLOTSZ(R5), R10; Get the maximum slot size
                                  0B75  1991
                                  0B75  1992          ASSUME SLT_B_DSTID EQ 0
                                  0B75  1993          ASSUME SLT_B_SRCID EQ <SLT_B_DSTID+1>
                                  0B75  1994          ASSUME SLT_B_COUNT EQ <SLT_B_SRCID+1>
                                  0B75  1995          ASSUME SLT_B_CRED EQ <SLT_B_COUNT+1>
                                  0B75  1996
                                  0B75  1997          ASSUME  UCB$B_LT_LOCID EQ UCB$B_LT_REMID+1
                                  0B75  1998
89              0140 C5      B0   0B75  1999          MOVW    UCB$B_LT_REMID(R5), (R9)+ ; Load remote and local ids
                89   5A      90   0B7A  2000          MOVB    R10, (R9)+                ; Save the character count
                                  0B7D  2001          ;
                                  0B7D  2002          ; Extend credits to concentrator if needed
                                  0B7D  2003          ;
                0147 C5      81   0B7D  2004          ADDB3   UCB$B_LT_XCRED(R5),-      ; See if we need to send
                50   01           0B81  2005                  #LAT$C_MAX_RCRED,R0       ;   a credit to the remote
0148 C5         50      80   0B83  2006          ADDB    R0,UCB$B_LT_RCRED(R5)     ; Add them to total credits to send
0147 C5         50      82   0B88  2007          SUBB    R0,UCB$B_LT_XCRED(R5)     ; Credits have been extended
89              0148 C5      90   0B8D  2008          MOVB    UCB$B_LT_RCRED(R5), (R9)+ ; Load credits
                0148 C5      94   0B92  2009          CLRB    UCB$B_LT_RCRED(R5)        ; Don't send them again!
                                  0B96  2010
                5A   52      D1   0B96  2011 110$:     CMPL    R2, R10                   ; Do we have more than we can handle?
                     03      1B   0B99  2012          BLEQU   120$                      ; Br if no, its just fine
                52   5A      D0   0B9B  2013          MOVL    R10, R2                   ; Use just what we can handle
                5A   52      C2   0B9E  2014 120$:     SUBL    R2, R10                   ; Adjust what we have left
                                  0BA1  2015 ; ***
                                  0BA1  2016 ; Reading the code I noticed the following line(s) would not work if
                                  0BA1  2017 ; the startio entry were to discover that it were balanced after
                                  0BA1  2018 ; previous calls stored data in the UCB extensions because
                                  0BA1  2019 ; we were not balanced. I guess that never happens, because R2 would
                                  0BA1  2020 ; not reflect the UCB extension data area. BEM
                                  0BA1  2021 ; ***
011C C5         52      C0   0BA1  2022          ADDL2   R2,UCB$L_TT_OUTADR(R5)  ; Update UCB state
0120 C5         52      A2   0BA6  2023          SUBW2   R2,UCB$W_TT_OUTLEN(R5)  ; Update UCB state
                55      DD   0BAB  2024          PUSHL   R5                      ; Save the UCB address
69              63      52   28   0BAD  2025          MOVC3   R2,(R3),(R9)            ; Copy the string
                55   8ED0        0BB1  2026          POPL    R5                      ; Restore the UCB address
                59      53   D0   0BB4  2027          MOVL    R3,R9                   ; Set address of next available byte
                        5A   D5   0BB7  2028          TSTL    R10                     ; Do we have any left?
                        28   13   0BB9  2029          BEQL    155$                    ; Br if no, all done.
                010C    D5   16   0BBB  2030          JSB     @UCB$L_TT_GETNXT(R5)    ; Else, call back class driver for more
                        19   13   0BBF  2031          BEQL    150$                    ; Br if no more to be had
                        0C   14   0BC1  2032          BGTR    140$                    ; Br is single character
                                  0BC3  2033          ;
                                  0BC3  2034          ;  Found a burst of data
                                  0BC3  2035          ;
53              011C C5      D0   0BC3  2036          MOVL    UCB$L_TT_OUTADR(R5),R3  ; Get output buffer address
52              0120 C5      3C   0BC8  2037          MOVZWL  UCB$W_TT_OUTLEN(R5),R2  ;  and charcter count
                     C7      11   0BCD  2038          BRB     110$                    ; Else buffer address
                                  0BCF  2039
                                  0BCF  2040 140$:     ;
                                  0BCF  2041          ;  Found a single character
                                  0BCF  2042          ;
                52   01      D0   0BCF  2043          MOVL    #1, R2                  ; Else, just one character returned
                69   53      90   0BD2  2044          MOVB    R3, (R9)                ;  and fake many chars returned
                53   59      D0   0BD5  2045          MOVL    R9, R3                  ; Copy buffer address
```

```
                BC   11  0BD8  2046          BRB     110$                    ;  and continue
                                0BDA  2047
      64 A5     02   AA  0BDA  2048 150$:    BICW    #UCB$M_INT,UCB$L_STS(R5);Allow more start io calls (checked)
                01   8A  0BDE  2049          BICB    #UCB$M_LT_DATA,-        ;  no output data available
          0142  C5       0BE0  2050                    UCB$B_LT_DATAW(R5)
   52     59    56   C3  0BE3  2051 155$:    SUBL3   R6, R9, R2              ; Make the length of the data to
    00000048    8F   C2  0BE7  2052          SUBL    #XMT_T_DATA, R2         ;  transmit
                52       0BED
                59   D6  0BEE  2053          INCL    R9                      ; Round end address of data
          59    01   CA  0BF0  2054          BICL    #1,R9                   ;  ...
      1C A6     59   D0  0BF3  2055          MOVL    R9,CXB$L_T_ENDADR(R6)   ; Store it in the CXB
          52    0C   C2  0BF7  2056          SUBL    #<<XMT_T_MDATA- -       ; Make total characters in slot
                         0BFA  2057                  XMT_T_DATA> + -         ; minus those already accounted for
                         0BFA  2058                  SLT_T_DATA>, R2         ; beyond the circuit and slot headers
      52 A6     52   90  0BFA  2059          MOVB    R2, <SLT_B_COUNT+-      ; Store that count away.
                         0BFE  2060                  XMT_T_MDATA>(R6)        ; In the first slot maxc
                         0BFE  2061
                         0BFE  2062          .IF DEFINED LT_XMT_CHECK
                         0BFE  2063          ;
                         0BFE  2064          ;  debug code
                         0BFE  2065          ;
                         0BFE  2066          MOVQ    CSB_T_CIRCHDR(R8),-             ; Move the circ header
                         0BFE  2067                  XMT_T_DATA(R6)                  ; to look at real msg
                         0BFE  2068          MOVZBL  XMT_B_NUM_SLOTS(R6), R1         ; number of valid slots
                         0BFE  2069          BNEQ    160$                            ; We have some, skip check
                         0BFE  2070          ; No slots - we must only have a circuit header!
                         0BFE  2071          MOVAB   XMT_T_MDATA(R6), R0             ; Past circuit header
                         0BFE  2072          CMPL    R0, R9                          ; Only have a circuit header?
                         0BFE  2073          BEQL    160$                            ; Br if yes, okay
                         0BFE  2074          DEBUG                                   ; Not correct
                         0BFE  2075
                         0BFE  2076          .ENDC   ;; DEFINED LT_XMT_CHECK
                         0BFE  2077
      54    58    D0   0BFE  2078 160$:    MOVL    R8, R4                  ; CSB address for call
          0E9C    30   0C01  2079          BSBW    LT$XMITONE              ; Transmit the buffer
      0760 8F     BA   0C04  2080          POPR    #^M<R5,R6,R8,R9,R10>
                  05   0C08  2081          RSB
                         0C09  2082
                         0C09  2083 168$:    ;
                         0C09  2084          ;  Discard output data, no session active !
                         0C09  2085          ;
            004D  30   0C09  2086          BSBW    LT$FLUSH_DATA           ; Flush the output data
                         0C0C  2087
                         0C0C  2088 170$:    ;
                         0C0C  2089          ;  Allow more calls to startio
                         0C0C  2090          ;
      64 A5     02   AA  0C0C  2091          BICW    #UCB$M_INT, -           ; Allow more calls here
                         0C10  2092                  UCB$L_STS(R5)           ;
      0760 8F     BA   0C10  2093 180$:    POPR    #^M<R5,R6,R8,R9,R10>
                  05   0C14  2094          RSB
                         0C15  2095
                         0C15  2096 ;
                         0C15  2097 ;        Not balanced, so buffer data if we can.
                         0C15  2098 ;        R2 = count
                         0C15  2099 ;        R3 -> data
                         0C15  2100 ;
                         0C15  2101
```

```
                     014C C5    9E  0C15  2102 190$:  MOVAB   UCB$B_LT_CBUF(R5), R4   ; Address of character buffer
                  54    53    D1  0C1A  2103         CMPL    R3, R4                 ; Is this the character buffer?
                        28    13  0C1D  2104         BEQL    210$                   ; Br if yes, check if its full
                  50    014B C5    9A  0C1F  2105         MOVZBL  UCB$B_LT_CURC(R5), R0  ; Calculate amount of room
                        51    D4  0C24  2106         CLRL    R1                     ;  left in the output buffer
                  014A C5    50    83  0C26  2107         SUBB3   R0, -                  ;       ...
                        51          0C2B
                                    0C2C  2108                 UCB$B_LT_MAXC(R5), R1  ;       ...
                        52    51    D1  0C2C  2109         CMPL    R1, R2                 ; Do we have room?
                              DF    15  0C2F  2110         BLEQ    180$                   ; Br if no, so shut output off
                        0120 C5    B4  0C31  2111         CLRW    UCB$W_TT_OUTLEN(R5)    ; Else, forget about the data here.
                  014B C5    52    80  0C35  2112         ADDB2   R2, UCB$B_LT_CURC(R5)  ; Calculate the number of chars
                     51    6440    9E  0C3A  2113         MOVAB   (R4)[R0], R1           ;  and get address to put them
                                    0C3E  2114 200$:
                              55    DD  0C3E  2115         PUSHL   R5                     ; Save the UCB address
                  61    63    52    28  0C40  2116         MOVC3   R2,(R3),(R1)           ; Copy the string
                              55  8ED0  0C44  2117         POPL    R5                     ; Restore the UCB address
                                    0C47  2118
                        014B C5    91  0C47  2119 210$:  CMPB    UCB$B_LT_CURC(R5), -   ; Is buffer full?
                        014A C5        0C4B
                                    0C4E  2120                 UCB$B_LT_MAXC(R5)
                              C0    18  0C4E  2121         BGEQ    180$                   ; Br if yes, shut off output
                        010C D5    16  0C50  2122         JSB     @UCB$L_TT_GETNXT(R5)   ; Else, get more data
                              B6    13  0C54  2123         BEQL    170$                   ; No more data, so just return
                                 FE9D   31  0C56  2124         BRW     5$                     ; Else, process data
                                    0C59  2125
```

```
                     0C59  2127          .SBTTL  LT$FLUSH_DATA  - Flush buffered output data and drain TT buffer
                     0C59  2128 ;++
                     0C59  2129 ; LT$FLUSH_DATA - Flush buffered output data and drain class driver buffer
                     0C59  2130 ;
                     0C59  2131 ; Functional description:
                     0C59  2132 ;
                     0C59  2133 ; Flush buffered output data and drain class driver output buffer.
                     0C59  2134 ;
                     0C59  2135 ; Inputs:
                     0C59  2136 ;       R5 = UCB address
                     0C59  2137 ;
                     0C59  2138 ; Outputs:
                     0C59  2139 ;       R3 is destroyed.
                     0C59  2140 ;--
                     0C59  2141
                     0C59  2142 LT$FLUSH_DATA:
010C D5    16        0C59  2143 20$:    JSB     @UCB$L_TT_GETNXT(R5)    ; Call back for more to class driver
      02   13        0C5D  2144         BEQL    90$                     ; Br if no more to be had
      F8   11        0C5F  2145         BRB     20$                     ; Else, drain all output data
0120 C5    B4        0C61  2146 90$:    CLRW    UCB$W_TT_OUTLEN(R5)     ; Clear CLASS driver output data
014B C5    94        0C65  2147         CLRB    UCB$B_LT_CURC(R5)       ; Dump UCB output buffering
           05        0C69  2148         RSB
```

```
                     0C6A  2150          .SBTTL  LT$ABORT  - Abort (flush) buffered output data
                     0C6A  2151 ;++
                     0C6A  2152 ; LT$ABORT - Abort buffered output data
                     0C6A  2153 ;
                     0C6A  2154 ; Functional description:
                     0C6A  2155 ;
                     0C6A  2156 ; Abort (flush) buffered output data
                     0C6A  2157 ;
                     0C6A  2158 ; Inputs:
                     0C6A  2159 ;       R5 = UCB address
                     0C6A  2160 ;
                     0C6A  2161 ; Outputs:
                     0C6A  2162 ;
                     0C6A  2163 ;--
                     0C6A  2164
                     0C6A  2165 LT$ABORT:
014B C5   94         0C6A  2166          CLRB    UCB$B_LT_CURC(R5)       ; Dump UCB buffering
0120 C5   B4         0C6E  2167          CLRW    UCB$W_TT_OUTLEN(R5)     ; Dump pending class driver data,
                     0C72  2168                                          ;  (might have dissappeared)
      04   88        0C72  2169          BISB    #UCB$M_LT_ABORT,-       ; Abort requested
0142 C5               0C74  2170           UCB$B_LT_DATAW(R5)
      05              0C77  2171          RSB
                     0C78  2172
                     0C78  2173
                     0C78  2174
                     0C78  2175
                     0C78  2176          .SBTTL  LT$FLOW_CHANGE  - Change flow control
                     0C78  2177
                     0C78  2178 ;+
                     0C78  2179 ;       LT$FLOW_CHANGE
                     0C78  2180 ;
                     0C78  2181 ; Function: Inform terminal server of change in host flow control.
                     0C78  2182 ;                The terminal server will mimic host policy.
                     0C78  2183 ;
                     0C78  2184 ; Inputs:
                     0C78  2185 ;       R5 = UCB address
                     0C78  2186 ;
                     0C78  2187 ; Outputs:
                     0C78  2188 ;
                     0C78  2189 ;-
                     0C78  2190
                     0C78  2191 LT$FLOW_CHANGE:
      02   88        0C78  2192          BISB    #UCB$M_LT_FLOW,-        ; Flow change
0142 C5               0C7A  2193           UCB$B_LT_DATAW(R5)
      05              0C7D  2194          RSB
                     0C7E  2195
```

```
                     0C7E  2197          .SBTTL  LT$XON  - Resume input stream into class driver
                     0C7E  2198 ;++
                     0C7E  2199 ; LT$XON - Resume input stream into class driver
                     0C7E  2200 ;
                     0C7E  2201 ; Inputs:
                     0C7E  2202 ;       R3 = XON character
                     0C7E  2203 ;       R5 = UCB address
                     0C7E  2204 ;
                     0C7E  2205 ; Outputs:
                     0C7E  2206 ;
                     0C7E  2207 ;       R3 is destroyed
                     0C7E  2208 ;--
                     0C7E  2209 LT$XON:
          1803 8F    BB  0C7E  2210          PUSHR   #^M<R0,R1,R11,AP>       ; Save registers
                     0C82  2211          CLRBIT  #UCB$V_LT_OVFLOW,-      ; Clear overflow indicator
                     0C82  2212                    UCB$B_LT_LATSTS(R5)
     50    013C C5    D0  0C88  2213          MOVL    UCB$L_LT_INPBUF(R5),R0  ; Get address of input buffer
              3A    13  0C8D  2214          BEQL    70$                     ; Br if none, nothing to do
          0147 C5    96  0C8F  2215          INCB    UCB$B_LT_XCRED(R5)      ; Okay to return credit now
                     0C93  2216
                     0C93  2217          .IF DEFINED LT_CRED_CHECK
                     0C93  2218          BLEQ    10$                     ; Br if okay
                     0C93  2219          DEBUG                           ; Else, we went too far
                     0C93  2220 10$:
                     0C93  2221          .ENDC   ;; DEFINED LT_CRED_CHECK
                     0C93  2222
                     0C93  2223          ASSUME  UCB$B_LT_LATSTS EQ UCB$B_LT_DATAW+1
                     0C93  2224
          1001 8F    A8  0C93  2225          BISW    #<UCB$M_LT_INPUT@8>!UCB$M_LT_DATA,- ; Data potentially available
          0142 C5        0C97  2226                    UCB$B_LT_DATAW(R5)      ;   and INPUT stream in enabled
        64 A5    02    88  0C9A  2227          BISB    #UCB$M_INT,UCB$L_STS(R5); Force no data to be returned
        5C   02 A0    3C  0C9E  2228          MOVZWL  INB_W_BCNT(R0),AP       ; Get count of bytes remaining
           5B   60    3C  0CA2  2229          MOVZWL  INB_W_BOFF(R0),R11      ; Get offset to start of input data
           5B   50    C0  0CA5  2230          ADDL    R0,R11                  ; Calculate address of input data
                     0CA8  2231          ;
                     0CA8  2232          ;  Feed remaining data into class driver
                     0CA8  2233          ;
           53   8B    9A  0CA8  2234 30$:     MOVZBL  (R11)+,R3               ; Get next character
          0110 D5    16  0CAB  2235          JSB     @UCB$L_TT_PUTNXT(R5)    ; Pass character to class driver
               02    E0  0CAF  2236          BBS     #UCB$V_LT_OVFLOW,-      ; Br if overflow, wait for next XON
        0F 0143 C5        0CB1  2237                    UCB$B_LT_LATSTS(R5),50$
             F0 5C    F5  0CB5  2238          SOBGTR  AP,30$                  ; Loop if more
     50    013C C5    D0  0CB8  2239          MOVL    UCB$L_LT_INPBUF(R5),R0  ; Get input buffer address
              0F19    30  0CBD  2240          BSBW    LT$DEALPOOL             ; Deallocate the input buffer
          013C C5    D4  0CC0  2241          CLRL    UCB$L_LT_INPBUF(R5)     ; Zero input buffer pointer
                10    8A  0CC4  2242 50$:     BICB    #UCB$M_LT_INPUT,-       ; Input stream is now disabled
          0143 C5        0CC6  2243                    UCB$B_LT_LATSTS(R5)
          1803 8F    BA  0CC9  2244 70$:     POPR    #^M<R0,R1,R11,AP>       ; Restore registers
                     05  0CCD  2245          RSB
                     0CCE  2246
                     0CCE  2247          .SBTTL  LT$XOFF  - Stop input stream into class driver
                     0CCE  2248
                     0CCE  2249 ;++
                     0CCE  2250 ; LT$XOFF - Stop input stream into class driver
                     0CCE  2251 ;
                     0CCE  2252 ; Inputs:
                     0CCE  2253 ;       R3 = XOFF character (^S or ^G)
```

```
                                                J 2
                     OCCE  2254 ;       R5 = UCB address
                     OCCE  2255 ;       AP = count of bytes remaining
                     OCCE  2256 ;       R11 = address of input data
                     OCCE  2257 ;
                     OCCE  2258 ; Implicit inputs:
                     OCCE  2259 ;
                     OCCE  2260 ;       UCB$V_LT_INPUT bit must be on UCB$B_LT_LATSTS status byte.
                     OCCE  2261 ;
                     OCCE  2262 ; Outputs:
                     OCCE  2263 ;       R3 is destroyed
                     OCCE  2264 ;
                     OCCE  2265 ;--
                     OCCE  2266 LT$XOFF:
          37   BB    OCCE  2267          PUSHR   #^M<R0,R1,R2,R4,R5>     ; Save registers
      07  53   91    OCD0  2268          CMPB    R3,#TTY_C_BELL          ; Is this a control G?
          44   13    OCD3  2269          BEQL    90$                     ; Br if yes, do nothing
          04   E1    OCD5  2270          BBC     #UCB$V_LT_INPUT,-       ; Br if input stream not enabled
   3E 0143 C5         OCD7  2271            UCB$B_LT_LATSTS(R5),90$
          01   5C    D1    OCDB  2272          CMPL    AP,#1               ; Any data to buffer? (counting current)
               39    15    OCDE  2273          BLEQ    90$                 ; Br if no, skip it
               5C    D7    OCE0  2274          DECL    AP                  ; Else, account for character just passed
                           OCE2  2275          SETBIT  #UCB$V_LT_OVFLOW,-  ; Set overflow indicator
                           OCE2  2276            UCB$B_LT_LATSTS(R5)
   52   013C C5    D0    OCE8  2277          MOVL    UCB$L_LT_INPBUF(R5),R2  ; Get input buffer
               16   12    OCED  2278          BNEQ    30$                 ; Br if one there
   51   5C   0C    C1    OCEF  2279          ADDL3   #INB_C_LENGTH,AP,R1     ; Else, calculate length of buffer
    00000000'GF     16    OCF3  2280          JSB     G^EXE$ALONONPAGED       ; Allocate a buffer
          1D   50   E9    OCF9  2281          BLBC    R0,90$                  ; Br if no buffer
   013C C5     52   D0    OCFC  2282          MOVL    R2,UCB$L_LT_INPBUF(R5)  ; Save address of input buffer
     08 A2     51   B0    OD01  2283          MOVW    R1,INB_W_SIZE(R2)       ; Set size of input buffer
                           OD05  2284 30$:     ASSUME  INB_B_SPARE EQ INB_B_TYPE+1
     0A A2     13   98    OD05  2285          MOVZBW  #DYN$C_BUFIO,INB_B_TYPE(R2) ; Set structure type
     02 A2     5C   B0    OD09  2286          MOVW    AP,INB_W_BCNT(R2)       ; Set size of remaining string
        62     0C   B0    OD0D  2287          MOVW    #INB_C_LENGTH,INB_W_BOFF(R2) ; Set offset to start of data
                           OD10  2288          ;
                           OD10  2289          ;  Don't return credit... wait for XON
                           OD10  2290          ;
      0147 C5    97    OD10  2291          DECB    UCB$B_LT_XCRED(R5)      ; Don't return credit
                           OD14  2292
                           OD14  2293          .IF DEFINED LT_CRED_CHECK
                           OD14  2294          CMPB    UCB$B_LT_XCRED(R5),-    ; Did we goof?
                           OD14  2295            #-LAT$C_MAX_RCRED
                           OD14  2296          BGEQU   10$                     ; Br if okay
                           OD14  2297          DEBUG                           ; Else, we went too far
                           OD14  2298 10$:
                           OD14  2299          .ENDC   ;; DEFINED LT_CRED_CHECK
                           OD14  2300
      6B   5C    28    OD14  2301          MOVC3   AP,(R11),INB_C_LENGTH(R2) ; Copy the data
        0C A2            OD17
          37   BA    OD19  2302 90$:     POPR    #^M<R0,R1,R2,R4,R5>     ; Restore registers
               05    OD1B  2303          RSB                             ; Return to caller
                     OD1C  2304
```

```
                        .SBTTL  LT$FFI_RCV_MSG - FAST Interface Receive Complete routine
                ;++
                ; LT$FFI_RCV_MSG - FAST Interface Receive Complete routine
                ;
                ; Functional description:
                ;
                ; Each time a frame is received, the data is passed to the class driver
                ; for each terminal represented within the frame. Echoed data is copied
                ; into a frame to be transmitted back to the concentrator, and finally
                ; the transmit frame is queued for transmission after appropriate
                ; virtual circuit maintenance functions are performed.
                ;
                ; Inputs:
                ;
                ;       R3 = CXB address
                ;       R4 = FFI address
                ;
                ;       IPL = SYNCH
                ;
                ; Outputs:
                ;
                ;       R1-R5 are preserved.
                ;
                ;       The interrupt is dismissed when the received frame
                ;       has been compeletely processed. The CXB is always
                ;       returned to the datalink driver.
                ;
                ;--
                ;
                        ASSUME  LAT$C_IPL EQ IPL$_SYNCH
                LT$FFI_RCV_MSG:
3FFE 8F     BB          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP> ; Save registers
                ;
  56   18 A3   3C          MOVZWL  CXB$W_BOFF(R3),R6       ; Get offset to start of data
    56   53   C0          ADDL    R3,R6                   ; Get pointer to start of data
                ;
                        .IF DEFINED LT_HISTORY
                ;
                ;       Save history of receives
                ;
    50   66   9E          MOVAB   RCV_B_FLAG(R6), R0      ; Address is circuit header
       F9FA   30          BSBW    LT$HISTORY              ; Store the data
                        .ENDC   ;; DEFINED LT_HISTORY
                ;
                        INC_CTR GHB$L_RCOUNT+GHB_AREA   , Count frame
                ;
                ;  Validate received frame
                ;
  50   02 A6   3C          MOVZWL  RCV_W_DSTID(R6),R0      ; Get remote vci
         76   12          BNEQ    120$                    ; Got index, okay
              91          CMPB    #<MTYP_C_START@FLAG_V_MTYPE>!- ; Is this a start message?
                                  FLAG_M_MASTER,-
  66   06                         RCV_B_FLAG(R6)
         29   12          BNEQ    91$                     ; Br if no, signal protocol error
                ;
                ;  START message received on an inactive circuit, try to start up a
                ;  new circuit.
```

```
                     0D42  2363 ;
F3EE CF    D5        0D42  2364          TSTL    GHB$L_UCB+GHB_AREA      ; Data link UCB still here?
     4F    13        0D46  2365          BEQL    99$                     ; No, then don't accept circuit
  04 A6    B5        0D48  2366          TSTW    RCV_W_SRCID(R6)         ; Is the source ID zero?
     32    13        0D4B  2367          BEQL    96$                     ; Br if yes, protocol error
  06 A6    95        0D4D  2368          TSTB    RCV_B_SEQ(R6)           ; Is this the correct sequenced msg?
     32    12        0D50  2369          BNEQ    97$                     ; Br if not, protocol error
   057B    30        0D52  2370          BSBW    LT$NEW_CIRCUIT          ; Initialize and/or allocate CSB
  10 50    E8        0D55  2371          BLBS    R0, 19$                 ; Go ahead, CSB address in R8
                     0D58  2372                                          ;  startup any transmits waiting
     50    D5        0D58  2373          TSTL    R0                      ; Are we ignoring a duplicate start msg?
     40    12        0D5A  2374          BNEQ    100$                    ; Br if yes, just retransmit start msg
                     0D5C  2375          INC_CTR GHB$L_RESOURCE+GHB_AREA ; Else, no CSB to use
     2F    11        0D66  2376          BRB     99$                     ; So just ignore it
                     0D68  2377
   00B5    31        0D68  2378 19$:     BRW     190$                    ; Long branch
                     0D68  2379
5D   00    9A        0D68  2380 91$:     MOVZBL  #GHB$V_START,FP         ; Invalid START message received
     17    11        0D6E  2381          BRB     98$                     ; continue
5D   02    9A        0D70  2382 93$:     MOVZBL  #GHB$V_CSBRANGE,FP      ; Bad range on CSB index
     12    11        0D73  2383          BRB     98$                     ; continue
5D   03    9A        0D75  2384 94$:     MOVZBL  #GHB$V_CSBINVALID,FP    ; Invalid CSB index
     0D    11        0D78  2385          BRB     98$                     ; continue
5D   04    9A        0D7A  2386 95$:     MOVZBL  #GHB$V_CSBSTALE,FP      ; Stale CSB index
     08    11        0D7D  2387          BRB     98$                     ; continue
5D   06    9A        0D7F  2388 96$:     MOVZBL  #GHB$V_INVALIDREMID,FP  ; Invalid remote circuit id
     03    11        0D82  2389          BRB     98$                     ; continue
5D   0A    9A        0D84  2390 97$:     MOVZBL  #GHB$V_INVALIDSEQ,FP    ; Invalid sequence number
                     0D87  2391 ...      BRB     98$                     ; continue
                     0D87  2392 98$:     SETBIT  FP,GHB$L_PROTOMASK+GHB_AREA ; Set bit mask
                     0D8D  2393          INC_CTR GHB$L_PROTOCOL+GHB_AREA ; Increment counter
3FFE 8F    BA        0D97  2394 99$:     POPR    #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP> ; Restore registers
           05        0D98  2395          RSB                             ; Return to caller
                     0D9C  2396
                     0D9C  2397          ;
                     0D9C  2398          ;  A duplicate frame was received
                     0D9C  2399          ;
                     0D9C  2400 100$:    INC_CTR GHB$L_DUPLMSG+GHB_AREA  ; We received a duplicate message
                     0DA6  2401          INC_CTR GHB$L_RETRANS+GHB_AREA  ; We are retransmitting a message
         04B2    31  0DB0  2402          BRW     LT$REPLY_REXMIT         ; This data is stale, resend buffers
                     0DB3  2403 ;
                     0DB3  2404 ;
                     0DB3  2405 ;  Non-zero circuit ID !
                     0DB3  2406 ;        - circuit must be active already
                     0DB3  2407 ;
                     0DB3  2408 ;  Inputs:       R0 = CSB index
                     0DB3  2409 ;                R3 = CXB address
                     0DB3  2410 ;                R6 = Pointer to start of data
                     0DB3  2411 ;
                     0DB3  2412 ;
      51   50    9A  0DB3  2413 120$:    MOVZBL  R0,R1                   ; Get CSB index
      20   51    D1  0DB6  2414          CMPL    R1, #LAT$C_MAX_CSBS     ; Range check
           B5    14  0DB9  2415          BGTR    93$                     ; Br if bad range, protocol error
58 F274 CF41    D0  0DBB  2416          MOVL    GHB_L_CSBTABLE-4[R1],R8 ; Get CSB address
           B2    13  0DC1  2417          BEQL    94$                     ; Br if invalid CSB, protocol error
   60 A8   50    B1  0DC3  2418          CMPW    R0,CSB_W_LOCID(R8)      ; Valid reference ?
           B1    12  0DC7  2419          BNEQ    95$                     ; Br if no, Stale reference to CSB
```

```
                        0DC9  2420          INC_CTR CSB_Z_LCB+LCB_L_MSG_RCV(R8) ; Count one more rcvd msg
                        0DD1  2421          ;
                        0DD1  2422          ;  Check ACK number and complete any transmits acknowledged
                        0DD1  2423          ;
      53    07 A6   9A  0DD1  2424 130$:    MOVZBL  RCV_B_ACK(R6), R3       ; Get the ack from the message
   65 A8    53      90  0DD5  2425          MOVB    R3, CSB_B_RACK(R8)      ;
      53    62 A8   82  0DD9  2426          SUBB    CSB_B_XSEQ(R8), R3      ; Get up the normallized ack
      52    3C A8   7E  0DDD  2427          MOVAQ   CSB_Q_XWAITQ(R8), R2    ; Look through the wait queue for it
            51 52   D0  0DE1  2428          MOVL    R2, R1                  ;
                        0DE4  2429
            51 61   D0  0DE4  2430 140$:    MOVL    (R1), R1                ; Next item in queue
            52 51   D1  0DE7  2431 150$:    CMPL    R1, R2                  ; End of queue
               34   13  0DEA  2432          BEQL    190$                    ; Done with that search
4E A1         62 A8   83  0DEC  2433          SUBB3   CSB_B_XSEQ(R8), -       ; Normallize the seq of message
                  50    0DF1
                        0DF2  2434                  XMT_B_SEQ(R1), -        ;
                        0DF2  2435                  R0                      ;
            53 50   91  0DF2  2436          CMPB    R0, R3                  ; If the seq is greater than ack,
               ED   14  0DF5  2437          BGTR    140$                    ; wait on another ack
            54 61   D0  0DF7  2438          MOVL    (R1), R4                ; Save link to next item
            51 61   0F  0DFA  2439          REMQUE  (R1), R1                ; Remove acked message from wait queue
   38 B8    61      0E  0DFD  2440          INSQUE  (R1), -                 ; And insert on tail of transmit queue
                        0E01  2441                  @CSB_Q_XBUFQ+4(R8)      ;
         59 A8   97     0E01  2442          DECB    CSB_B_XMTCNT(R8)        ; One less to transmit
            51 54   D0  0E04  2443          MOVL    R4, R1                  ; Restore link
               DE   11  0E07  2444          BRB     150$                    ;
                        0E09  2445 170$:    ;
                        0E09  2446          ;  Invalid sequence number received
                        0E09  2447          ;
                        0E09  2448          INC_CTR CSB_Z_LCB+LCB_W_SEQ_ERR(R8),W ; Bad sequence number
                  89    11  0E11  2449          BRB     100$                    ; Also count as duplicate
                        0E13  2450 180$:    ;
                        0E13  2451          ;  No transmit buffer available
                        0E13  2452          ;
                        0E13  2453          INC_CTR GHB$L_NOXBFR+GHB_AREA   ; Not transmit buffer available
                  FF77  31    0E1D  2454          BRW     99$                     ; And exit
                        0E20  2455          ;
                        0E20  2456          ;  Make sure that we have a transmit buffer to respond with
                        0E20  2457          ;
      52    34 A8   7E  0E20  2458 190$:    MOVAQ   CSB_Q_XBUFQ(R8), R2     ; Look to see if we have at least one
         62    52   D1  0E24  2459          CMPL    R2, (R2)                ; transmit buffer
               EA   13  0E27  2460          BEQL    180$                    ; Br if None, just retransmit and wait
                        0E29  2461
            66 A8   B0  0E29  2462          MOVW    CSB_W_TIMRESET(R8),-    ; Reset timer for timeout of circuit
               50 A8        0E2C  2463                  CSB_W_TIMEOUT(R8)       ; Since we got a valid message.
64 A8       06 A6   91  0E2E  2464          CMPB    RCV_B_SEQ(R6),CSB_B_RSEQ(R8) ; Right sequence number ?
               D4   12  0E33  2465          BNEQ    170$                    ; Br if no, retransmit all messages
                        0E35  2466
                        0E35  2467
                        0E35  2468 ;
                        0E35  2469 ;  Dispatch on message type
                        0E35  2470 ;
                        0E35  2471 ;  Inputs:
                        0E35  2472 ;
                        0E35  2473 ;       R8 = CSB address
                        0E35  2474 ;       R6 = CXB address
                        0E35  2475 ;
```

```
                64 A8   96  0E35  2476          INCB    CSB_B_RSEQ(R8)              ; Next time it's the next number
        63 A8   06 A6   90  0E38  2477          MOVB    RCV_B_SEQ(R6),CSB_B_XACK(R8) ; Tell other guy we got this one
                   02   EF  0E3D  2478          EXTZV   #FLAG_V_MTYPE,-             ; Obtain the message type
                   06       0E3F  2479                  #FLAG_S_MTYPE,-
                50 66       0E40  2480                  RCV_B_FLAG(R6), R0
                            0E42  2481
                            0E42  2482          $DISPATCH       R0,TYPE=B,-         ; Dispatch on message type
                            0E42  2483          <-          ; msg type          ; action
                            0E42  2484                  <MTYP_C_RUN     LT$RCV_RUN_MSG>,- ; RUN message received
                            0E42  2485                  <MTYP_C_START   LT$RCV_START_MSG>,- ; START message received
                            0E42  2486                  <MTYP_C_HALT    LT$RCV_HALT_MSG>,- ; HALT message received
                            0E42  2487          >
                            0E4C  2488
                            0E4C  2489 ;;       BRB     LT$FORCE_HALT               ; All others, force halt
```

```
                     0E4C  2491          .SBTTL  PROCESS RECEIVED MESSAGE
                     0E4C  2492 ;++
                     0E4C  2493 ;
                     0E4C  2494 ;  Other guy is halted or all screwed up
                     0E4C  2495 ;
                     0E4C  2496 ;--
                     0E4C  2497
                     0E4C  2498 LT$FORCE_HALT:          ; receive halt msg
                     0E4C  2499                         ; invalid circuit state in msg
                     0E4C  2500                         ; invalid circuit state in CSB
                     0E4C  2501                         ; invalid local vci
                     0E4C  2502          SETBIT  #GHB$V_HALT,-
                     0E4C  2503                  GHB$L_PROTOMASK+GHB_AREA ; Circuit forced to halted
                     0E52  2504          INC_CTR GHB$L_PROTOCOL+GHB_AREA ; Increment protocol error counter
                     0E5C  2505
                     0E5C  2506 LT$STATE_HALT:
08CB   30            0E5C  2507          BSBW    LT$CIRCDEAD             ; Circuit is dead now
0403   31            0E5F  2508          BRW     LT$REPLY_REXMIT         ; Transmit halt message
                     0E62  2509
                     0E62  2510 ;++
                     0E62  2511 ;
                     0E62  2512 ;  Other system is halted
                     0E62  2513 ;
                     0E62  2514 ;--
                     0E62  2515
                     0E62  2516 LT$RCV_HALT_MSG:
08C5   30            0E62  2517          BSBW    LT$CIRCDEAD             ; Circuit is dead now
03F8   31            0E65  2518          BRW     LT$REPLY_DONE           ; All done
                     0E68  2519
                     0E68  2520 ;++
                     0E68  2521 ;
                     0E68  2522 ;  Other guy is starting up
                     0E68  2523 ;
                     0E68  2524 ;--
                     0E68  2525
                     0E68  2526 LT$RCV_START_MSG:
                     0E68  2527
                     0E68  2528          $DISPATCH CSB_B_STATE(R8),TYPE=B,- ; Dispatch on our CSB state
                     0E68  2529          <- ; state              ; action
                     0E68  2530          <CSB_C_STATE_HALT       LT$STATE_START>,- ; Enter the START state
                     0E68  2531          <CSB_C_STATE_START      LT$REPLY_MSG>,-  ; Re-send START message
                     0E68  2532          <CSB_C_STATE_RUN        LT$STATE_HALT>,- ; Enter the HALTED state
                     0E68  2533          >
                     0E73  2534
D7     11            0E73  2535          BRB     LT$FORCE_HALT           ; What kind of kinky state is this ?
                     0E75  2536
                     0E75  2537 ;++
                     0E75  2538 ;
                     0E75  2539 ;  Other guy is running
                     0E75  2540 ;
                     0E75  2541 ;--
                     0E75  2542
                     0E75  2543 LT$RCV_RUN_MSG:
                     0E75  2544
                     0E75  2545          $DISPATCH       CSB_B_STATE(R8),TYPE=B,- ; Dispatch on our CSB state
                     0E75  2546          <-      ; state                 ; action
                     0E75  2547                  <CSB_C_STATE_HALT       LT$REPLY_MSG>,- ; Send STOP message
```

```
                0E75  2548                <CSB_C_STATE_START      LT$STATE_ENTER_RUN>,- ; Enter RUN state
                0E75  2549                <CSB_C_STATE_RUN        LT$STATE_RUN>,- ; Stay in RUN state
                0E75  2550          >
                0E80  2551
CA   11         0E80  2552          BRB     LT$FORCE_HALT           ; What kind of kinky state is this ?
```

```
                                0E82  2554          .SBTTL  CIRCUIT STATE TRANSITIONS
                                0E82  2555 ;++
                                0E82  2556 ;
                                0E82  2557 ;  We can go only from halted to starting
                                0E82  2558 ;
                                0E82  2559 ;--
                                0E82  2560
                                0E82  2561 LT$STATE_START:
          01    90              0E82  2562          MOVB    #CSB_C_STATE_START,-    ;
      0B  A8                    0E84  2563                  CSB_B_STATE(R8)         ; Now we are starting
          04    90              0E86  2564          MOVB    #MTYP_C_START@FLAG_V_MTYPE,- ; Set the start message type
      5C  A8                    0E88  2565                  CSB_B_FLAG(R8)          ;
      04  A6    B0              0E8A  2566          MOVW    RCV_W_SRCID(R6),-       ;
      5E  A8                    0E8D  2567                  CSB_W_REMID(R8)         ; Save his circuit id
  59  34  A8    D0              0E8F  2568          MOVL    CSB_Q_XBUFQ(R8),R9      ; Format a start message
50  F569  CF    9E              0E93  2569          MOVAB   GHB_T_STRT_MSG,R0       ; Get start address of template data
60  F562  CF    28              0E98  2570          MOVC3   GHB_W_STRT_LEN,(R0),-   ; Copy the start message
      50  A9                    0E9D  2571                  XMT_T_MDATA(R9)
    1C A9     53    D0          0E9F  2572          MOVL    R3,CXB$L_T_ENDADR(R9)   ; Store end address of data
        03B4    31              0EA3  2573          BRW     LT$REPLY_ALT            ; Go send the reply message
                                0EA6  2574
                                0EA6  2575 ;++
                                0EA6  2576 ;
                                0EA6  2577 ;  We transit to the running state only from the starting state
                                0EA6  2578 ;
                                0EA6  2579 ;--
                                0EA6  2580
                                0EA6  2581 LT$STATE_ENTER_RUN:                      ; Enter the RUNNING state
          02    90              0EA6  2582          MOVB    #CSB_C_STATE_RUN,-      ; Now we are running
      0B  A8                    0EA8  2583                  CSB_B_STATE(R8)
          00    90              0EAA  2584          MOVB    #MTYP_C_RUN@FLAG_V_MTYPE,- ; Set the run message type
      5C  A8                    0EAC  2585                  CSB_B_FLAG(R8)
                                0EAE  2586
                                0EAE  2587 ;++
                                0EAE  2588 ;
                                0EAE  2589 ;  Update state from validated frame
                                0EAE  2590 ;
                                0EAE  2591 ;--
                                0EAE  2592
                                0EAE  2593 LT$STATE_RUN:
      04  A6    B1              0EAE  2594          CMPW    RCV_W_SRCID(R6),-       ; Referencing a valid circuit?
      5E  A8                    0EB1  2595                  CSB_W_REMID(R8)
          97    12              0EB3  2596          BNEQ    LT$FORCE_HALT           ; Br if not, force a halt
59    34  A8    D0              0EB5  2597          MOVL    CSB_Q_XBUFQ(R8),R9      ; Get transmit buffer address
                                0EB9  2598          ;
                                0EB9  2599          ;  Set up for the receive loop
                                0EB9  2600          ;
                                0EB9  2601          CLRBIT  #FLAG_V_RRF,-           ; No response requested until
                                0EB9  2602                  CSB_B_FLAG(R8)          ;  a credit is used
      5D  A8    94              0EBE  2603          CLRB    CSB_B_NUM_SLOTS(R8)     ; Assume no output data (balanced)
      52  A8    B4              0EC1  2604          CLRW    CSB_W_XMTTMO(R8)        ; Circuit state is no longer timed
57    08  A6    9E              0EC4  2605          MOVAB   RCV_T_MDATA(R6),R7      ; Receive buffer first slot address
59    50  A9    9E              0EC8  2606          MOVAB   XMT_T_MDATA(R9),R9      ; Transmit buffer first slot address
5A    6A  A8    3C              0ECC  2607          MOVZWL  CSB_W_MAX_MSGSIZ(R8),R10 ; Total chars available to slots
5D    01  A6    9A              0ED0  2608          MOVZBL  RCV_B_NUM_SLOTS(R6),FP  ; Get slot loop count
          03    12              0ED4  2609          BNEQ    SLOT_LOOP_IN            ; Got some input data
        016B    31              0ED6  2610          BRW     SLOT_LOOP_OUT           ; Go process output buffer
```

0ED9 2611

```
                        0ED9  2613          .SBTTL  PROCESS RECEIVED SLOT DATA
                        0ED9  2614 ;++
                        0ED9  2615 ; PROCESS RECEIVED SLOT DATA
                        0ED9  2616 ;
                        0ED9  2617 ; Functional description:
                        0ED9  2618 ;
                        0ED9  2619 ; Now we are set up to pass data to and from the class driver:
                        0ED9  2620 ;
                        0ED9  2621 ; The following loop copies data from the receive buffer into the class
                        0ED9  2622 ; driver.
                        0ED9  2623 ;
                        0ED9  2624 ; Inputs:
                        0ED9  2625 ;       R0-R4 available
                        0ED9  2626 ;       R5 =  UCB address
                        0ED9  2627 ;       R6 =  CXB address
                        0ED9  2628 ;       R7 =  current receive buffer slot address
                        0ED9  2629 ;       R8 =  CSB address
                        0ED9  2630 ;       R9 =  current transmit buffer slot address
                        0ED9  2631 ;       R10 = transmit buffer bytes left
                        0ED9  2632 ;       R11 = scratch (used as current input slot character copy base register)
                        0ED9  2633 ;       AP =  scratch (used as loop count register)
                        0ED9  2634 ;       FP =  number of slots to process in received message
                        0ED9  2635 ;
                        0ED9  2636 ; Outputs:
                        0ED9  2637 ;       none.
                        0ED9  2638 ;
                        0ED9  2639 ;  We continue by processing the output slot loop.
                        0ED9  2640 ;
                        0ED9  2641 ;--
                        0ED9  2642          .ENABLE LOCAL_BLOCK
                        0ED9  2643 SLOT_LOOP_IN:                                ; Input data slot loop
              51  67 9A 0ED9  2644          MOVZBL  SLT_B_DSTID(R7),R1         ; Is this slot alive ?
                 3D 12  0EDC  2645          BNEQ    20$                        ; Br if possible
                01 A7 95  0EDE  2646          TSTB    SLT_B_SRCID(R7)            ; Is the remote index valid?
                 35 13  0EE1  2647          BEQL    10$                        ; Br if not, ignore slot
                        0EE3  2648          ;
                        0EE3  2649          ;  Zero local slot index with non-zero remote (someone's attempting
                        0EE3  2650          ;    to login). Must be a start slot!
                        0EE3  2651          ;
              04  04 EF 0EE3  2652          EXTZV   #4,#4,SLT_B_CRED(R7),R0 ; Get slot type
           50  03 A7    0EE6
              50  09 91  0EE9  2653          CMPB    #SLT_C_STR_SLOT,R0         ; Is this a start slot ?
                 57 12  0EEC  2654          BNEQ    LOOP_ABORT                 ; Br if not, ignore slot
                0656 30  0EEE  2655          BSBW    LT$ALC_UCB                 ; Try to allocate an LT UCB
                24 50 E8 0EF1  2656          BLBS    R0, 10$                    ; Br if success
                        0EF4  2657          INC_CTR GHB$L_RESOURCE+GHB_AREA ; Else, no UCB for us to use
                        0EFE  2658          ;
                        0EFE  2659          ;  Send back a reject slot in the case of resource errors
                        0EFE  2660          ;
;JAY0002         04  5A D1  0EFE    .1          CMPL    R10,-                      ; Enough room for REJECT slot header?
;JAY0002                    0F01    .2                  #SLT_S_REJ_SLOT
;JAY0002            15 19  0F01    .3          BLSS    10$                        ; Nope, just ignore the data
;JAY0002         5D A8 96  0F03    .4          INCB    CSB_B_NUM_SLOTS(R8)        ; Increment slot count
;JAY0002      89 01 A7 90  0F06    .5          MOVB    SLT_B_SRCID(R7),(R9)+      ; Copy local id from receive message
;JAY0002            89 94  0F0A    .6          CLRB    (R9)+                      ; Tell him we are not here !
;JAY0002            89 94  0F0C    .7          CLRB    (R9)+                      ; Load status byte count
;JAY0002      69 C0 8F 90  0F0E    .8          MOVB    #SLT_C_REJ_SLOT@4,(R9)     ; Load REJECT slot format
```

```
:JAY0002        89   05   88  0F12     .9          BISB    #SLT_C_RESOURCE,(R9)+   ; Load reason code (resource error)
:JAY0002        5A   04   C2  0F15     .10         SUBL2   #SLT_S_REJ_SLOT,R10     ; Adjust bytes left for other slots
-13                00EC   31  0F18  2674 10$:      BRW     NEXT_SLOT               ; Just process the next slot.
                          0F1B  2675 20$:      ;
                          0F1B  2676           ;  Validate UCB reference - non-zero local index
                          0F1B  2677           ;
                    69 A8   51   91  0F1B  2678          CMPB    R1, CSB_B_MAX_SLOTS(R8) ; Do a range check on the local index
                           58   1A  0F1F  2679          BGTRU   90$                     ; Br if not in range
                    55  68 A841 D0  0F21  2680          MOVL    CSB_L_UCBLST-4(R8)[R1],R5 ; Get UCB address
                           2D   18  0F26  2681          BGEQ    70$                     ; that slot doesn't live here anymore
                        01 A7   91  0F28  2682          CMPB    SLT_B_SRCID(R7),UCB$B_LT_REMID(R5) ; Valid remote reference?
                      0140 C5           0F2B
                           63   13  0F2E  2683          BEQL    LOAD_CREDITS            ; Br if yes, process slot
                        01 A7   95  0F30  2684          TSTB    SLT_B_SRCID(R7)         ; Else, are we hanging up from remote?
                           10   12  0F33  2685          BNEQ    LOOP_ABORT              ; Br if no, invalid slot
                                0F35  2686 ;
                                0F35  2687 ;  Received a zero in the slot source-id - is this a stop slot?
                                0F35  2688 ;
                    04   04   EF  0F35  2689 40$:       EXTZV   #4,#4,SLT_B_CRED(R7),R0 ; Get slot type
                 50     03 A7        0F38
                    50     0D   91  0F3B  2690          CMPB    #SLT_C_STP_SLOT,R0      ; Is this a STOP slot?
                           05   12  0F3E  2691          BNEQ    LOOP_ABORT              ; Br if not, do nothing
                         0740   30  0F40  2692          BSBW    LT$HANGUP_UCB_NOW       ; Hangup the UCB to logout process
                           4B   11  0F43  2693          BRB     NEXT_SLOT0              ; Process next slot
                                0F45  2694 ;
                                0F45  2695 ;  Flag protocol error and ignore slot
                                0F45  2696 ;
                                0F45  2697 LOOP_ABORT:
                                0F45  2698          SETBIT  #GHB$V_INVALIDLOCID,-
                                0F45  2699                  GHB$L_PROTOMASK+GHB_AREA ; Invalid remote ID or slot format
                                0F4B  2700          INC_CTR GHB$L_PROTOCOL+GHB_AREA ;
                                0F55  2701
                                0F55  2702 ;
                                0F55  2703 ;  Conditionally send stop slot
                                0F55  2704 ;
                        01 A7   95  0F55  2705 70$:      TSTB    SLT_B_SRCID(R7)         ; Check remote slot index
                           36   13  0F58  2706          BEQL    NEXT_SLOT0              ; Br if empty slot
                                0F5A  2707
                                0F5A  2708          ASSUME  SLT_S_STP_SLOT&1 EQ 1   ; Assume we need rounding
                        06   5A   D1  0F5A  2709          CMPL    R10,-                   ; Enough room for STOP slot header?
                                0F5D  2710                  #SLT_S_STP_SLOT+1
                           31   19  0F5D  2711          BLSS    NEXT_SLOT0              ; Nope, just ignore the data
                        5D A8   96  0F5F  2712          INCB    CSB_B_NUM_SLOTS(R8)     ; Increment slot count
                    89  01 A7   90  0F62  2713          MOVB    SLT_B_SRCID(R7),(R9)+   ; Copy local id from receive message
                           89   94  0F66  2714          CLRB    (R9)+                   ; Tell him we are not here !
                        89  01   90  0F68  2715          MOVB    #SLT_S_STP_LEN,(R9)+    ; Load slot byte count
                    89     D0 8F   90  0F6B  2716          MOVB    #SLT_C_STP_SLOT&^X4,(R9)+ ; Load stop slot format
                        89  03   90  0F6F  2717          MOVB    #SLT_C_INVSLOT,(R9)+    ; load reason code
                                0F72  2718          ASSUME  SLT_S_STP_SLOT&1 EQ 1   ; Assume we need rounding
                           89   94  0F72  2719          CLRB    (R9)+                   ; round up to even boundary
                        5A   06   C2  0F74  2720          SUBL2   #SLT_S_STP_SLOT+1,R10   ; Adjust bytes left for other slots
                           17   11  0F77  2721          BRB     NEXT_SLOT0              ; Go do next input slot
                                0F79  2722 ;
                                0F79  2723 ;  Range check error, is this really a disconnect slot ?
                                0F79  2724 ;
                        01 A7   95  0F79  2725 90$:      TSTB    SLT_B_SRCID(R7)         ; Empty remote slot index? (stop slot?)
                           12   13  0F7C  2726          BEQL    NEXT_SLOT0              ; Br if yes, ignore the slot.
```

```
                     0F7E  2727          SETBIT  #GHB$V_INVALIDREMID,-
                     0F7E  2728                  GHB$L_PROTOMASK+GHB_AREA; Invalid LOCID with REMID non_zero
                     0F84  2729          INC_CTR GHB$L_PROTOCOL+GHB_AREA ; Count as protocol error
                  C5 11  0F8E  2730          BRB     70$                     ; Send a stop slot
                     0F90  2731
                     0F90  2732 NEXT_SLOT0:
                0074 31  0F90  2733          BRW     NEXT_SLOT               ; Branch aid
                     0F93  2734
                     0F93  2735          .DISABLE LOCAL_BLOCK
```

```
                     0F93  2737          .SBTTL  PROCESS SLOT CREDITS RECEIVED
                     0F93  2738 ;++
                     0F93  2739 ;
                     0F93  2740 ;  Update UCB credit account from received message slots
                     0F93  2741 ;
                     0F93  2742 ;--
                     0F93  2743 LOAD_CREDITS:
50   03 A7   90      0F93  2744          MOVB    SLT_B_CRED(R7),R0       ; Get credits and slot type
                     0F97  2745          ASSUME  SLT_C_DA_SLOT EQ 0      ; assumption
     2C      18      0F97  2746          BGEQ    50$                     ; Got a DATA_A slot
04   00      EF      0F99  2747          EXTZV   #0, #4, SLT_B_CRED(R7), R0 ; Get the credit counts
50   03 A7
                     0F9C
04   04      EF      0F9F  2748          EXTZV   #4, #4, SLT_B_CRED(R7), R1 ; Get the slot type
51   03 A7
                     0FA2
                     0FA5  2749          $DISPATCH       R1,TYPE=B,-     ; Dispatch on slot type
                     0FA5  2750          <-      ; slot type     ; action
                     0FA5  2751                  <SLT_C_DB_SLOT  20$>,-  ; DATA_B slot????
                     0FA5  2752                  <SLT_C_ATT_SLOT 40$>,-  ; Attention slot
                     0FA5  2753          >
                     0FAD  2754
     FF95    31      0FAD  2755          BRW     LOOP_ABORT              ; All others, including Reject slot
                     0FB0  2756
                     0FB0  2757 ;  DATA_B slot
                     0FB0  2758
     01      88      0FB0  2759 20$:     BISB    #UCB$M_LT_DATA,-        ; Remember to at least send back the
0142 C5              0FB2  2760           UCB$B_LT_DATAW(R5)             ;    credits!
0147 C5      96      0FB5  2761          INCB    UCB$B_LT_XCRED(R5)      ; One less credit on remote end (negative)
     03      15      0FB9  2762          BLEQ    40$                     ; Br if credits not exceeded, proceed
   06D2      30      0FBB  2763          BSBW    LT$HANGUP_UCB           ; Else, Hangup the UCB to logout process
                     0FBE  2764
                     0FBE  2765 ;  ATTention slot, IGNORE DATA
                     0FBE  2766
0146 C5      50      0FBE  2767 40$:     ADDB    R0,UCB$B_LT_TCRED(R5)   ; Update credits
             42  11  0FC3  2768          BRB     NEXT_SLOT               ; Skip to next slot
                     0FC5  2769
                     0FC5  2770 ;  RUN slot
                     0FC5  2771
0146 C5      50      0FC5  2772 50$:     ADDB    R0,UCB$B_LT_TCRED(R5)   ; Update credits
                     0FCA  2773
                     0FCA  2774
                     0FCA  2775          .IF DEFINED LT_CRED_CHECK
                     0FCA  2776          CMPB    UCB$B_LT_TCRED(R5),#2
                     0FCA  2777          BLEQ    60$                     ; Br if ok
                     0FCA  2778          SETBIT  #GHB$V_BADCREDITS,-
                     0FCA  2779                  GHB$L_PROTOMASK+GHB_AREA, Too many credits in host
                     0FCA  2780          INC_CTR GHB$L_PROTOCOL+GHB_AREA ; Too many credits in host
                     0FCA  2781          BSBW    LT$HANGUP_UCB           ; Hangup the UCB to logout process
                     0FCA  2782 60$:
                     0FCA  2783          .ENDC   ;; DEFINED LT_CRED_CHECK
                     0FCA  2784
                     0FCA  2785 ;
                     0FCA  2786 ;  Move terminal data through class driver
                     0FCA  2787 ;
                     0FCA  2788 ;       R3 - contains the character
                     0FCA  2789 ;       R5 - UCB address
                     0FCA  2790 ;
5C   02 A7   9A      0FCA  2791          MOVZBL  SLT_B_COUNT(R7),AP      ; Character count for this slot
```

```
            37   13 0FCE          BEQL    NEXT_SLOT                      ; No input here
      5B 04 A7   9E 0FD0          MOVAB   SLT_T_DATA(R7),R11             ; Source of characters
   0143 C5  00   E0 0FD4          BBS     #UCB$V_LT_HANGUP, -            ; If device is hanging up, do not con-
                 2D 0FD9
                    0FDA                  UCB$B_LT_LATSTS(R5),NEXT_SLOT ; fuse class driver with more data
       0147 C5   96 0FDA          INCB    UCB$B_LT_XCRED(R5)             ; One less credit on remote end (negative)
            05   15 0FDE          BLEQ    70$                            ; Br if credits not exceeded, proceed
          06AD   30 0FE0          BSBW    LT$HANGUP_UCB                  ; Else, Hangup the UCB to logout process
            22   11 0FE3          BRB     NEXT_SLOT                      ; Continue
                    0FE5 70$:     ;
                    0FE5          ;  Indicate that input stream is running again, and data is potentially
                    0FE5          ;  available.
                    0FE5          ;
                    0FE5          ASSUME  UCB$B_LT_LATSTS EQ UCB$B_LT_DATAW+1
                    0FE5
       1001 8F   A8 0FE5          BISW    #<UCB$M_LT_INPUT@8>!UCB$M_LT_DATA,- ; Data potentially available
       0142 C5      0FE9                   UCB$B_LT_DATAW(R5)        ;   and INPUT stream in enabled
      64 A5 02   88 0FEC          BISB    #UCB$M_INT,UCB$L_STS(R5); Force no data to be returned
                    0FF0          ;
                    0FF0          ;  Dump characters into class driver
                    0FF0          ;
         53 8B   9A 0FF0 80$:     MOVZBL  (R11)+,R3                      ; Get next character
            02   E0 0FF3          BBS     #UCB$V_LT_OVFLOW,-             ; Br if overflow, wait for XON
    09 0143 C5      0FF5                   UCB$B_LT_LATSTS(R5),90$
       0110 D5   16 0FF9          JSB     @UCB$L_TT_PUTNXT(R5)           ; Pass character to class driver
            1B   12 0FFD          BNEQ    ECHO_ERROR                     ; Br if character being echoed
         EE 5C   F5 0FFF          SOBGTR  AP,80$                         ; Do all characters
                    1002
            10   8A 1002 90$:     BICB    #UCB$M_LT_INPUT,-              ; Input stream is now disabled
       0143 C5      1004                   UCB$B_LT_LATSTS(R5)
                    1007 ;
                    1007 ;  Bump pointers to the next slot
                    1007 ;
                    1007 NEXT_SLOT:
      50 02 A7   9A 1007          MOVZBL  SLT_B_COUNT(R7), R0            ; Max bytes in the slot
    05 A740      9E 100B          MOVAB   <SLT_T_DATA+1>(R7)-            ; Compute address of next slot +1
            57      100F                   [R0], R7                      ; to help round to word
         57 01   CA 1010          BICL    #1, R7                         ; make it a word boundary.
            5D   D7 1013          DECL    FP                             ; Loop on all slots
            2D   13 1015          BEQL    SLOT_LOOP_OUT                  ; Done all input slots, now process
                    1017                                                 ;  any output data
          FEBF   31 1017          BRW     SLOT_LOOP_IN                   ; Process all received slots
                    101A
                    101A ECHO_ERROR:                                     ; Character being echoed???
                    101A          DEBUG                                  ;;&& Should not get to here
```

```
                     1020  2837          .SBTTL  BUILD A STOP SLOT
                     1020  2838 ;++
                     1020  2839 ; BUILD A STOP SLOT
                     1020  2840 ;
                     1020  2841 ; Functional description:
                     1020  2842 ;
                     1020  2843 ; Build a stop slot in the out message buffer, then delete the UCB.
                     1020  2844 ;
                     1020  2845 ; Inputs:
                     1020  2846 ;       R5  = UCB address
                     1020  2847 ;       R8  = CSB address
                     1020  2848 ;       R9  = current transmit buffer slot address
                     1020  2849 ;       R10 = transmit buffer bytes left
                     1020  2850 ;
                     1020  2851 ; Outputs:
                     1020  2852 ;       R5  = UCB address
                     1020  2853 ;       R8  = CSB address
                     1020  2854 ;       R9  = current transmit buffer slot address
                     1020  2855 ;       R10 = transmit buffer bytes left
                     1020  2856 ;
                     1020  2857 ;--
                     1020  2858
                     1020  2859 SEND_STOP_SLOT:
014B C5    95        1020  2860          TSTB    UCB$B_LT_CURC(R5)       ; Is UCB data waiting?
       03  13        1024  2861          BEQL    30$                     ; Br if no, send the STOP slot
    00B9   31        1026  2862          BRW     CREDIT_CHECK            ; Send data if our credit is good
                     1029  2863
04   5A    D1        1029  2864 30$:    CMPL    R10, #SLT_T_DATA        ; Enough room for STOP slot?
       1E  15        102C  2865          BLEQ    ONE_LESS                ; Br if no, send it next time
     5D A8   96      102E  2866          INCB    CSB_B_NUM_SLOTS(R8)     ; Update slot count
                     1031  2867
                     1031  2868          ASSUME  SLT_B_DSTID EQ 0
89   0140 C5    90   1031  2869          MOVB    UCB$B_LT_REMID(R5), (R9)+ ; Send the remote slot index
                     1036  2870
                     1036  2871          ASSUME  SLT_B_SRCID EQ SLT_B_DSTID+1
                     1036  2872          ASSUME  SLT_B_COUNT EQ SLT_B_SRCID+1
          89   B4    1036  2873          CLRW    (R9)+                   ; Zero SRCID for a stop slot
                     1038  2874                                          ;   and no data
                     1038  2875          ASSUME  SLT_B_CRED EQ SLT_B_COUNT+1
89   D0 8F    90     1038  2876          MOVB    #SLT_C_STP_SLOT@4,(R9)+ ; Load stop slot format & no credits
   5A   04    C2     103C  2877          SUBL    #SLT_T_DATA,R10         ; Adjust bytes left for other slots
                     103F  2878                                          ;;&&;;** Skip call if HOST BIND
        0694   30     103F  2879          BSBW    LT$KILLUCB              ; Wipe out the UCB now.
          08   11     1042  2880          BRB     ONE_LESS                ; Get next slot
```

```
                     1044  2882          .SBTTL  PROCESS OUTPUT SLOT DATA
                     1044  2883 ;++
                     1044  2884 ; PROCESS OUTPUT SLOT DATA
                     1044  2885 ;
                     1044  2886 ; Functional description:
                     1044  2887 ;
                     1044  2888 ; Now output any waiting data from each UCB on this circuit and any
                     1044  2889 ; data returned from the class driver is transfered to the appropriate
                     1044  2890 ; output slot in the transmit buffer.
                     1044  2891 ;
                     1044  2892 ; Inputs:
                     1044  2893 ;       R0-R4 available
                     1044  2894 ;       R5  = UCB address
                     1044  2895 ;       R6,R7 =  scratch
                     1044  2896 ;       R8  = CSB address
                     1044  2897 ;       R9  = current transmit buffer slot address
                     1044  2898 ;       R10 = transmit buffer bytes left
                     1044  2899 ;       R11-FP = scratch
                     1044  2900 ;
                     1044  2901 ; Outputs:
                     1044  2902 ;       none.
                     1044  2903 ;
                     1044  2904 ; We continue processing in the send reply message section.
                     1044  2905 ;
                     1044  2906 ;--
                     1044  2907 SLOT_LOOP_OUT:
5D   2D A8   9A      1044  2908          MOVZBL  CSB_B_REFC(R8),FP       ; Set output UCB loop count
57   6C A8   DE      1048  2909          MOVAL   CSB_L_UCBLST(R8),R7     ; Set base address of UCB vector
                     104C  2910 ONE_LESS:                               ; One less unit left
        5D   D7      104C  2911          DECL    FP                      ; Any UCBs left ?
        03   18      104E  2912          BGEQ    10$                     ; Br if yes
      01EB   31      1050  2913          BRW     LOOP_EXIT               ; Else, exit loop
                     1053  2914 10$:
    55   87   D0      1053  2915          MOVL    (R7)+,R5                ; Get next UCB address
        FB   13      1056  2916          BEQL    10$                     ; ignore holes in non-dense vector
                     1058  2917          ;
                     1058  2918          ;  Dispatch on our terminal UCB state
                     1058  2919          ;
                     1058  2920          $DISPATCH   UCB$B_LT_STATE(R5),TYPE=B,-
                     1058  2921          <-      ; state             ; action
                     1058  2922          <UCB$C_LT_STATE_STOP    SEND_STOP_SLOT>,- ; STOP state, send stop slot
                     1058  2923          <UCB$C_LT_STATE_KILL    ONE_LESS>,-     ; KILL state, ignore this
                     1058  2924          >
                     1062  2925                                          ; RUN and START, just fall through
                     1062  2926
  0142 C5   95       1062  2927          TSTB    UCB$B_LT_DATAW(R5)      ; Is there any data waiting?
        E4   13      1066  2928          BEQL    ONE_LESS                ; Br if no, skip this UCB
                     1068  2929 ;
                     1068  2930 ;  Check for ABORT message first
                     1068  2931 ;
        02   E5      1068  2932          BBCC    #UCB$V_LT_ABORT,-       ; Br if no abort requested
1A 0142 C5           106A  2933                  UCB$B_LT_DATAW(R5),30$ ;   and clear flag
                     106E  2934          ;
                     106E  2935          ;  Send an ATTention slot
                     106E  2936          ;
                     106E  2937          ASSUME  SLT_S_ATT_SLOT&1 EQ 1   ; Assume we need rounding
    06   5A   D1      106E  2938          CMPL    R10, #SLT_S_ATT_SLOT+1  ; Enough room for ATTENTION slot ?
```

```
                     30   15  1071  2939          BLEQ    50$                             ; Br if no, skip it (use br helper)
                 5D  A8   96  1073  2940          INCB    CSB_B_NUM_SLOTS(R8)             ; Update slot count
                          1076  2941
                          1076  2942          ASSUME  UCB$B_LT_LOCID EQ UCB$B_LT_REMID+1
                          1076  2943          ASSUME  SLT_B_DSTID EQ 0
                          1076  2944          ASSUME  SLT_B_SRCID EQ SLT_B_DSTID+1
                          1076  2945          ASSUME  SLT_B_COUNT EQ SLT_B_SRCID+1
                          1076  2946          ASSUME  SLT_B_TYPE EQ SLT_B_COUNT+1
                          1076  2947          ASSUME  SLT_B_ATT_CONTROL EQ SLT_B_TYPE+1
                          1076  2948
             89   0140 C5   B0  1076  2949          MOVW    UCB$B_LT_REMID(R5), (R9)+       ; Send the remote slot id
                  89   01   90  107B  2950          MOVB    #SLT_S_ATT_LEN,(R9)+            ; Set ATTENTION slot data size
                89   B0  8F   90  107E  2951          MOVB    #SLT_C_ATT_SLOT&4,(R9)+         ; Load ATTn type & no credits
                          1082  2952
                          1082  2953          ASSUME  SLT_S_ATT_SLOT&1 EQ 1           ; Assume we need rounding
                  89   20   B0  1082  2954          MOVW    #SLT_ATT_M_ABORT,(R9)+          ; Send abort flag & zero byte
                  5A   06   C2  1085  2955          SUBL    #SLT_S_ATT_SLOT+1,R10           ; Account from room taken
                          1088  2956 ;
                          1088  2957 ;  Check for flow control message
                          1088  2958 ;
                          1088  2959 30$:
                       01   E5  1088  2960          BBCC    #UCB$V_LT_FLOW,-                ; Br if no flow requested
                 54 0142 C5        108A  2961                   UCB$B_LT_DATAW(R5),90$ ;     and clear flag
                          108E  2962          ;
                          108E  2963          ;  Send DATA_B slot
                          108E  2964          ;
                  0146 C5   97  108E  2965          DECB    UCB$B_LT_TCRED(R5)              ; Use up a credit
                       0C   18  1092  2966          BGEQ    40$                             ; Br if okay, continue
                  0146 C5   96  1094  2967          INCB    UCB$B_LT_TCRED(R5)              ; Else, restore credits
                          1098  2968          SETBIT  #UCB$V_LT_FLOW,-                ; Remember to send flow control
                          1098  2969                   UCB$B_LT_DATAW(R5)             ;  when we have good credit
                       42   11  109E  2970          BRB     90$                             ; And continue
                          10A0  2971
                          10A0  2972 40$:
                          10A0  2973          ASSUME  SLT_S_DB_SLOT&1 EQ 1            ; Assume we need rounding
                    0A    5A   D1  10A0  2974          CMPL    R10,-                           ; Enough room for DATA_B slot ?
                          10A3  2975                   #<SLT_S_DB_SLOT+1>             ;     rounded up
                       3D   15  10A3  2976 50$:     BLEQ    90$                             ; Br if no, skip it (branch helper)
                  5D   A8   96  10A5  2977          INCB    CSB_B_NUM_SLOTS(R8)             ; Update slot count
                          10A8  2978
                          10A8  2979          ASSUME  UCB$B_LT_LOCID EQ UCB$B_LT_REMID+1
                          10A8  2980          ASSUME  SLT_B_DSTID EQ 0
                          10A8  2981          ASSUME  SLT_B_SRCID EQ SLT_B_DSTID+1
                          10A8  2982          ASSUME  SLT_B_COUNT EQ SLT_B_SRCID+1
                          10A8  2983          ASSUME  SLT_B_TYPE EQ SLT_B_COUNT+1
                          10A8  2984          ASSUME  SLT_B_DB_CONTROL EQ SLT_B_TYPE+1
                          10A8  2985          ASSUME  SLT_B_DB_OFOF EQ SLT_B_DB_CONTROL+1
                          10A8  2986          ASSUME  SLT_B_DB_OFON EQ SLT_B_DB_OFOF+1
                          10A8  2987          ASSUME  SLT_B_DB_IFOF EQ SLT_B_DB_OFON+1
                          10A8  2988          ASSUME  SLT_B_DB_IFON EQ SLT_B_DB_IFOF+1
                          10A8  2989
                89   0140 C5   B0  10A8  2990          MOVW    UCB$B_LT_REMID(R5), (R9)+       ; Send the remote slot id
                     89   05   90  10AD  2991          MOVB    #SLT_S_DB_LEN,(R9)+             ; Set DATA_B data size
                  89   A0  8F   90  10B0  2992          MOVB    #SLT_C_DB_SLOT&4,(R9)+          ; Load DATA_B type & no credits
;JAY0001                           10B4    .1          ;
;JAY0001                           10B4    .2          ;  If the terminal is set to PASSALL, then we will disable all
;JAY0001                           10B4    .3          ;  flow control. Else, we will use XON/XOFF flow control if we
```

```
;JAY0001                                10B4     .4          ;  are set /HOSTSYNC or we will use NULL/BELL flow control if
;JAY0001                                10B4     .5          ;  we are set /NOHOSTSYNC.
;JAY0001                                10B4     .6          ;
;JAY0001                     89  0A  90 10B4     .7          MOVB    #SLT_DB_M_IFDIS!SLT_DB_M_OFDIS,(R9)+   ; Assume no flow control
;JAY0001               11131113  8F  D0 10B7     .8          MOVL    #<17@245!^I19@16>!<17@85!^I19@0>,(R9)+   ; Assume default values
                                89     10BD
;JAY0001           1A 44 A5    00  E0 10BE     .9          BBS     #TT$V_PASSALL,UCB$L_DEVDEPEND(R5),70$ ; Br if PASSALL
;JAY0001              FB A9    05  90 10C3    .10          MOVB    #SLT_DB_M_IFENA!SLT_DB_M_OFENA,-5(R9)  ; Else, use flow control
;JAY0001           04 44 A5    04  E0 10C7    .11          BBS     #TT$V_HOSTSYNC,UCB$L_DEVDEPEND(R5),60$ ; Assumed ^S and ^Q
;JAY0001              FE A9    07  B0 10CC    .12          MOVW    #<7@8>,-2(R9)                           ; Else, we were wrong
;JAY0001                                10D0    .13
;JAY0001                                10D0    .14 60$:    ;  Check /TTSYNC
;JAY0001                                10D0    .15
;JAY0001           08 44 A5    05  E0 10D0    .16          BBS     #TT$V_TTSYNC,UCB$L_DEVDEPEND(R5),70$ ; Br if ^S and ^Q
;JAY0001                                10D5    .17                                                   ;  assumption was correct
-10                  FB A9    04  8A 10D5   3003          BICB    #SLT_DB_M_OFENA,-5(R9)  ; Else, clear output flow enable
                     FB A9    08  88 10D9   3004          BISB    #SLT_DB_M_OFDIS,-5(R9)  ; and set output flow disable
                                      10DD   3005
                                      10DD   3006 70$:
                                      10DD   3007          ASSUME  SLT_S_DB_SLOT&1 EQ 1    ; Assume we need rounding
                                89  94 10DD   3008          CLRB    (R9)+                   ; Round up
                                   5A  0A  C2 10DF   3009          SUBL    #SLT_S_DB_SLOT+1,R10    ; Adjust bytes left (rounded up)
                                      10E2   3010
                                      10E2   3011 ;
                                      10E2   3012 ;  Check if there is any output data waiting
                                      10E2   3013 ;
                                      10E2   3014 90$:
                                      10E2   3015
```

```
                     10E2  3017 ;++
                     10E2  3018 ;
                     10E2  3019 ;  Any credits and enough room in output buffer for this slot ?
                     10E2  3020 ;
                     10E2  3021 ;--
                     10E2  3022 CREDIT_CHECK:                            ; UNUSED LABEL (breaks up LSB)
                     10E2  3023 ;
                     10E2  3024 ;  Flush all characters to transmit buffer
                     10E2  3025 ;
                     10E2  3026 ;       R8 = CSB address
                     10E2  3027 ;       R5 = UCB address
                     10E2  3028 ;
 56  0149 C5   9A   10E2  3029         MOVZBL  UCB$B_LT_SLOTSZ(R5),R6  ; Get maximum slot size
     56   5A   D1   10E7  3030         CMPL    R10,R6                  ; Is space left greater?
          03   14   10EA  3031         BGTR    10$                     ; Br if yes, then we can use it
     56   5A   D0   10EC  3032         MOVL    R10,R6                  ; Else, use lesser size
     56   04   C2   10EF  3033 10$:     SUBL    #SLT_T_DATA, R6         ; Subtract slot header size
          08   14   10F2  3034         BGTR    30$                     ; Br if enough room
               10F4  3035 20$:         SETBIT  #FLAG_V_RRF,-           ; Else, indicate response requested
               10F4  3036                       CSB_B_FLAG(R8)         ;  since not all data fit in buffer
        FF50   31   10F9  3037         BRW     ONE_LESS                ; No room left in output slot
 5C     56   D0   10FC  3038 30$:     MOVL    R6,AP                   ; Get room left in output slot
 5B   04 A9   9E   10FF  3039         MOVAB   SLT_T_DATA(R9),R11      ; Sink of characters
                     1103  3040         ;
                     1103  3041         ;  Check if we have credits, or if we need to return credits
                     1103  3042         ;  before building the message.
                     1103  3043         ;
     0146 C5   95   1103  3044         TSTB    UCB$B_LT_TCRED(R5)      ; Any credits we can use to send msg?
          09   14   1107  3045         BGTR    40$                     ; Br if yes, okay to build data message
                     1109  3046         ASSUME  LAT$C_MAX_RCRED EQ 1
     0147 C5   95   1109  3047         TSTB    UCB$B_LT_XCRED(R5)      ; Remote have any credits?
          E5   12   110D  3048         BNEQ    20$                     ; Br if yes, just leave now
        00C4   31   110F  3049         BRW     USE_CREDIT              ; Else, send back credits
                     1112  3050 40$:     ;
                     1112  3051         ;  Any data from startio ?
                     1112  3052         ;
 52  014B C5   9A   1112  3053         MOVZBL  UCB$B_LT_CURC(R5),R2    ; Get characters in startio ?
          4A   13   1117  3054         BEQL    70$                     ; Br if none
 53  014C C5   9E   1119  3055         MOVAB   UCB$B_LT_CBUF(R5),R3    ; Set address of buffer
     52   5C   D1   111E  3056         CMPL    AP,R2                   ; Got enough room in xmit buffer?
          29   18   1121  3057         BGEQ    60$                     ; Br if yes
                     1123  3058 ;
                     1123  3059 ;       R2 = COUNT
                     1123  3060 ;       R3 -> DATA
                     1123  3061 ;
014B C5   5C   82   1123  3062         SUBB2   AP, UCB$B_LT_CURC(R5)   ; Save residual count
          55   DD   1128  3063         PUSHL   R5                      ; Save the UCB address
 6B   63   5C   28   112A  3064         MOVC3   AP,(R3),(R11)           ; Copy the string from UCB to output bfr
               55 8ED0  112E  3065         POPL    R5                      ; Restore the UCB address
     5B   53   D0   1131  3066         MOVL    R3,R11                  ; Set address of next available byte
 52  014C C5   9E   1134  3067         MOVAB   UCB$B_LT_CBUF(R5), R2   ; Copy residual chars to beginning
 53  014B C5   9A   1139  3068         MOVZBL  UCB$B_LT_CURC(R5), R3   ;  of UCB buffer area
          55   DD   113E  3069         PUSHL   R5                      ; Save the UCB address
 62   61   53   28   1140  3070         MOVC3   R3,(R1),(R2)            ; Move the string up in the UCB
               55 8ED0  1144  3071         POPL    R5                      ; Restore the UCB address
          5C   D4   1147  3072         CLRL    AP                      ; No more room left
        008A   31   1149  3073         BRW     USE_CREDIT              ; Use a credit and send them
```

```
                        1140  3074
        014B C5   94    1140  3075 60$:    CLRB    UCB$B_LT_CURC(R5)          ; No data now
        5C   52   C2    1150  3076         SUBL2   R2,AP                      ; Adjust room left in slot
             55   DD    1153  3077         PUSHL   R5                         ; Save the UCB address
6B      63   52   28    1155  3078         MOVC3   R2,(R3),(R11)              ; Copy the string
             55 8ED0    1159  3079         POPL    R5                         ; Restore the UCB address
        5B   53   D0    115C  3080         MOVL    R3,R11                     ; Set address of next available byte
             5C   D5    115F  3081         TSTL    AP                         ; Any room left in slot ?
             73   15    1161  3082         BLEQ    USE_CREDIT                 ; No, just transmit data
                        1163  3083 ;
                        1163  3084 ;  Check if there is any data from last time
                        1163  3085 ;
52      0120 C5   32    1163  3086 70$:    CVTWL   UCB$W_TT_OUTLEN(R5),R2     ; Any data left from last time ?
             2C   15    1168  3087         BLEQ    GET_MORE                   ; Br if no, try for new data
53      011C C5   D0    116A  3088         MOVL    UCB$L_TT_OUTADR(R5),R3     ; Get address of data
        52   5C   D1    116F  3089         CMPL    AP,R2                      ; Enough room in buffer ?
             03   18    1172  3090         BGEQ    80$                        ; Br if yes
        52   5C   D0    1174  3091         MOVL    AP, R2                     ; Use all the space we have left
0120    C5   52   A2    1177  3092 80$:    SUBW2   R2,UCB$W_TT_OUTLEN(R5)     ; Adjust descriptor
011C    C5   52   C0    117C  3093         ADDL2   R2,UCB$L_TT_OUTADR(R5)     ;  ...
        5C   52   C2    1181  3094         SUBL    R2, AP                     ; Adjust count of chars left
             55   DD    1184  3095         PUSHL   R5                         ; Save the UCB address
6B      63   52   28    1186  3096         MOVC3   R2,(R3),(R11)              ; Copy the string
             55 8ED0    118A  3097         POPL    R5                         ; Restore the UCB address
        5B   53   D0    118D  3098         MOVL    R3,R11                     ; Set address of next available byte
             5C   D5    1190  3099         TSTL    AP                         ; Any room left now?
             02   14    1192  3100         BGTR    GET_MORE                   ; Br if yes, get some more data
             40   11    1194  3101         BRB     USE_CREDIT                 ; Else, try next slot
                        1196  3102
                        1196  3103 ;
                        1196  3104 ;  Loop draining class driver until no data is left or buffer is full
                        1196  3105 ;
                        1196  3106 GET_MORE:
        010C D5   16    1196  3107         JSB     @UCB$L_TT_GETNXT(R5)       ; Get characters from class driver
                        119A  3108 ;
                        119A  3109 ; R1-R4          modified
                        119A  3110 ; R3 = ??        and cc = 0                 ! No more data
                        119A  3111 ; R3 = char      and cc = +                 ! One character
                        119A  3112 ; R3 = ??        and cc = -                 ! Burst data (TT_OUTADR, TT_OUTLEN)
                        119A  3113 ; R5 - ucb address
                        119A  3114 ;
             40   13    119A  3115         BEQL    USE_CREDIT1                ; No characters
             09   19    119C  3116         BLSS    GOT_A_LOT                  ; Got more than one character
                        119E  3117
        8B   53   90    119E  3118         MOVB    R3,(R11)+                  ; Copy character to output
             5C   D7    11A1  3119         DECL    AP                         ; Adjust room in slot
             F1   14    11A3  3120         BGTR    GET_MORE                   ; Go get more
             2F   11    11A5  3121         BRB     USE_CREDIT                 ; No room left in slot
                        11A7  3122 GOT_A_LOT:
53      011C C5   D0    11A7  3123         MOVL    UCB$L_TT_OUTADR(R5),R3     ; Get output buffer address
52      0120 C5   3C    11AC  3124         MOVZWL  UCB$W_TT_OUTLEN(R5),R2     ;  and charcter count
        52   5C   D1    11B1  3125         CMPL    AP,R2                      ; Got enough room in output buffer?
             03   18    11B4  3126         BGEQ    10$                        ; Br if yes
        52   5C   D0    11B6  3127         MOVL    AP,R2                      ; Output maximum possible
        5C   52   C2    11B9  3128 10$:    SUBL2   R2,AP                      ; Adjust room left in slot
011C    C5   52   C0    11BC  3129         ADDL    R2,UCB$L_TT_OUTADR(R5)     ; Update output address
0120    C5   52   A2    11C1  3130         SUBW2   R2,UCB$W_TT_OUTLEN(R5)     ;  and charcter count
```

```
                55    DD   11C6  3131          PUSHL   R5                      ; Save the UCB address
        6B  63  52    28   11C8  3132          MOVC3   R2,(R3),(R11)           ; Copy the string
                55  8ED0   11CC  3133          POPL    R5                      ; restore the UCB address
            5B  53    D0   11CF  3134          MOVL    R3,R11                  ; set address of next available byte
                5C    D5   11D2  3135          TSTL    AP                      ; Any room left in slot ?
                C0    14   11D4  3136          BGTR    GET_MORE                ; Br if yes, look for more
```

```
                     11D6  3138          .SBTTL  CONSUME CREDIT WHEN FILLING IN SLOT FIELD
                     11D6  3139 ;
                     11D6  3140 ;  If any data was added to output slot, use credit locally, pass credit
                     11D6  3141 ;  to other end if needed, and set circuit mode to unbalanced.
                     11D6  3142 ;
                     11D6  3143          .ENABL  LSB
                     11D6  3144 USE_CREDIT:
   64 A5   02   A8   11D6  3145          BISW    #UCB$M_INT,UCB$L_STS(R5); Set interrupt expected, locks
                     11DA  3146                                          ;  out transmits from STARTIO
          0F   11    11DA  3147          BRB     10$                     ; Continue
                     11DC  3148
                     11DC  3149 USE_CREDIT1:
   64 A5   02   AA   11DC  3150          BICW    #UCB$M_INT,UCB$L_STS(R5); Clear interrupt expected
      02   E0   11E0  3151          BBS     #UCB$V_LT_OVFLOW,-          ; Don't clear data waiting flag
05 0143 C5          11E2  3152           UCB$B_LT_LATSTS(R5),10$;  if we're waiting for OVFLOW to clear
      01   8A   11E6  3153          BICB    #UCB$M_LT_DATA,-            ; No more output data waiting
    0142 C5          11E8  3154           UCB$B_LT_DATAW(R5)
                     11EB  3155 10$:
                     11EB  3156 ;;&&     SUBL3   AP,R6,AP                ; form character count in AP
                     11EB  3157 ;;&&     DECB    UCB$B_LT_TCRED(R5)      ; Use up a credit
                     11EB  3158 ;;&&     BISB    #FLAG_M_RRF,-           ; Set response requested flag
                     11EB  3159 ;;&&              CSB_B_FLAG(R8)         ;
      01   88   11EB  3160          BISB    #FLAG_M_RRF,-               ; Set response requested flag
    5C A8          11ED  3161           CSB_B_FLAG(R8)                 ;
5C  56   5C   C3   11EF  3162          SUBL3   AP,R6,AP                ; Calculate character count in AP
      0C   13   11F3  3163          BEQL    20$                     ; Br if no data, DON'T use credit
    0146 C5    97   11F5  3164          DECB    UCB$B_LT_TCRED(R5)      ; Else, use up a credit
                     11F9  3165          ASSUME  SLT_C_DA_SLOT EQ 0
    0147 C5    81   11F9  3166          ADDB3   UCB$B_LT_XCRED(R5),-    ; Calculate credits to send to remote
      50   01        11FD  3167                  #LAT$C_MAX_RCRED,R0     ;   system as DATA_A slot
           08   11    11FF  3168          BRB     40$                     ; Send credits and data
                     1201  3169          ASSUME  SLT_C_DA_SLOT EQ 0
    0147 C5    81   1201  3170 20$:     ADDB3   UCB$B_LT_XCRED(R5),-    ; See if we need to send
      50   01        1205  3171                  #LAT$C_MAX_RCRED,R0     ;   a credit to the remote
           32   13   1207  3172          BEQL    60$                     ; Br if no, send nothing
0148 C5    50   80   1209  3173 40$:     ADDB    R0,UCB$B_LT_RCRED(R5)   ; Else, add in credits to send
0147 C5    50   82   120E  3174          SUBB    R0,UCB$B_LT_XCRED(R5)   ; Credits have been extended
                     1213  3175 ;
                     1213  3176 ;       Setup slot header and advance pointer to next transmit slot header
                     1213  3177 ;
                     1213  3178          ASSUME  UCB$B_LT_LOCID EQ UCB$B_LT_REMID+1
                     1213  3179          ASSUME  SLT_B_SRCID EQ SLT_B_DSTID+1
                     1213  3180
    0140 C5    B0   1213  3181          MOVW    UCB$B_LT_REMID(R5),-    ; Set up slot connection ids
      69        1217  3182           SLT_B_DSTID(R9)
      5D   A8   96   1218  3183          INCB    CSB_B_NUM_SLOTS(R8)     ; Update slot count
   02 A9   5C   90   121B  3184          MOVB    AP,SLT_B_COUNT(R9)      ; Load character count
    0148 C5    90   121F  3185          MOVB    UCB$B_LT_RCRED(R5),-    ; Load slot type and credits
      03 A9        1223  3186           SLT_B_CRED(R9)
                     1225  3187          ASSUME  SLT_C_DA_SLOT EQ 0
    0148 C5    94   1225  3188          CLRB    UCB$B_LT_RCRED(R5)      ; Set RUN state, no more credits
      50   59   D0   1229  3189          MOVL    R9, R0                  ; Copy slot starting address
      59   5B   D0   122C  3190          MOVL    R11, R9                 ; Address of start of next slot
           59   D6   122F  3191          INCL    R9                      ; Make word address
      59   01   CA   1231  3192          BICL    #1, R9                  ; which is even
50    59   50   C3   1234  3193          SUBL3   R0, R9, R0              ; Compute bytes used by slot
      5A   50   C2   1238  3194          SUBL2   R0, R10                 ; Adjust bytes left for other slots
```

```
                FE0E   31   123B  3195 60$:    BRW     ONE_LESS                ; go do next UCB
                           123E  3196
                           123E  3197 LOOP_EXIT:
                           123E  3198 ;
                           123E  3199 ;       Compute and save the number of bytes in the transmit buffer.
                           123E  3200 ;
                           123E  3201 ;       R9 -> end of data in the transmit buffer
                           123E  3202 ;
                           123E  3203         ;
                           123E  3204         ;  There is a special case where the XMIT buffer can be taken,
                           123E  3205         ;  and that is when the circuit is halted. So if the circuit is
                           123E  3206         ;  in the HALT state, then we can assume we no longer have the
                           123E  3207         ;  XMIT buffer that we copied data into.
                           123E  3208         ;
                           123E  3209         $DISPATCH       CSB_B_STATE(R8),TYPE=B,- ; Dispatch on our CSB state
                           123E  3210         <-      ; state                 ; action
                           123E  3211                 <CSB_C_STATE_HALT       LT$REPLY_DONE>,- ; All done, exit
                           123E  3212         >
                           1245  3213
                   5D A8   95  1245  3214         TSTB    CSB_B_NUM_SLOTS(R8)     ; Are there any slots?
                      04   12  1248  3215         BNEQ    80$                     ; Br if yes, continue
                      01   8A  124A  3216         BICB    #FLAG_M_RRF,-           ; Else, clear RRF flag
                   5C A8          124C  3217                 CSB_B_FLAG(R8)          ;
                51    34 A8   D0  124E  3218 80$:    MOVL    CSB_Q_XBUFQ(R8), R1     ; Address of buffer
                1C A1    59   D0  1252  3219         MOVL    R9, CXB$L_T_ENDADR(R1)  ; Save end address of data
                           1256  3220         .DSABL  LSB
```

```
                     1256  3222          .SBTTL  SEND REPLY RESPONSE TO RECEIVED MESSAGE
                     1256  3223
                     1256  3224 LT$REPLY_MSG:
                     1256  3225 ;
                     1256  3226 ;  We come here to transmit a new frame in response to a received frame
                     1256  3227 ;
                     1256  3228 ;       R8   =  CSB address
                     1256  3229 ;
59   34 A8   D0      1256  3230          MOVL    CSB_Q_XBUFQ(R8),R9      ; Transmit buffer address
                     125A  3231
                     125A  3232 LT$REPLY_ALT:
                     125A  3233          .IF DEFINED LT_XMT_CHECK
                     125A  3234 ;
                     125A  3235 ;       Debug code
                     125A  3236 ;
                     125A  3237 ;       check the format of the slots and make sure that they are
                     125A  3238 ;       well formatted.
                     125A  3239 ;
                     125A  3240 ;       R8 = CSB address
                     125A  3241 ;       R9 = transmit buffer address
                     125A  3242 ;       R0-R3 clobbered
                     125A  3243 ;
                     125A  3244          MOVAQ   CSB_Q_XBUFQ(R8), R0             ; Address of transmit head
                     125A  3245          CMPL    R9, R0                          ; Is there a transmit buffer
                     125A  3246          BNEQ    5$                              ; Nope, its the queue head
                     125A  3247          DEBUG                                   ; Can't be the queue head
                     125A  3248
                     125A  3249 5$:      MOVQ    CSB_T_CIRCHDR(R8),-             ; Move the circ header
                     125A  3250                  XMT_T_DATA(R9)                  ; to look at real msg
                     125A  3251          CMPB    XMT_B_FLAG(R9),-                ; Only analyse running msgs
                     125A  3252                  #MTYP_C_RUN@FLAG_V_MTYPE        ;  with slots
                     125A  3253          BNEQ    50$                             ; Not a running message
                     125A  3254          MOVL    CXB$L_T_ENDADR(R9), R2
                     125A  3255          MOVAB   XMT_T_MDATA(R9), R0             ; First slot address
                     125A  3256          MOVZBL  XMT_B_NUM_SLOTS(R9), R1         ; number of valid slots
                     125A  3257          BNEQ    10$                             ; We have some
                     125A  3258          ;  No slot - message must only have a circuit header!
                     125A  3259          CMPL    R2, R0                          ; Only circuit header?
                     125A  3260          BEQL    50$                             ; Br if yes, okay
                     125A  3261          DEBUG                                   ; Not correct
                     125A  3262          BRB     50$
                     125A  3263
                     125A  3264 10$:     MOVZBL  SLT_B_COUNT(R0), R3             ; Look to next slot
                     125A  3265          BGEQ    30$                             ; Must be greater or zero
                     125A  3266          DEBUG                                   ; Maxc not valid
                     125A  3267 30$:     ADDL    #SLT_T_DATA, R3                 ; Slot header length
                     125A  3268          ADDL    R3, R0                          ;
                     125A  3269          INCL    R0                              ; Round to word boundary
                     125A  3270          BICL    #1, R0                          ;
                     125A  3271          CMPL    R0, R2                          ; Still inside buffer?
                     125A  3272          BLEQU   40$                             ; Yep
                     125A  3273          DEBUG                                   ; Slot beyond buffer
                     125A  3274 40$:     SOBGTR  R1, 10$                         ; Check all slots
                     125A  3275 50$:
                     125A  3276          .ENDC   ;; DEFINED LT_XMT_CHECK
                     125A  3277
54   58   D0         125A  3278          MOVL    R8, R4                          ; CSB address to r4
```

```
                                ;
                                ;  Retransmit last message
                                ;
                                ;       R8 = CSB address
                                ;
54   58   D0  1265  3292          MOVL    R8, R4                  ; CSB address to R4
     0869 30  1268  3293          BSBW    LT$REXMIT               ; Transmit all waiting buffers
       F3 11  126B  3294          BRB     LT$REPLY_DONE           ; Finish up
                126D  3295
```

```
                                126D  3297          .SBTTL  LT$CREATECSB  - Create a CSB
                                126D  3298 ;++
                                126D  3299 ; LT$CREATECSB - Create a CSB
                                126D  3300 ;
                                126D  3301 ;       Allocate a CSB for use and initialize it partially.
                                126D  3302 ;
                                126D  3303 ; Inputs:
                                126D  3304 ;       R4 -> next entry in CSB table beyond the one that is free
                                126D  3305 ;       R1 = length of UCB list at end of CSB
                                126D  3306 ;
                                126D  3307 ; Outputs:
                                126D  3308 ;       R8 = new CSB address
                                126D  3309 ;       R0 = success or failure
                                126D  3310 ;
                                126D  3311 ;       CSB table entry set to its address
                                126D  3312 ;
                                126D  3313 ;       R1-R2 are destroyed.
                                126D  3314 ;--
                                126D  3315
                                126D  3316 LT$CREATECSB:
00000070 8F    C0               126D  3317          ADDL    #CSB_C_FIXLENGTH+4,R1   ; Add in size of fixed block +
         51                     1273
                                1274  3318                                          ;  one "no access" UCB slot
         53    DD               1274  3319          PUSHL   R3                      ; Save R3
00000000'GF    16               1276  3320          JSB     G^EXE$ALONONPAGED       ; Get block
         53  8ED0               127C  3321          POPL    R3                      ; Restore R3
       4D 50   E9               127F  3322          BLBC    R0, 10$                 ; Got block
         3E    BB               1282  3323          PUSHR   #^M<R1,R2,R3,R4,R5>     ; Save registers
   00  6E    00    2C           1284  3324          MOVC5   #0,(SP),#0,R1,(R2)      ; Zero block
         62    51               1288
         3E    BA               128A  3325          POPR    #^M<R1,R2,R3,R4,R5>     ; Restore registers
   08 A2    51    B0           128C  3326          MOVW    R1,CSB_W_SIZE(R2)       ; Save size of CSB in CSB
   0A A2    33    90           1290  3327          MOVB    #DYN$C_CDB, -           ; Store an unlikely type in the
                               1294  3328                  CSB_B_TYPE(R2)          ; the block
50    EDA0  CF    DE           1294  3329          MOVAL   GHB_L_CSBTABLE, R0      ; Create the slave index from the
50    54    50    C3           1299  3330          SUBL3   R0, R4, R0              ; address in the CSB table
      50    04    C6           129D  3331          DIVL2   #4, R0                  ; from 1 to n
   60 A2    50    90           12A0  3332          MOVB    R0, CSB_B_LOCIDN(R2)    ; and save it in the CSB
   68 A2    50    90           12A4  3333          MOVB    R0, CSB_B_INX(R2)       ; Save it here also
    EE0B CF40    90           12A8  3334          MOVB    GHB_B_LOC_CCHK[R0],-    ; Obtain the check field for this
         61 A2                 12AD
                               12AF  3335                  CSB_B_LOCIDS(R2)        ;     CSB slot
                               12AF  3336 ;
                               12AF  3337 ;  Initialize queue headers
                               12AF  3338 ;
50    34 A2    9E               12AF  3339          MOVAB   CSB_Q_XBUFQ(R2), R0     ; Initialize the queue headers for the
   60    50    D0               12B3  3340          MOVL    R0, (R0)                ; transmit buffers
04 A0    50    D0               12B6  3341          MOVL    R0, 4(R0)               ;
50    3C A2    9E               12BA  3342          MOVAB   CSB_Q_XWAITQ(R2), R0    ;
   60    50    D0               12BE  3343          MOVL    R0, (R0)                ;
04 A0    50    D0               12C1  3344          MOVL    R0, 4(R0)               ;
   58    52    D0               12C5  3345          MOVL    R2, R8                  ; Return address of CSB here
   74    52    D0               12C8  3346          MOVL    R2, -(R4)               ; Store address of CSB in table
50    00'8F    9A               12CB  3347          MOVZBL  #SS$_NORMAL, R0
                  05            12CF  3348 10$:     RSB
```

```
                                        .SBTTL   LT$NEW_CIRCUIT - Get Concentrator State Block
                                ;++
                                ; LT$NEW_CIRCUIT - Get concentrator state block and initialize it
                                ;
                                ; Functional description:
                                ;
                                ; Whenever a frame with a zero svci is seen and the circuit state is
                                ; starting, we attempt to allocate a CSB and start a circuit.  First
                                ; we check for a CSB with the same source address.  If any are found
                                ; and the mvci matches and the state is starting, then we retransmit
                                ; our starting message for that circuit.  If the other things are not
                                ; true, then the circuit is clobbered to that source since the mvci
                                ; does not match or the circuit is already running and the concentrator
                                ; is trying to restart it perhaps because it rebooted.
                                ;
                                ; Inputs:
                                ;       R3 =  CXB address
                                ;       R6 =  receive buffer address
                                ;       R10 = IDB address
                                ;
                                ; Outputs:
                                ;       R6 =  receive buffer address
                                ;       R8 =  CSB address
                                ;       R0,R1,R2,R3,R4,R7 and R9 are modified
                                ;--
                                LT$NEW_CIRCUIT:
54   ED64 CF    9E              MOVAB   GHB_L_CSBTABLE,R4       ; Find free entry in CSB_TABLE
     52   20    9A              MOVZBL  #LAT$C_MAX_CSBS,R2      ; Length of table  in longwords
     58   84    D0              MOVL    (R4)+,R8                ; Get CSB address
          2A    13              BEQL    30$                     ; Ignore unallocated ones for pass
                                ;
                                ;       If this is the same source address, then we are looking for a duplicate
                                ;       starting message.  It will have the same local index and a state of
                                ;       starting.  If this is not the case, then just clobber the circuit.
                                ;
     28 A3      D1              CMPL    CXB$Q_STATION(R3),-     ; NI source address match?
     2E A8                              CSB_Z_DST(R8)           ;
          23    12              BNEQ    30$                     ; Br if not, look for more
     2C A3      B1              CMPW    CXB$Q_STATION+4(R3),-   ; Rest match too?
     32 A8                              CSB_Z_DST+4(R8)         ;
          1C    12              BNEQ    30$                     ; Br if not, look for more
     02 A6      B1              CMPW    RCV_W_DSTID(R6),-       ; Is this a duplicate starting msg
     60 A8                              CSB_W_LOCID(R8)         ;   with the same local vc indx/seq?
          0A    12              BNEQ    20$                     ; Br if not, kill circuit
     08 A8      91              CMPB    CSB_B_STATE(R8),-       ; Is the state still starting?
          01                            #CSB_C_STATE_START      ;
          04    12              BNEQ    20$                     ; Br if not, kill circuit
50        02    D0              MOVL    #2, R0                  ; Else, tell higher code to retrans
                05              RSB                             ;   our starting message
                                ;
          042B  30              BSBW    LT$CIRCDEAD             ; Get rid of UCBs
2E A8       01  CE              MNEGL   #1, CSB_Z_DST(R8)       ; Form illegal address
32 A8       01  AE              MNEGW   #1, CSB_Z_DST+4(R8)     ;  to stop next compare
                                ;
     CE 52      F5              SOBGTR  R2,10$                  ; Try whole table
                                ;
```

```
                                130A  3407          ;  Search failed, try to find a free CSB
                                130A  3408          ;
                          02    E0  130A  3409          BBS     #GHB_STS_V_SHUT,-       ; Br if we are shutting down,
                       0F EE89  CF  130C  3410                    GHB_B_STATUS,50$      ;   don't let connection succeed
                    54 ED24     CF  DE  1310  3411          MOVAL   GHB_C_CSBTABLE, R4      ; Look again for a free CSB
                       52       20  9A  1315  3412          MOVZBL  #LAT$C_MAX_CSBS, R2     ; Set max CSBs
                                84  D5  1318  3413  40$:    TSTL    (R4)+                   ; Is this slot free?
                                06  13  131A  3414          BEQL    60$                     ; Br if yes, great
                             F9 52  F5  131C  3415          SOBGTR  R2, 40$                 ; Loop til done
                                131F  3416          ;
;JAY0002                         131F    .1          ;  Return resource error
-1                              131F  3418          ;
                                50  D4  131F  3419  50$:    CLRL    R0                      ; Return failure indicating resource
                                05  1321  3420          RSB                             ; failure
                                1322  3421
                                1322  3422  60$:    ;
                                1322  3423          ;  Check the protocol version of the incoming message and reject all but
                                1322  3424          ;  our own.
                                1322  3425          ;
                          50    08 A6  9E  1322  3426          MOVAB   RCV_T_MDATA(R6), R0     ; Address of first slot
                          05    02 A0  91  1326  3427          CMPB    STRT_B_PVER(R0), -      ; Check version number
                                132A  3428                    #LAT$C_CUR_VER
                                F3  12  132A  3429          BNEQ    50$                     ; Must be equal
                                132C  3430          ;
                                132C  3431          ;  Check the circuit timer value (in multiples of 10 ms).
                                132C  3432          ;  We will accept anything in the range 10 to 1270 ms; legal
                                132C  3433          ;  values are 1 to 127 in this byte field.
                                132C  3434          ;
                             06 A0  95  132C  3435          TSTB    STRT_B_CIR_TIMR(R0)     ; Is this field positive and non-zero?
                                EE  15  132F  3436          BLEQ    50$                     ; Br if not, illegal value
                                1331  3437  ;
                                1331  3438  ;       ECO is not checked
                                1331  3439  ;
                                1331  3440  ;       R4 -> CSB table address -4
                                1331  3441  ;
                                1331  3442          ;
                                1331  3443          ;  Compute size needed for UCB list at end of CSB,
;JAY0002                         1331    .1          ;  size is based on MIN [GHB_B_MAX_SLOTS, STRT_B_MAXSLOTS(R0)]
;JAY0002                         1331    .2          ;
;JAY0002                55    EDD7  CF  9A  1331    .3          MOVZBL  W^GHB_B_MAX_SLOTS,R5    ; Assume ours is MIN value
-3                        55    04 A0  91  1336  3447          CMPB    STRT_B_MAXSLOTS(R0),R5  ; Is rcvd slot count larger?
                                04  1E  133A  3448          BGEQU   70$                     ; Br if yes, use ours
                          55    04 A0  90  133C  3449          MOVB    STRT_B_MAXSLOTS(R0),R5  ; Else, use smaller of two
                       51 55    02  78  1340  3450  70$:    ASHL    #2,R5,R1                ; Multiply by 4, and save in R1
                                1344  3451
                             FF26  30  1344  3452          BSBW    LT$CREATECSB            ; Create a CSB
                          D5 50    E9  1347  3453          BLBC    R0, 50$                 ; Br if resource error
                          69 A8    55  90  134A  3454          MOVB    R5,CSB_B_MAX_SLOTS(R8)  ; Save the maximum allowed slot count
                                134E  3455          ;
                                134E  3456          ;  Save the server's name
                                134E  3457          ;
                                134E  3458          ASSUME  STRT_B_SYS_LEN EQ STRT_T_NODE+GHB$K_NAMELEN
                          50    08 A6  9E  134E  3459          MOVAB   RCV_T_MDATA(R6),R0      ; Get address of first slot
                          51    0C A0  9A  1352  3460          MOVZBL  STRT_B_NODE_LEN(R0),R1  ; Get length of dest node name
                          52    0D A0  9E  1356  3461          MOVAB   STRT_T_NODE(R0),R2      ; Get address of dest node name
                             52    51  C0  135A  3462          ADDL    R1,R2                   ; Skip dest node name, point to src name
                             51    62  9A  135D  3463          MOVZBL  (R2),R1                 ; Get length of src (server) name
```

```
                10    51    D1  1360  3464          CMPL    R1,#GHB$K_NAMELEN       ; Is the name length okay?
                      06    15  1363  3465          BLEQ    75$                     ; Br if yes, continue
                51    10    9A  1365  3466          MOVZBL  #GHB$K_NAMELEN,R1       ; Else, use maximum allowed
                62    51    90  1368  3467          MOVB    R1,(R2)                 ; Make size of name valid
                      51    D6  136B  3468 75$:       INCL    R1                      ; Account for length byte
                      09    BB  136D  3469          PUSHR   #^M<R0,R3>              ; Save registers
                62    51    28  136F  3470          MOVC3   R1,(R2),CSB_B_SERVER(R8) ; Copy the server name
                  1C  A8        1372
                      09    BA  1374  3471          POPR    #^M<R0,R3>              ; Restore registers
                                1376  3472          ;
                                1376  3473          ;  Make the circuit timeout value twice the keep_alive value
                                1376  3474          ;  received from the terminal server.
                                1376  3475          ;
              0E10  8F    B0  1376  3476          MOVW    #3600, CSB_W_TIMRESET(R8); Assume a default of one hour
                66  A8        137A
            51    07    A0  9A  137C  3477          MOVZBL  STRT_B_KPA_TIMR(R0), R1 ; Make timer reset value
                      11    13  1380  3478          BEQL    80$                     ; Not a good value
                51    05    C4  1382  3479          MULL2   #5,R1                   ; Look for five times the period,
                      51    D6  1385  3480          INCL    R1                      ;  and one more
                55    51    3C  1387  3481          MOVZWL  R1,R5                   ; Check for overflow
                55    51    D1  138A  3482          CMPL    R1,R5                   ; Did we overflow?
                      04    12  138D  3483          BNEQ    80$                     ; Br if yes, use default value
            66  A8    51    B0  138F  3484          MOVW    R1, CSB_W_TIMRESET(R8)  ; Set the value in the CSB
                                1393  3485 80$:      ;
                                1393  3486          ;  Compute maximum buffer size to use
                                1393  3487          ;
              05D4  8F    B0  1393  3488          MOVW    #XMT_C_MAXDATA,-        ; Assume ours is maximum allowed
                6A  A8        1397  3489            CSB_W_MAX_MSGSIZ(R8)    ;    message size
                      08    A2  1399  3490          SUBW    #XMT_C_HDRLEN,-         ; Account for header is remotes size
                      60        139B  3491            STRT_B_MSGSIZ(R0)
                      0A    15  139C  3492          BLEQ    90$                     ; Br if not large enough
                      60    B1  139E  3493          CMPW    STRT_W_MSGSIZ(R0),-     ; Is remotes buffer larger?
                        6A  A8        13A0  3494            CSB_W_MAX_MSGSIZ(R8)
                      04    1E  13A2  3495          BGEQU   90$                     ; Br if yes, use ours
                      60    B0  13A4  3496          MOVW    STRT_W_MSGSIZ(R0),-     ; Else, use smaller of two
                        6A  A8        13A6  3497            CSB_W_MAX_MSGSIZ(R8)
                                13A8  3498 90$:      ;
                                13A8  3499          ;  Note that locidn and xmit queue headers were setup by CREATECSB
                                13A8  3500          ;
            5E  A8    04  A6  B0  13A8  3501          MOVW    RCV_W_SRCID(R6),CSB_W_REMID(R8) ; In case of retransmit
                      5D  A8  94  13AD  3502          CLRB    CSB_B_NUM_SLOTS(R8)     ; Mode default is balanced
                      52  A8  B4  13B0  3503          CLRW    CSB_W_XMTTMO(R8)        ; Not timed state
                      66  A8  B0  13B3  3504          MOVW    CSB_W_TIMRESET(R8),-    ; Set a nonzero timeout value
                      50  A8        13B6  3505            CSB_W_TIMEOUT(R8)       ; even if the normal value is zero.
                      2D  A8  94  13B8  3506          CLRB    CSB_B_REFC(R8)          ; No UCBs here
                      14  B0  13BB  3507          MOVW    #LAT$C_PROGRESS,-       ; Set for allowing progress
                      54  A8        13BD  3508            CSB_W_PROGRESS(R8)      ; while we start up.
                                13BF  3509
                      28  A3  D0  13BF  3510          MOVL    CXB$Q_STATION(R3),-     ; Store remotes's NI address.
                      2E  A8        13C2  3511            CSB_Z_DST(R8)
                      2C  A3  B0  13C4  3512          MOVW    CXB$Q_STATION+4(R3),-   ;       ...
                      32  A8        13C7  3513            CSB_Z_DST+4(R8)
                          00  90  13C9  3514          MOVB    #CSB_C_STATE_HALT,-     ; Circuit is halted
                      08  A8        13CB  3515            CSB_B_STATE(R8)
                62  A8    01  AE  13CD  3516          MNEGW   #1,CSB_W_XSEQ(R8)       ; Initialize transmit sequence
                    FF00  8F  B0  13D1  3517          MOVW    #<-1@8!05>,CSB_W_RSEQ(R8); Initialize receive sequence
                      64  A8        13D5
```

```
                        13D7  3518          ;
                        13D7  3519          ;  Obtain all necessary transmit buffers for use
                        13D7  3520          ;
      58 A8    94       13D7  3521          CLRB    CSB_B_XMTBSY(R8)          ; Not busy transmitting
      59 A8    94       13DA  3522          CLRB    CSB_B_XMTCNT(R8)          ; No buffers waiting to transmit
54    34 A8    DE       13DD  3523          MOVAL   CSB_Q_XBUFQ(R8), R4       ; Address of the transmit queue head
   55    02    9A       13E1  3524          MOVZBL  #LAT$C_XMT_BUFFERS, R5    ; Set number of buffers
                        13E4  3525          ;
51    0624 8F    3C       13E4  3526 100$:    MOVZWL  #XMT_C_MAXLEN, R1         ; Size of the buffers
 00000000'GF    16       13E9  3527          JSB     G^EXE$ALONONPAGED         ; Allocate the buffer
      25 50    E9         13EF  3528          BLBC    R0, 110$                  ; No such buffer
   08 A2    51    B0       13F2  3529          MOVW    R1, CXB$W_SIZE(R2)        ; Store the size,
                        13F6  3530          ASSUME  CXB$B_CODE EQ CXB$B_TYPE+1
   0A A2    1B    B0       13F6  3531          MOVW    #DYN$C_CXB,-              ; the block type
                        13FA  3532                  CXB$B_TYPE(R2)            ; and indicate not multicast buffer
                        13FA  3533          ;
   0C A2    58    D0       13FA  3534          MOVL    R8,CXB$L_T_CSB(R2)        ; Save CSB address in CXB
      2E A8    7D         13FE  3535          MOVQ    CSB_Z_DST(R8),-           ; Initialize the destination address
      28 A2               1401  3536                  CXB$Q_STATION(R2)
      48 A2    7C         1403  3537          CLRQ    XMT_T_DATA(R2)            ; Zero circuit header
    0048 8F    B0         1406  3538          MOVW    #XMT_T_DATA,-             ; Store offset to start of data
      18 A2               140A  3539                  CXB$W_BOFF(R2)
   64    62    0E         140C  3540          INSQUE  (R2), (R4)                ; Queue the buffer
      D2 55    F5         140F  3541          SOBGTR  R5, 100$                  ; for each of the buffers
                        1412  3542          ;
50    00'8F    9A       1412  3543          MOVZBL  #SS$_NORMAL, R0
                  05       1416  3544          RSB
                        1417  3545          ;
                        1417  3546          ;
                        1417  3547          ;  Resource error return
                        1417  3548          ;
54    58    D0       1417  3549 110$:    MOVL    R8, R4                    ; Move the CSB address for the call
    0003    30       141A  3550          BSBW    LT$DEALOCSB               ; Clean up the CSB before we leave
      50    D4       141D  3551          CLRL    R0                        ; Return an error
                  05       141F  3552          RSB
                        1420  3553
```

```
                     1420  3555          .SBTTL  LT$DEALOCSB  - Deallocate a CSB
                     1420  3556 ;++
                     1420  3557 ; LT$DEALOCSB
                     1420  3558 ;
                     1420  3559 ; FUNCTIONAL DESCRIPTION:
                     1420  3560 ;
                     1420  3561 ;       Deallocate the transmit buffers.
                     1420  3562 ;
                     1420  3563 ; INPUTS:
                     1420  3564 ;       R4 = CSB address
                     1420  3565 ;
                     1420  3566 ; OUTPUTS:
                     1420  3567 ;       R0-R3 are destroyed
                     1420  3568 ;
                     1420  3569 ;--
                     1420  3570
                     1420  3571 LT$DEALOCSB:
      51    01   9A  1420  3572          MOVZBL  #LAT$C_XMT_BUFFERS-1,R1 ; Get number of transmits -1
50    44 A441   D0  1423  3573 10$:     MOVL    CSB_L_XCXB[R4][R1], R0  ; Is an FFI transmit in progress ?
         03   13  1428  3574          BEQL    20$                     ; no
      0C A0   D4  142A  3575          CLRL    CXB$L_T_CSB(R0)         ; Delete backpointer to CSB in CXB
      F3 51   F4  142D  3576 20$:     SOBGEQ  R1,10$                  ; Loop if more
50    34 B4   0F  1430  3577 25$:     REMQUE  @CSB_Q_XBUFQ(R4), R0    ; Now dump all the buffers
         05   1D  1434  3578          BVS     30$                     ; No more buffers
       07A0   30  1436  3579          BSBW    LT$DEALPOOL             ; Dump a buffer
         F5   11  1439  3580          BRB     25$                     ;
                     1438  3581
50    3C B4   0F  1438  3582 30$:     REMQUE  @CSB_Q_XWAITQ(R4), R0   ; Now dump all the waiters
         05   1D  143F  3583          BVS     40$                     ; No more buffers
       0795   30  1441  3584          BSBW    LT$DEALPOOL             ; Dump a buffer
         F5   11  1444  3585          BRB     30$                     ;
                     1446  3586
   53    68 A4   9A  1446  3587 40$:     MOVZBL  CSB_B_INX(R4), R3       ; Find our place in the table
         06   12  144A  3588          BNEQ    50$                     ; Great
                     144C  3589          DEBUG                           ; Not properly initialized CSB
                     1452  3590
   EC61 CF43   96  1452  3591 50$:     INCB    GHB_B_LOC_CCHK[R3]      ; Bump the slot check digit
50 EBD8 CF43   DE  1457  3592          MOVAL   GHB_L_CSBTABLE-4[R3],R0 ; Address in table
      54   60   D1  145D  3593          CMPL    (R0), R4                ; Check this address with the one
         06   13  1460  3594          BEQL    60$                     ; we have.
                     1462  3595          DEBUG                           ; Table is clobbered or CSB addr bad
                     1468  3596
         60   D4  1468  3597 60$:     CLRL    (R0)                    ; No CSB address here anymore
                     146A  3598          ;
                     146A  3599          ;  We will now save this CSB on the OLD CSB queue, but we will
                     146A  3600          ;  have to remove an old entry if we exceed our limit.
                     146A  3601          ;
      50   54   D0  146A  3602          MOVL    R4,R0                   ; Copy CSB address just in case
                     146D  3603                                          ;  we want to deallocate it
      0C A4   D5  146D  3604          TSTL    CSB_Z_LCB+LCB_L_MSG_XMT(R4) ; Any information in counters?
         08   12  1470  3605          BNEQ    70$                     ; Br if yes, save it
      10 A4   D5  1472  3606          TSTL    CSB_Z_LCB+LCB_L_MSG_RCV(R4) ; Any information in counters?
         03   12  1475  3607          BNEQ    70$                     ; Br if yes, save it
       008C   31  1477  3608          BRW     180$                    ; Else, nothing interesting here
                     147A  3609 70$:     ;
                     147A  3610          ;  We will now scan the CSB list for any old CSBs that match this
                     147A  3611          ;  one, that way we will only save 1 entry per server.
```

```
                     147A  3612                ;
51   ECC6 CF    9E   147A  3613                MOVAB   GHB$Q_OLD_CSBS+GHB_AREA,R1 ; Get listhead of old CSBs
     52   51    D0   147F  3614                MOVL    R1,R2                        ; ..twice
     52   62    D0   1482  3615  80$:          MOVL    (R2),R2                      ; Get next in queue
     51   52    D1   1485  3616                CMPL    R2,R1                        ; Are we back to start?
          68    13   1488  3617                BEQL    100$                         ; Br if yes, no more - insert new entry
        2E A4   D1   148A  3618                CMPL    CSB_Z_DST(R4),-              ; Does the Ethernet address match?
        2E A2        148D  3619                        CSB_Z_DST(R2)
          F1    12   148F  3620                BNEQ    80$                          ; Br if no, try next
        32 A4   B1   1491  3621                CMPW    CSB_Z_DST+4(R4),-            ; Full address match?
        32 A2        1494  3622                        CSB_Z_DST+4(R2)
          EA    12   1496  3623                BNEQ    80$                          ; Br if no, try next
   53    1C A4   9A   1498  3624                MOVZBL  CSB_B_SERVER(R4),R3          ; Get length of server name
 1C A2    53   91   149C  3625                CMPB    R3,CSB_B_SERVER(R2)          ; Lengths match?
          E0   12   14A0  3626                BNEQ    80$                          ; Br if no, try next
          07   BB   14A2  3627                PUSHR   #^M<R0,R1,R2>                ; Save registers
 1D A4    53   29   14A4  3628                CMPC3   R3,CSB_T_SERVER(R4),-        ; Does the server name match?
       1D A2        14A8  3629                        CSB_T_SERVER(R2)
          07   BA   14AA  3630                POPR    #^M<R0,R1,R2>
          D4   12   14AC  3631                BNEQ    80$                          ; Br if no, try next
                    14AE  3632                ;
                    14AE  3633                ;    Take sum of counters and store in CSB @ R2
                    14AE  3634                ;
                    14AE  3635  ADD_CTR       CSB_Z_LCB+LCB_L_MSG_XMT(R4),- ; XMITs
                    14AE  3636                        CSB_Z_LCB+LCB_L_MSG_XMT(R2)
                    14B9  3637  ADD_CTR       CSB_Z_LCB+LCB_L_MSG_RCV(R4),- ; RCVs
                    14B9  3638                        CSB_Z_LCB+LCB_L_MSG_RCV(R2)
                    14C4  3639  ADD_CTR       CSB_Z_LCB+LCB_L_MSG_REXMT(R4),- ; REXMTs
                    14C4  3640                        CSB_Z_LCB+LCB_L_MSG_REXMT(R2)
                    14CF  3641  ADD_CTR       CSB_Z_LCB+LCB_W_SEQ_ERR(R4),- ; SEQ_ERRs
                    14CF  3642                        CSB_Z_LCB+LCB_W_SEQ_ERR(R2),W
                    14DA  3643  ADD_CTR       CSB_Z_LCB+LCB_B_INV_MSG(R4),- ; INV_MSGs
                    14DA  3644                        CSB_Z_LCB+LCB_B_INV_MSG(R2),B
                    14E5  3645  ADD_CTR       CSB_Z_LCB+LCB_B_INV_SLOT(R4),- ; INV_SLOTs
                    14E5  3646                        CSB_Z_LCB+LCB_B_INV_SLOT(R2),B
          14   11   14F0  3647                BRB     180$                         ; And delete the CSB @ R4
                    14F2  3648  100$:         ;
                    14F2  3649                ;  Insert new CSB into list
                    14F2  3650                ;
EC51 DF    64   0E   14F2  3651                INSQUE  (R4), @GHB$Q_OLD_CSBS+ -; Save CSB at end of CSB queue
                    14F7  3652                          GHB_AREA+4           ;   since queue is time ordered
     EC9F CF    97   14F7  3653                DECB    GHB_B_OLD_CSBCNT             ; Use up one more slot on queue
          0C    18   14FB  3654                BGEQ    190$                         ; Br if still within CSB limits
     EC99 CF    96   14FD  3655                INCB    GHB_B_OLD_CSBCNT             ; Else, restore count
50   EC3F DF    0F   1501  3656                REMQUE  @GHB$Q_OLD_CSBS+ -           ; Remove oldest entry
                    1506  3657                          GHB_AREA,R0
        06D0    30   1506  3658  180$:         BSBW    LT$DEALPOOL                  ; Deallocate the CSB block
                05   1509  3659  190$:         RSB
```

```
                        150A  3661          .SBTTL  LT$CREATEUCB  - Create an LT UCB
                        150A  3662 ;++
                        150A  3663 ; LT$CREATEUCB - Create an LT UCB
                        150A  3664 ;
                        150A  3665 ;       Create a UCB by cloning it from the master UCB zero and
                        150A  3666 ;       initialize it as part of the io database.  We can't lock
                        150A  3667 ;       the io database for write access, because we aren't a
                        150A  3668 ;       process.  So we just do it and hope for the best.  The
                        150A  3669 ;       modifications are such that code scanning the database will
                        150A  3670 ;       not be fooled if they are straight forward.  It is possible
                        150A  3671 ;       to write code that will be confused when a new ucb appears,
                        150A  3672 ;       but we don't have any solution other than waiting for the
                        150A  3673 ;       the request to come here when the database is not locked.
                        150A  3674 ;       That would probably mean that many connection attempts would
                        150A  3675 ;       time out.
                        150A  3676 ;
                        150A  3677 ;                       NB !!
                        150A  3678 ;
                        150A  3679 ;       The I/O data base is not locked!!
                        150A  3680 ;
                        150A  3681 ; Inputs:
                        150A  3682 ;       none
                        150A  3683 ;
                        150A  3684 ; Outputs:
                        150A  3685 ;       R5 = new UCB address
                        150A  3686 ;       R0 = success or failure
                        150A  3687 ;
                        150A  3688 ;       R1-R4 clobbered
                        150A  3689 ;--
                        150A  3690
                        150A  3691 LT$CREATEUCB:
55   EC72 CF    D0      150A  3692          MOVL    GHB_L_UCB0, R5          ; Get the UCB to clone from
 00000000'GF    16      150F  3693          JSB     G^IOC$CLONE_UCB         ; Clone the UCB
       06 50    E9      1515  3694          BLBC    R0,90$                  ; Exit on error
    55    52    D0      1518  3695          MOVL    R2,R5                   ; Else, copy new UCB address
      5C A5    B4      151B  3696          CLRW    UCB$W_REFC(R5)          ; No refs yet!
               05      151E  3697 90$:     RSB
```

```
                                                D 5
                     151F  3699          .SBTTL  LT$DESTROYUCB  - Destroy an LT UCB
                     151F  3700 ;++
                     151F  3701 ; LT$DESTROYUCB - Destroy an LT UCB
                     151F  3702 ;
                     151F  3703 ;       Delete a ucb from the io data base that has been disconnected.
                     151F  3704 ;       This routine must be called at high IPL when the I/O database
                     151F  3705 ;       is not in use by anyone.  We are going to deallocate a block
                     151F  3706 ;       and if anyone has its address in a register, we are going
                     151F  3707 ;       to go crash with a bang.  This situation is not quite the
                     151F  3708 ;       same as the creation case where one can be much more cavalier.
                     151F  3709 ;
                     151F  3710 ;       Also note, that we must ensure that the LT UCB has been detached
                     151F  3711 ;       from any VT UCB, and that the REFC has gone to zero. There is
                     151F  3712 ;       a condition where the class driver will not detach a VT UCB
                     151F  3713 ;       even though the disconnect entry point is called. This occurs if
                     151F  3714 ;       loginout has not been run yet, but is about to be run. In this
                     151F  3715 ;       case, the class driver will not have detached the LT UCB from the
                     151F  3716 ;       VT UCB and we can check for that condition.
                     151F  3717 ;
                     151F  3718 ;               NB !!!
                     151F  3719 ;       The I/O database is NOT locked for write access, BUT the MUTEX
                     151F  3720 ;       is available!
                     151F  3721 ;
                     151F  3722 ; Inputs:
                     151F  3723 ;       R5 = UCB to destroy
                     151F  3724 ;
                     151F  3725 ; Outputs:
                     151F  3726 ;       R5 = UCB address
                     151F  3727 ;       R0-R1 are destroyed.
                     151F  3728 ;--
                     151F  3729
                     151F  3730 LT$DESTROYUCB:
00C0 C5    55   D1   151F  3731                CMPL    R5,UCB$L_TT_LOGUCB(R5)  ; Is this UCB ready for deletion?
           12   12   1524  3732                BNEQ    50$                     ; Br if no, requeue it
        5C A5   B5   1526  3733                TSTW    UCB$W_REFC(R5)          ; All refs gone?
           0D   12   1529  3734                BNEQ    50$                     ; Br if no, requeue it
                     152B  3735                SETBIT  #UCB$V_DELETEUCB, -     ; Else, allow UCB to be deleted
                     152B  3736                        UCB$L_STS(R5)
00000000'GF   16   1530  3737                JSB     G^IOC$DELETE_UCB        ; Delete the UCB
           0E   11   1536  3738                BRB     90$                     ; leave
                     1538  3739
           04   90   1538  3740 50$:          MOVB    #UCB$C_LT_STATE_KILL,-  ; Enter the KILL UCB state
        0144 C5         153A  3741                      UCB$B_LT_STATE(R5)
                     153D  3742                CLRBIT  #UCB$V_LT_DEAD,-        ; Clear the dead indicator
                     153D  3743                        UCB$B_LT_LATSTS(R5)
              014A   30   1543  3744                BSBW    LT$HANGUP_UCB           ; And hangup the UCB again!
                 05   1546  3745 90$:          RSB                             ; Return
```

```
                                                E 5
                     1547  3747          .SBTTL   LT$ALC_UCB - Allocate New LT UCB
                     1547  3748 ;++
                     1547  3749 ; LT$ALC_UCB - Allocate a new LT UCB
                     1547  3750 ;
                     1547  3751 ; Subroutine to select a UCB for a slot.
                     1547  3752 ;
                     1547  3753 ; Inputs:
                     1547  3754 ;       R7 =  receive buffer slot address
                     1547  3755 ;       R8 =  CSB address
                     1547  3756 ;
                     1547  3757 ; Outputs:
                     1547  3758 ;       R0 =  success or failure
                     1547  3759 ;       R5 =  UCB address
                     1547  3760 ;       R7 =  receive buffer slot address
                     1547  3761 ;       R8 =  CSB address
                     1547  3762 ;       R1,R2,R3,R4 modified
                     1547  3763 ;--
                     1547  3764
                     1547  3765 LT$ALC_UCB:
          50   D4    1547  3766          CLRL    R0                          ; Return failure if no UCB
                     1549  3767
                     1549  3768 ;;&&    *TEMP*  concentrator bug
                     1549  3769
    54    6C A8  DE  1549  3770          MOVAL   CSB_L_UCBLST(R8), R4        ; Search the list for this CSB
    53    54     D0  154D  3771          MOVL    R4, R3                      ; Save the address
    51    69 A8  9A  1550  3772          MOVZBL  CSB_B_MAX_SLOTS(R8), R1     ; Get maximum in list
    55    84     D0  1554  3773 1$:      MOVL    (R4)+, R5                   ; Look in the table
          0B     13  1557  3774          BEQL    2$                          ; A free slot, my word
        01 A7    91  1559  3775          CMPB    SLT_B_SRCID(R7),-           ; Is this the same terminal calling
      0140 C5        155C  3776                  UCB$B_LT_REMID(R5)          ;  again?
          03     12  155F  3777          BNEQ    2$                          ; No, then continue
         010C    31  1561  3778          BRW     95$                         ; Yes, then setup message again
        ED 51    F5  1564  3779 2$:      SOBGTR  R1, 1$                      ; look again
                     1567  3780
                     1567  3781 ;;&&    *TEMP* concentrator bug
                     1567  3782
                     1567  3783
    54    6C A8  DE  1567  3784          MOVAL   CSB_L_UCBLST(R8), R4        ; Search the list for this CSB
    53    54     D0  156B  3785          MOVL    R4, R3                      ; save the address
    51    69 A8  9A  156E  3786          MOVZBL  CSB_B_MAX_SLOTS(R8), R1     ; Get maximum in list
    55    84     D0  1572  3787 10$:     MOVL    (R4)+, R5                   ; Look in the table
          06     13  1575  3788          BEQL    20$                         ; A free slot, my word
        F8 51    F5  1577  3789          SOBGTR  R1, 10$                     ; look again
         00F2    31  157A  3790 15$:     BRW     90$                         ; None to be had here, so fail
                     157D  3791
        02 A7    91  157D  3792 20$:     CMPB    SLT_B_COUNT(R7),-           ; Is slot big enough?
                 05  1580  3793                  #<SLT_B_DST_NAMLEN-SLT_T_DATA>+2
                 0A  18  1581  3794          BGEQ    30$                         ; Br if yes, okay
                     1583  3795          INC_CTR CSB_Z_LCB+LCB_B_INV_SLOT(R8) ; Else, say invalid slot
                 ED  11  158B  3796          BRB     15$                         ; And exit
                     158D  3797                                              ;
                 54  DD  158D  3798 30$:     PUSHL   R4                          ; Save the address in table
          54     53  C2  158F  3799          SUBL2   R3, R4                      ; Compute the index for this CSB
          54     04  C6  1592  3800          DIVL2   #4, R4                      ;  as a small integer
                 54  DD  1595  3801          PUSHL   R4                          ; Save the index too
             FF70    30  1597  3802          BSBW    LT$CREATEUCB                ; Create a raw lt UCB
                 54 8ED0  159A  3803          POPL    R4                          ; Obtain the index
```

```
                53 8ED0   159D  3804          POPL    R3                      ;  and its address in CSB table
                          15A0  3805          ;                               ;
          D7 50    E9     15A0  3806          BLBC    R0, 15$                 ; Do we have one?, nope, too bad
       73   55    D0     15A3  3807          MOVL    R5, -(R3)               ; Store its address in CSB table
          F4F2    30     15A6  3808          BSBW    LT$UCB_INIT             ; Initialize the ucb
0141 C5    54    90     15A9  3809          MOVB    R4,UCB$B_LT_LOCID(R5)   ; Set local slot index
      01   A7    90     15AE  3810          MOVB    SLT_B_SRCID(R7),-       ; Load remote circuit id
      0140 C5           15B1  3811                  UCB$B_LT_REMID(R5)      ;
      04   00    EF     15B4  3812          EXTZV   #0,#4,SLT_B_CRED(R7),R0 ; Get credit total extended
50    03   A7           15B7
      50   01    81     15BA  3813          ADDB3   #1,R0,UCB$B_LT_TCRED(R5); Give ourself an extra credit for
      0146 C5           15BD
                        15C0  3814                                          ;   sending the start slot
      91   8F    90     15C0  3815          MOVB    #SLT_C_STR_SLOT@4!LAT$C_MAX_RCRED,- ; Set slot type and credits
      0148 C5           15C3  3816            UCB$B_LT_RCRED(R5)
           01   8E     15C6  3817          MNEGB   #LAT$C_MAX_RCRED,-      ; Set credits extended to remote end
      0147 C5           15C8  3818            UCB$B_LT_XCRED(R5)          ;   kept as negative value
      0082 C5    B4     15CB  3819          CLRW    UCB$W_ERRCNT(R5)        ; Reset the error counter that
                        15CF  3820                                          ; we bump as we shut down.
                A8      15CF  3821          BISW    #<UCB$M_INT!-           ; Expecting interrupts & online
                        15D0  3822                  UCB$M_ONLINE>,-         ;
64 A5    12             15D0  3823                  UCB$L_STS(R5)           ;
0134 C5    58    D0     15D3  3824          MOVL    R8,UCB$L_LT_CSB(R5)     ; Load CSB address
      2D   A8    96     15D8  3825          INCB    CSB_B_REFC(R8)          ; One more UCB here
                        15DB  3826          ;
                        15DB  3827          ;  Compute maximum allowed slot size
                        15DB  3828          ;
      FF   8F    90     15DB  3829          MOVB    #LAT$C_MAX_SLOTSIZ,-    ; Assume ours is the maximum
      0149 C5           15DE  3830            UCB$B_LT_SLOTSZ(R5)           ;  slot size
      06   A7    91     15E1  3831          CMPB    SLT_B_DT_SLTSIZ(R7),-   ; Is remotes size smaller?
      0149 C5           15E4  3832            UCB$B_LT_SLOTSZ(R5)
           06    1E     15E7  3833          BGEQU   40$                     ; Br if no, use ours
      06   A7    90     15E9  3834          MOVB    SLT_B_DT_SLTSIZ(R7),-   ; Else, take remotes
      0149 C5           15EC  3835            UCB$B_LT_SLOTSZ(R5)
           02    90     15EF  3836 40$:     MOVB    #UCB$C_LT_STATE_RUN,-   ; Set the initial state to RUN
      0144 C5           15F1  3837            UCB$B_LT_STATE(R5)
           05    90     15F4  3838          MOVB    #GHB_C_STRT_SLOTL,-     ; Build a start slot
      014B C5           15F6  3839            UCB$B_LT_CORC(R5)
           55    DD     15F9  3840          PUSHL   R5                      ; Save UCB address
           05    28     15FB  3841          MOVC3   #GHB_C_STRT_SLOTL,-     ; Make it look like output data
      EEB1 CF           15FD  3842            GHB_T_STRT_SLOT,-
      014C C5           1600  3843            UCB$B_LT_CBUF(R5)
           55 8ED0      1603  3844          POPL    R5                      ; Save UCB address
           01   88      1606  3845          BISB    #UCB$M_LT_DATA,-        ; Force data through output slot
      0142 C5           1608  3846            UCB$B_LT_DATAW(R5)
                        160B  3847          ;
                        160B  3848          ;  Check for the disable login flag
                        160B  3849          ;
   51   07   A7    9E     160B  3850          MOVAB   SLT_B_DST_NAMLEN(R7),R1 ; Skip to dest. slot name field
      50   81    9A     160F  3851          MOVZBL  (R1)+,R0                ; Get size of dest. name field
      51   50    C0     1612  3852          ADDL    R0,R1                   ; Skip the dest. name field
      50   81    9A     1615  3853          MOVZBL  (R1)+,R0                ; Get size of src name field
      51   50    C0     1618  3854          ADDL    R0,R1                   ; Skip the src name field
   50   04   A7    9E     161B  3855          MOVAB   SLT_T_DATA(R7),R0       ; Compute size of slot skipped
52    51   50    C3     161F  3856          SUBL3   R0,R1,R2                ;  ...
   50   02   A7    9A     1623  3857          MOVZBL  SLT_B_COUNT(R7),R0      ; Get size of entire slot
      50   52    C2     1627  3858          SUBL    R2,R0                   ; Compute bytes remaining
```

```
                30   15  162A  3859                BLEQ    70$                     ; Br if no more data, do login
           52   81   9A  162C  3860 60$:           MOVZBL  (R1)+,R2                ; Get parameter code
                2B   13  162F  3861                BEQL    70$                     ; Br if none, assume auto-login
                50   D7  1631  3862                DECL    R0                      ; One less byte left
                27   15  1633  3863                BLEQ    70$                     ; Br if no more data, do login
           53   81   9A  1635  3864                MOVZBL  (R1)+,R3                ; Get size of parameter
           50   53   C2  1638  3865                SUBL    R3,R0                   ; Get remaining byte count
                1F   15  163B  3866                BLEQ    70$                     ; Br if no more data, do login
           54   51   D0  163D  3867                MOVL    R1,R4                   ; Save current parameter address
           51   53   C0  1640  3868                ADDL    R3,R1                   ; Skip to next parameter
           01   52   D1  1643  3869                CMPL    R2,#SLT_C_START_FLAG    ; Is this the start slot flag byte?
                E4   12  1646  3870                BNEQ    60$                     ; Br if no, look for next
      0C 64     02   E1  1648  3871                BBC     #SLT_FLAG_V_BIND,(R4),65$ ; Br if not a bind request
                     164C  3872                CLRBIT  #TT2$V_HANGUP,-         ; Else, set terminal to NOHANGUP
                     164C  3873                        UCB$L_TT_DECHA1(R5)     ;   on logout
                     1652  3874                SETBIT  #UCB$V_LT_BOUND,-       ; Don't delete UCB after STOP slot
                     1652  3875                        UCB$B_LT_CATSTS(R5)     ;... only if we must KILL the UCB!
      0F 64     01   E0  1658  3876 65$:           BBS     #SLT_FLAG_V_LOGIN,(R4),80$ ; Br if login is disabled
                     165C  3877                ;
                     165C  3878                ;  Ask the class driver to log us in
                     165C  3879                ;
   51   0114 C5   D0  165C  3880 70$:           MOVL    UCB$L_TT_CLASS(R5), R1  ; Class dispatch vector
    00002000 8F   D0  1661  3881                MOVL    #1@13, R3               ; Framming error with zero byte
                53        1667
          14 B1   16  1668  3882                JSB     @CLASS_READERROR(R1)    ; Cause read error
   50    00'8F   9A  166B  3883 80$:           MOVZBL  #SS$_NORMAL, R0         ; Do we have a UCB for you.
                05  166F  3884 90$:           RSB
                    1670  3885
                    1670  3886 ;;&&           *TEMP* concentrator  bug
                    1670  3887
                    1670  3888 95$:           SETBIT  #GHB$V_REPCREATE, -     ; Set mask bit
                    1670  3889                        GHB$L_PROTOMASK+GHB_AREA ; Repeat create of slot
                    1676  3890                INC_CTR GHB$L_PROTOCOL+GHB_AREA ; Increment protocol error counter
          FF6C  31  1680  3891                BRW     40$
                    1683  3892 ;;&&           *TEMP*  concentrator bug
```

```
                        1683  3894          .SBTTL  LT$HANGUP_UCB  - Cause Terminal Hangup On UCB
                        1683  3895 ;++
                        1683  3896 ; LT$HANGUP_UCB - Hangup an LT UCB
                        1683  3897 ;
                        1683  3898 ; Cause all the hangup effects on a terminal device.  This should cause
                        1683  3899 ; process rundown and deallocation of all channels to the UCB.
                        1683  3900 ;
                        1683  3901 ; Inputs:
                        1683  3902 ;       R5 = UCB address
                        1683  3903 ;
                        1683  3904 ; Outputs:
                        1683  3905 ;       none.
                        1683  3906 ;
                        1683  3907 ;--
                        1683  3908          .ENABL  LSB
                        1683  3909 LT$HANGUP_UCB_NOW:
                        1683  3910          ;
                        1683  3911          ;  Dispatch on UCB (slot) state
                        1683  3912          ;
                        1683  3913          $DISPATCH  UCB$B_LT_STATE(R5),TYPE=B,-
                        1683  3914          <-      ; state         ; action
                        1683  3915          <UCB$C_LT_STATE_STOP    LT$KILLUCB>,-   ; Already disconnected,
                        1683  3916          >
                        1688  3917
                        1688  3918          ;  Else
                        1688  3919
    04    90            1688  3920          MOVB    #UCB$C_LT_STATE_KILL,-          ; Enter the KILL UCB state
0144  C5                168D  3921                   UCB$B_LT_STATE(R5)
                        1690  3922
                        1690  3923 LT$HANGUP_UCB:
          1F    BB      1690  3924          PUSHR   #^M<R0,R1,R2,R3,R4>             ; Save registers
  64  A5    02    AA      1692  3925          BICW    #UCB$M_INT,UCB$L_STS(R5)        ; No interrupt expected (allows
                        1696  3926                                                  ;   STARTIO to be called again
                        1696  3927                                                  ;   to flush data).
                        1696  3928          SETBIT  #UCB$V_LT_HANGUP,-              ; Indicate that we're shutting
                        1696  3929                   UCB$B_LT_LATSTS(R5)            ;   down
  50    0114  C5    D0      169C  3930          MOVL    UCB$L_TT_CLASS(R5),R0           ; Class driver vector address
        18  B0    16      16A1  3931          JSB     @CLASS_DISCONNECT(R0)           ; Call routine
          1F    BA      16A4  3932          POPR    #^M<R0,R1,R2,R3,R4>             ; Restore the regs
                05      16A6  3933 90$:     RSB
                        16A7  3934
                        16A7  3935          .DSABL  LSB
```

```
                                                I 5
                     16A7  3937          .SBTTL  LT$DISCONNECT  - Port Disconnect
                     16A7  3938 ;++
                     16A7  3939 ; LT$DISCONNECT - Disconnect the terminal Port driver
                     16A7  3940 ;
                     16A7  3941 ;       Port driver entry to disconnect a terminal.  called when
                     16A7  3942 ;       Last reference is gone.  here we free the slot in the buffer
                     16A7  3943 ;       And bump the sequence number.  also we set the UCB offline.
                     16A7  3944 ;
                     16A7  3945 ; Inputs:
                     16A7  3946 ;       R5 = UCB address
                     16A7  3947 ;       R0 -> lbc = delete UCB
                     16A7  3948 ;       R0 -> lbs = do not delete UCB
                     16A7  3949 ;
                     16A7  3950 ; Outputs:
                     16A7  3951 ;       R5 = UCB address
                     16A7  3952 ;       All other registers are preserved.
                     16A7  3953 ;
                     16A7  3954 ; Implicit inputs:
                     16A7  3955 ;       UCB$B_LT_STATE = KILL -> kill UCB immediately.
                     16A7  3956 ;                        STOP -> send stop slot, then kill UCB.
                     16A7  3957 ;                        ???? -> all others, set to STOP state.
                     16A7  3958 ;
                     16A7  3959 ;--
                     16A7  3960
                     16A7  3961 LT$DISCONNECT:
                     16A7  3962          ;
                     16A7  3963          ; Do not touch the UCB if it is the template UCB0
                     16A7  3964          ;
EAD4 CF    55   D1   16A7  3965          CMPL    R5,GHB_L_UCB0           ; Is this the template ucb ?
           27   13   16AC  3966          BEQL    90$                     ; Br if yes, don't touch it!
                     16AE  3967          ;
                     16AE  3968          ;   Dispatch on our UCB state
                     16AE  3969          ;
                     16AE  3970          $DISPATCH  UCB$B_LT_STATE(R5),TYPE=B,-
                     16AE  3971          <-      ; state             ; action
                     16AE  3972          <UCB$C_LT_STATE_KILL    50$>,-  ; KILLed, just kill the UCB now
                     16AE  3973          <UCB$C_LT_STATE_STOP    90$>,-  ; STOPed, we've been here before
                     16AE  3974          >
                     16B8  3975                                          ; ALL other states,
                     16B8  3976          ;
                     16B8  3977          ;  If the circuit is gone, or there is no CSB for the UCB
                     16B8  3978          ;  then just delete the UCB immediately
                     16B8  3979          ;
0134 C5    D5        16B8  3980          TSTL    UCB$L_LT_CSB(R5)        ; Do we have a circuit?
           14   13   16BC  3981          BEQL    50$                     ; Br if none
                     16BE  3982          ;
                     16BE  3983          ;   Conditionally skip deallocation of UCB, on "hangup".
                     16BE  3984          ;
  14 50    E8        16BE  3985          BLBS    R0,90$                  ; Br if not to "hangup" UCB
;JAY0001    10   AA   16C1     .1        BICW    #UCB$M_ONLINE,-         ; No longer ONLINE
;JAY0001    64   A5   16C3     .2                UCB$L_STS(R5)           ; ..so no one can use again
           03   90   16C5  3986          MOVB    #UCB$C_LT_STATE_STOP,-  ; Else, set STOP state,
0144 C5             16C7  3987                   UCB$B_LT_STATE(R5)     ;  to send STOP slot
                     16CA  3988          ;
                     16CA  3989          ;  Make sure that no data is given to CLASS driver, after
                     16CA  3990          ;  we've been here!
                     16CA  3991          ;
```

```
                16CA  3992          SETBIT  #UCB$V_LT_HANGUP,-       ; Indicate that we're shutting
                16CA  3993                    UCB$B_LT_LATSTS(R5)    ;   down
          03    16D0  3994          BRB     90$                      ; Delete UCB when STOP is sent!
                16D2  3995
        0001    16D2  3996 50$:     BSBW    LT$KILLUCB               ; Get rid of the UCB now
                16D5  3997
                16D5  3998 90$:     RSB                              ; Done
                16D6  3999
```

```
                        16D6  4001          .SBTTL  LT$KILLUCB  - Delete a dried up UCB
                        16D6  4002 ;++
                        16D6  4003 ; LT$KILLUCB - Delete an LT UCB
                        16D6  4004 ;
                        16D6  4005 ; The UCB is all dried up one way or another.  So we must delete it if
                        16D6  4006 ; possible, else put it on the deadlink list to be deleted on the next
                        16D6  4007 ; timer tick.
                        16D6  4008 ;
                        16D6  4009 ; Inputs:
                        16D6  4010 ;       R5 = UCB address
                        16D6  4011 ;
                        16D6  4012 ; Outputs:
                        16D6  4013 ;       R5 = UCB address
                        16D6  4014 ;       All other registers preserved.
                        16D6  4015 ;
                        16D6  4016 ;--
                        16D6  4017
                        16D6  4018 LT$KILLUCB:
              1F    BB  16D6  4019          PUSHR   #^M<R0,R1,R2,R3,R4>     ; Save some volatile regs
                    AA  16D8  4020          BICW    #<UCB$M_INT! -          ; No interrupt expected
                        16D9  4021                      UCB$M_ONLINE>,-     ; Not online, but free now
        64 A5       12  16D9  4022                      UCB$L_STS(R5)       ; For use again
                F57A    30  16DC  4023          BSBW    LT$FLUSH_DATA       ; Flush all output data
                0027    30  16DF  4024          BSBW    LT$SIDELINEUCB      ; Set the UCB aside from the circuit
                  01    E3  16E2  4025          BBCS    #UCB$V_LT_DEAD,-    ; If we already linked UCB to deadlist
        12 0143 C5          16E4  4026                  UCB$B_LT_LATSTS(R5),20$ ;  then we have made an error
                            16E8  4027          SETBIT  #GHB$V_REPDISC,-    ; KILLUCB called twice
                            16E8  4028                  GHB$L_PROTOMASK+GHB_AREA;
                            16EE  4029          INC_CTR GHB$L_PROTOCOL+GHB_AREA ; Increment protocol error counter
                      0C    11  16F8  4030          BRB     90$             ;
                            16FA  4031 20$:     ;
                            16FA  4032          ;  We must destroy the UCB (not called from $DASSGN, so queue the
                            16FA  4033          ;  UCB to the deadlist so it can be deallocated by the timer service.
                            16FA  4034          ;  This is because the CLASS driver still uses the UCB even if we
                            16FA  4035          ;  deallocate it!
                            16FA  4036          ;
           EA86 CF      D0  16FA  4037          MOVL    GHB_L_DEADLINK,-    ; Link UCB into chain of dead UCBs
                0138 C5        16FE  4038                  UCB$L_LT_DEADLINK(R5) ; for disposal at timeout
    EA7E CF     55      D0  1701  4039          MOVL    R5, GHB_L_DEADLINK  ;
                  1F    BA  1706  4040 90$:     POPR    #^M<R0,R1,R2,R3,R4>     ; Restore the regs
                      05    1708  4041          RSB
```

```
                        1709  4043          .SBTTL  LT$SIDELINEUCB  - Set a UCB aside
                        1709  4044 ;++
                        1709  4045 ; LT$SIDELINEUCB
                        1709  4046 ;
                        1709  4047 ; Functional description:
                        1709  4048 ;
                        1709  4049 ; Set a UCB aside to dry up on its own.  Disconnect it from a running
                        1709  4050 ; circuit.
                        1709  4051 ;
                        1709  4052 ; Inputs:
                        1709  4053 ;       R5 = UCB address
                        1709  4054 ;
                        1709  4055 ; Outputs:
                        1709  4056 ;       none
                        1709  4057 ;       All registers preserved.
                        1709  4058 ;
                        1709  4059 ;--
                        1709  4060
                        1709  4061 LT$SIDELINEUCB:
                  18    BB  1709  4062          PUSHR   #^M<R3,R4>                      ; Save registers
54          0134 C5     D0  1708  4063          MOVL    UCB$L_LT_CSB(R5), R4            ; Get the CSB address
                  15    13  1710  4064          BEQL    10$                             ; None, this is already done
53          0141 C5     9A  1712  4065          MOVZBL  UCB$B_LT_LOCID(R5), R3          ; Use the local slot index in UCB
            68 A443     D4  1717  4066          CLRL    CSB_L_UCBLST-4(R4)[R3]          ; to adjust state of UCB list in CSB
            0134 C5     D4  171B  4067          CLRL    UCB$L_LT_CSB(R5)                ; Remember we have been here
              2D A4     97  171F  4068          DECB    CSB_B_REFC(R4)                  ; One less UCB on this circuit
                03      12  1722  4069          BNEQ    10$                             ; We have more UCBs here
              0044      30  1724  4070          BSBW    LT$STOPCIRC                     ; Then don't wait for return traffic
                18      BA  1727  4071 10$:     POPR    #^M<R3,R4>                      ; Restore registers
                05      1729  4072              RSB                                     ; Return to caller
```

```
                        172A  4074          .SBTTL  LT$CIRCDEAD  - Declare circuit dead
                        172A  4075  ;++
                        172A  4076  ; LT$CIRCDEAD  - Declare circuit dead.
                        172A  4077  ;
                        172A  4078  ; Functional description:
                        172A  4079  ;
                        172A  4080  ;       Log off all the processes on each UCB and waiting for them
                        172A  4081  ;       to free their UCBs.  This will cause the CSB to go free too.
                        172A  4082  ;       If this CSB has no ucb's attached, then the CSB is just cleaned up.
                        172A  4083  ;
                        172A  4084  ; Inputs:
                        172A  4085  ;       R8 = CSB address
                        172A  4086  ;
                        172A  4087  ; Outputs:
                        172A  4088  ;       All UCB's hungup.
                        172A  4089  ;
                        172A  4090  ;--
                        172A  4091
                        172A  4092  LT$CIRCDEAD:
    03FF 8F     BB      172A  4093          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Save registers
      2D A8     95      172E  4094          TSTB    CSB_B_REFC(R8)                  ; Do we have any UCBs?
        18      13      1731  4095          BEQL    100$                            ; Br if no, so deallocate CSB
57      69 A8   9A      1733  4096          MOVZBL  CSB_B_MAX_SLOTS(R8), R7         ; Else, number of UCBs to scan
56      6C A8   DE      1737  4097          MOVAL   CSB_L_UCBLST(R8), R6            ; List of the UCBs
55        86    D0      173B  4098  20$:    MOVL    (R6)+, R5                       ; Get next UCB address
          03    18      173E  4099          BGEQ    60$                             ; Not system address
        FF40    30      1740  4100          BSBW    LT$HANGUP_UCB_NOW               ; Hangup the process on this UCB
        F5 57   F5      1743  4101  60$:    SOBGTR  R7, 20$                         ; For all the units here
                        1746  4102
    03FF 8F     BA      1746  4103  90$:    POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Restore registers
                05      174A  4104          RSB                                     ; Return to caller
                        174B  4105          ;
                        174B  4106          ; Refc has gone to zero, send a stop message
                        174B  4107          ;
54      58      D0      174B  4108  100$:   MOVL    R8, R4                          ; Setup CSB address
  .       1B    10      174E  4109          BSBB    LT$STOPCIRC                     ; Send a stop message on circ
          F4    11      1750  4110          BRB     90$                             ; And exit
                        1752  4111
```

```
                                1752  4113          .SBTTL  LT$STOPCIRC  - Transmit stop message
                                1752  4114 ;++
                                1752  4115 ; LT$STOPCIRC - Transmit stop message
                                1752  4116 ;
                                1752  4117 ; Functional description:
                                1752  4118 ;
                                1752  4119 ; Format and transmit stop message
                                1752  4120 ;
                                1752  4121 ; Inputs
                                1752  4122 ;       R4 = CSB address
                                1752  4123 ;       CSB_L_STOPREASON        addr of stop reason block
                                1752  4124 ;
                                1752  4125 ; Outputs:
                                1752  4126 ;       none
                                1752  4127 ;
                                1752  4128 ;--
                                1752  4129
                                1752  4130 DEFREASN:
20 65 64 6F 4E 00'              1752  4131          .ASCIC  /Node terminated circuit./
6E 69 6D 72 65 74               1758
63 20 64 65 74 61               175E
74 69 75 63 72 69               1764
                     2E         176A
                     18         1752
                                176B  4132
                                176B  4133 LT$STOPCIRC:
      03FF  8F       BB         176B  4134          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Save registers
      58    54       D0         176F  4135          MOVL    R4, R8                          ; Set CSB addr in r8
        2D  A8       95         1772  4136          TSTB    CSB_B_REFC(R8)                  ; We are not supposed to send
            06       13         1775  4137          BEQL    10$                             ; stop messages with
                                1777  4138          DEBUG                                   ; outstanding ucbs on circ
          0B  A8     91         177D  4139 10$:     CMPB    CSB_B_STATE(R8),-               ; Have we been here before?
                     00         1780  4140                  #CSB_C_STATE_HALT
                     03         12    1781  4141          BNEQ    15$                     ;
                   008A         31    1783  4142          BRW     90$                     ; Br if no
66 A8                01         B0    1786  4143 15$:     MOVW    #1,CSB_W_TIMRESET(R8)   ; Else, exit
50 A8                01         B0    178A  4144          MOVW    #1,CSB_W_TIMEOUT(R8)    ; Set timers for quick timeout
                     00         90    178E  4145          MOVB    #CSB_C_STATE_HALT,-     ; and set state to halted
          0B  A8               1790  4146                  CSB_B_STATE(R8)
          60  A8     B4         1792  4147          CLRW    CSB_W_LOCID(R8)                 ; Zero the local ID
50        34  A8     7E         1795  4148          MOVAQ   CSB_Q_XBUFQ(R8), R0             ; Transmit buffer queue
57        00  B0     0F         1799  4149          REMQUE  @(R0), R7                       ; Do we have a free buffer?
                     0F         1C    179D  4150          BVC     30$                     ; Yep, got it
57        3C  B8     0F         179F  4151          REMQUE  @CSB_Q_XWAITQ(R8), R7           ; Get a waiting bfr
                     06         1C    17A3  4152          BVC     20$                     ; Ok, so use it
                                17A5  4153          DEBUG                                   ; All buffers gone?????
          59  A8     97         17AB  4154 20$:     DECB    CSB_B_XMTCNT(R8)                ; One less buffer on waitq
       60     67     0E         17AE  4155 30$:     INSQUE  (R7), (R0)                      ; Make it a free buffer
       59     57     D0         17B1  4156          MOVL    R7, R9                          ; Save bfr header adr
57        48  A7     9E         17B4  4157          MOVAB   XMT_T_DATA(R7), R7              ; Stop message data
                     08         90    17B8  4158          MOVB    #MTYP_C_HALT@FLAG_V_MTYPE,-     ; Send a HALT message
          5C  A8               17BA  4159                  CSB_B_FLAG(R8)
          5D  A8     94         17BC  4160          CLRB    CSB_B_NUM_SLOTS(R8)             ; No slot data
53        50  A9     9E         17BF  4161          MOVAB   XMT_T_MDATA(R9), R3             ; Set default end of data
                     83         94    17C3  4162          CLRB    (R3)+                   ; Set default reason code
50        8A  AF     9E         17C5  4163          MOVAB   DEFREASN, R0                    ; Store a default text string
       51     60     9A         17C9  4164          MOVZBL  (R0), R1                        ; the count
```

```
                 51   D6  17CC  4165          INCL    R1                              ; including the count
          63     60   51   28  17CE  4166          MOVC3   R1, (R0), (R3)                  ; move count and string
            56     4C A8   D0  17D2  4167          MOVL    CSB_L_STOPREASON(R8), R6        ; Do we have a reason
                    29   13  17D6  4168          BEQL    40$                             ; No reason
              50    02 A6   9A  17D8  4169          MOVZBL  2(R6), R0                       ; Set the reason bit
                            17DC  4170          SETBIT  R0,GHB$L_PROTOMASK+GHB_AREA     ; and count the error
                            17E2  4171          INC_CTR GHB$L_PROTOCOL+GHB_AREA         ; as a halt
        50 A9    03 A6   90  17EC  4172          MOVB    3(R6), -                        ; Store the reason code
                            17F1  4173                  STOP_B_RCODE+XMT_T_MDATA(R9)    ;
                 50   66   3C  17F1  4174          MOVZWL  (R6), R0                        ; Offset to the string
                 50   56   C0  17F4  4175          ADDL    R6, R0                          ; Set the address
                 51   60   9A  17F7  4176          MOVZBL  (R0), R1                        ; Get the count
                      51   D6  17FA  4177          INCL    R1                              ; and set the count
                 60     51   28  17FC  4178          MOVC3   R1, (R0), -                     ; copy the counted string
                   51 A9          17FF
                                1801  4179                  STOP_B_RLEN+XMT_T_MDATA(R9)     ; Leave r3 pointing beyond
                                1801  4180 40$:
              1C A9    53   D0  1801  4181          MOVL    R3, CXB$L_T_ENDADR(R9)          ; and store in buffer
                 54    58   D0  1805  4182          MOVL    R8,R4                           ; copy CSB address
                     0264   30  1808  4183          BSBW    LT$XMIT                         ; and transmit the msg
                 03FF 8F    BA  180B  4184 50$:     POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Restore registers
                            05  180F  4185          RSB                                     ; Return to caller
                                1810  4186
                                1810  4187 ; Stop message sent once
                                1810  4188 ; Just wipe CSB now
                                1810  4189
                 54    58   D0  1810  4190 90$:     MOVL    R8, R4                          ; CSB address to R4
                     FC0A   30  1813  4191          BSBW    LT$DEALOCSB                     ; Deallocate CSB to pool
                       F3   11  1816  4192          BRB     50$                             ; And exit
                                1818  4193
```

```
                                1818  4195          .SBTTL  LT$SET_TIMER  - Start Timer
                                1818  4196 ;++
                                1818  4197 ; LT$SET_TIMER - Start the timer
                                1818  4198 ;
                                1818  4199 ; Functional description:
                                1818  4200 ;
                                1818  4201 ;       Start timer ticking to scan the CSB's for dead circuits
                                1818  4202 ;
                                1818  4203 ; Inputs:
                                1818  4204 ;       R0-R2 available
                                1818  4205 ;
                                1818  4206 ; Implicit Inputs:
                                1818  4207 ;
                                1818  4208 ;       IPL = LAT$C_IPL (8)
                                1818  4209 ;
                                1818  4210 ; Outputs:
                                1818  4211 ;       R0-R2 are destroyed.
                                1818  4212 ;
                                1818  4213 ;--
                                1818  4214
                                1818  4215 LT$SET_TIMER:
50   E8C0 CF    9E              1818  4216          MOVAB   GHB_G_TQE, R0           ; Address of timer queue element
                01  E2          181D  4217          BBSS    #GHB_STS_V_ACTIVE,-     ; Br if timer already active
  1F E976 CF                    181F  4218                    GHB_B_STATUS,10$      ;   and mark as active
000D'CF     04  88              1823  4219          BISB    #DPT$M_NOUNLOAD,DPT$TAB+DPT$B_FLAGS ; Make driver unloadable
            30  B0              1828  4220          MOVW    #TQE$C_LENGTH,-         ; Setup timer queue entry block
      08    A0                  182A  4221                  TQE$W_SIZE(R0)          ; For its first use
            0F  90              182C  4222          MOVB    #DYN$C_TQE,-            ; With both the size and the type
      0A    A0                  182E  4223                  TQE$B_TYPE(R0)          ; Code.
            38  BB              1830  4224          PUSHR   #^M<R3,R4,R5>           ; Save the fork context
      55    50  D0              1832  4225          MOVL    R0, R5                  ; Use TQE as a fork block
                                1835  4226 ;
                                1835  4227 ; Use the TQE as a fork block to start the timer. This is because the
                                1835  4228 ; timer must be started at or below QUEUEAST to synchronize access to the
                                1835  4229 ; TIMER QUEUE.
                                1835  4230 ;
            06  90              1835  4231          MOVB    #IPL$_QUEUEAST,-        ; Set delivery IPL
      0B    A5                  1837  4232                  FKB$B_FIPL(R5)          ;  of fork block
            08  10              1839  4233          BSBB    40$                     ; Create fork process
E8F8 CF     01  CE              183B  4234          MNEGL   #1,GHB$L_TIM_ACT+GHB_AREA ; Mark the timer as active for LATCP
            38  BA              1840  4235          POPR    #^M<R3,R4,R5>           ; Restore our context
            05  1842  4236 10$:    RSB                                             ; And return
                                1843  4237
                                1843  4238 ;
                                1843  4239 ;       Fork process to start the timer.  We need to go to synch from
                                1843  4240 ;       fork ipl to start the timer.  We do this by going through
                                1843  4241 ;       Queueast IPL to be safe - using the TQE as a fork block.
                                1843  4242 ;
                                1843  4243
00000000'GF     16              1843  4244 40$:    JSB     G^EXE$FORK              ; Fork to start the timer ticking
                                1849  4245          DSBINT  #IPL$_SYNCH             ; Raise to synch
    188C'CF     9E              184F  4246          MOVAB   W^LT$TICK,-             ;  Address of tick routine
      0C    A5                  1853  4247                  TQE$L_FPC(R5)           ;  is the fork pc
00000000'GF     D0              1855  4248          MOVL    G^TTY$GL_DELTA,-        ;  The tick time is from tty sysgen
      20    A5                  1858  4249                  TQE$Q_DELTA(R5)         ;  parameter for modem control
            05  90              185D  4250          MOVB    #TQE$C_SSREPT,-         ;  Repeating tick
      0B    A5                  185F  4251                  TQE$B_RQTYPE(R5)        ;
```

```
                        1861  4252          MOVQ    G^EXE$GQ_SYSTIME, R0    ; Set the time
                        1867
                        1868  4253          ADDL    G^TTY$GL_DELTA, R0      ;    from now
                        186E
                        186F  4254          ADWC    #0, R1                  ; Carry to top longword
                        1872  4255          JSB     G^EXE$INSTIMQ           ; Queue the timer element
                        1878  4256          ENBINT                          ; Restore IPL and leave
                        187B  4257          RSB
```

```
                        187C  4259          .SBTTL  LT$TICK  - Timer Service Routine
                        187C  4260 ;++
                        187C  4261 ; LT$TICK - Timer Service Routine
                        187C  4262 ;
                        187C  4263 ; Functional description:
                        187C  4264 ;
                        187C  4265 ;       Enter here on each timer tick.  We scan the CSB table and
                        187C  4266 ;       timeout each UCB with a matching CSB for each CSB that has
                        187C  4267 ;       timed out.  For each receive on a circuit that has not timed
                        187C  4268 ;       out we will reset its timer.
                        187C  4269 ;
                        187C  4270 ; Inputs:
                        187C  4271 ;       R5 = TQE address
                        187C  4272 ;
                        187C  4273 ;       IPL = TIMER
                        187C  4274 ;
                        187C  4275 ; Outputs:
                        187C  4276 ;       R0-R4 are destroyed.
                        187C  4277 ;
                        187C  4278 ;--
                        187C  4279          .ENABLE LOCAL_BLOCK
                        187C  4280
                        187C  4281 STOP_TQE:                                    ; Stop the TQE
      0B A5   04   8A   187C  4282          BICB    #TQE$M_REPEAT,TQE$B_RQTYPE(R5)  ; Clear the repeat flag
  000D'CF     04   8A   1880  4283          BICB    #DPT$M_NOUNLOAD,DPT$TAB+DPT$B_FLAGS ; Make driver reloadable again
      EBAF CF     D4   1885  4284          CLRL    GHB$L_TIM_ACT+GHB_AREA          ; Tell LATCP we are no longer
                        1889  4285                                                  ;    active
         01DB   31   1889  4286          BRW     300$                            ; And leave
                        188C  4287
                        188C  4288 LT$TICK:
                        188C  4289          DSBINT  #LAT$C_IPL                      ; Raise to synchronize LAT
      0FE0 8F   BB   1892  4290          PUSHR   #^M<R5,R6,R7,R8,R9,R10,R11>     ; Save all the registers
           01   E1   1896  4291          BBC     #GHB_STS_V_ACTIVE,-             ; Br if we went inactive
   E0 E8FD CF        1898  4292                   GHB_B_STATUS,STOP_TQE
                        189C  4293 ;
                        189C  4294 ; Scan all CSBs to retransmit in balanced mode and time out circuits
                        189C  4295 ; that appear to be dead.
                        189C  4296 ;
      5B   20   D0   189C  4297          MOVL    #LAT$C_MAX_CSBS, R11            ; Scan the CSB table for
5A    E795 CF   DE   189F  4298          MOVAL   GHB_L_CSBTABLE, R10             ; Active entries and check
      58   8A   D0   18A4  4299 20$:     MOVL    (R10)+ , R8                     ; Get the CSB address
           43   13   18A7  4300          BEQL    90$                             ; Br if no CSB in this slot
                        18A9  4301 ;
                        18A9  4302 ;       Are we coming out of balanced mode and need to retransmit the
                        18A9  4303 ;       First message again??
                        18A9  4304 ;
   02   0B A8   91   18A9  4305          CMPB    CSB_B_STATE(R8), -              ; State must be running
                        18AD  4306                  #CSB_C_STATE_RUN            ;    for active circuits
           26   12   18AD  4307          BNEQ    60$                             ; Br if not
         52 A8  B5   18AF  4308          TSTW    CSB_W_XMTTMO(R8)                ; Timing out restart message?
           21   13   18B2  4309          BEQL    60$                             ; Br if no
         52 A8  B7   18B4  4310          DECW    CSB_W_XMTTMO(R8)                ; Check for timeout
           1C   14   18B7  4311          BGTR    60$                             ; Br if not time yet
                        18B9  4312          INC_CTR GHB$L_RETRANS+GHB_AREA          ; Count a retransmission
                        18C3  4313          INC_CTR CSB_Z_LCB+LCB_L_MSG_REXMT(R8)   ;    ...
      54   58   D0   18CB  4314          MOVL    R8, R4                          ; CSB address for call
              0203   30   18CE  4315          BSBW    LT$REXMIT                       ; Transmit the message
```

```
                52 A8   02    B0  18D1  4316          MOVW    #LAT$C_HOST_TIMER,CSB_W_XMTTMO(R8) ; Reset the timeout
                                  18D5  4317 ;
                                  18D5  4318 ; Timeout circuit if needed
                                  18D5  4319 ;
                    50 A8   B7    18D5  4320 60$:     DECW    CSB_W_TIMEOUT(R8)                  ; Check the circuit timeout
                       12   12    18D8  4321          BNEQ    90$                                ; Br if not expired
                                  18DA  4322          INC_CTR GHB$L_CIRCDOWN+GHB_AREA            ; Count circ down event
                     FE43   30    18E4  4323          BSBW    LT$CIRCDEAD                        ; Declare the circuit dead
                    66 A8   B0    18E7  4324          MOVW    CSB_W_TIMRESET(R8),-               ; Reset the timeout to wait
                    50 A8                             CSB_W_TIMEOUT(R8)                  ; a while.
                    B5 5B   F5    18EC  4326 90$:     SOBGTR  R11, 20$                           ; Loop till we are done
                                  18EF  4327
                                  18EF  4328 ;
                                  18EF  4329 ;       Now check to see if there are dead ucbs waiting to be cleaned up
                                  18EF  4330 ;       and if noone is using the io database right now, we can delete them
                                  18EF  4331 ;       all in a flash.
                                  18EF  4332 ;
                                  18EF  4333
          00000000'GF    DE       18EF  4334 100$:    MOVAL   G^IOC$GL_MUTEX, R0                 ; Is anyone owning the io data
                         50       18F5
                FFFF 8F   60   B1  18F6  4335          CMPW    MTX$W_OWNCNT(R0), #-1              ; base?  If so, then we must
                          1A   12  18FB  4336          BNEQ    130$                               ; try again later.
                54   E883 CF   D0  18FD  4337          MOVL    GHB_L_DEADLINK, R4                 ; Good we own it then
                     E87E CF   D4  1902  4338          CLRL    GHB_L_DEADLINK                     ; Reset list pointer
                          54   D5  1906  4339 110$:     TSTL    R4                                 ; Any work to do?
                          0D   13  1908  4340          BEQL    130$                               ; All done
                     55   54   D0  190A  4341          MOVL    R4, R5                             ; UCB to delete
                54   0138 C5   D0  190D  4342          MOVL    UCB$L_LT_DEADLINK(R5), R4          ; Save link to next one
                        FCOA   30  1912  4343          BSBW    LT$DESTROYUCB                      ; Destroy this UCB
                          EF   11  1915  4344          BRB     110$                               ; and go back for more
                                   1917  4345
                                   1917  4346 130$:
                                   1917  4347
                                   1917  4348 ;
                                   1917  4349 ;       Transmit a configuration message if its time.  We tell all
                                   1917  4350 ;       concentrators our node name, announcement string every
                                   1917  4351 ;       so often so they know we exist.
                                   1917  4352 ;
                                   1917  4353
                    EAD3 CF    B7  1917  4354          DECW    GHB_W_MULTIMR                      ; Time to transmit?
                         03    15  191B  4355          BLEQ    150$                               ; Br if yes, do it
                       0147    31  191D  4356 140$:     BRW     300$                               ; Else, still time to go
                                   1920  4357
                52   EACC CF    D0  1920  4358 150$:     MOVL    GHB_L_MULTIBFR, R2                 ; Get multicast buffer
                          0D   13  1925  4359          BEQL    155$                               ; Br if none
                                   1927  4360          ASSUME  GHB_STS_V_MULTI EQ 0
                  F1 E86D CF   E8  1927  4361          BLBS    GHB_B_STATUS,140$                  ; Ignore, if already in progress
                     E869 CF   91  192C  4362          CMPB    GHB_B_INCARN, -                    ; Has it been rebuilt?
                          4E A2                             XMT_B_MC_INCARN(R2)                ;  if so, incarn has been
                          06   13  1932  4364          BEQL    160$                               ;  bumped to say so
                        F02E   30  1934  4365 155$:     BSBW    LT$SETENTRY                        ; Go rebuild the message
                     E3 50     E9  1937  4366          BLBC    R0, 140$                           ; Br if failure
                    E860 CF    9B  193A  4367 160$:     MOVZBW  <XMT_B_MC_MULTIMR-XMT_B_MC_SET>-   ; Reset multicast timer
                    EAAD CF        193E
                                   1941  4368                    +GHB_T_MC_DATA,GHB_W_MULTIMR   ;     from LATCP set data
                          05   12  1941  4369          BNEQ    161$                               ; Br if value is okay
```

```
                     EAA6 CF   0A   B0  1943  4370          MOVW    #LAT$C_MULTI_TIMER,GHB_W_MULTIMR ; Reset timer
                52   EAA4 CF   D0  1948  4371  161$:       MOVL    GHB_L_MULTIBFR, R2               ; Get multicast buffer
                                   194D  4372  ;
                                   194D  4373  ; Adjust host rating
                                   194D  4374  ;
                   51   54 A2   9E  194D  4375              MOVAB   XMT_B_MC_GRP_LEN(R2), R1         ; Compute address of node name:
                     50   81   9A  1951  4376              MOVZBL  (R1)+, R0                        ;   get count of groups
                     51   50   C0  1954  4377              ADDL    R0, R1                           ;   add in start address
                     50   81   9A  1957  4378              MOVZBL  (R1)+, R0                        ; Compute address of node desc.
                     51   50   C0  195A  4379              ADDL    R0, R1                           ;   add in start address
                     50   81   9A  195D  4380              MOVZBL  (R1)+, R0                        ; Calc start address of services
                     51   50   C0  1960  4381              ADDL    R0, R1                           ;   add in start address
                     54   51   D0  1963  4382              MOVL    R1, R4                           ; Save address of service count
                     02   81   91  1966  4383              CMPB    (R1)+, #2                        ; Do we have two service names?
                          03   18  1969  4384              BGEQ    163$                             ; Br if yes, process it
                        00BE   31  196B  4385              BRW     170$                             ; Else, skip it
                          51   D6  196E  4386  163$:       INCL    R1                               ; Skip rating byte
                     50   81   9A  1970  4387              MOVZBL  (R1)+, R0                        ; Calc start of first svc desc.
                     51   50   C0  1973  4388              ADDL    R0, R1                           ;   ...
                     50   81   9A  1976  4389              MOVZBL  (R1)+, R0                        ; Calc start of second svc rate
                     51   50   C0  1979  4390              ADDL    R0, R1                           ;     ...
                                   197C  4391              ;
                                   197C  4392              ;  Compute the service rating as follows:
                                   197C  4393              ;
                                   197C  4394              ;       cpu_idle_time * 255
                                   197C  4395              ;     ---------------------- * cpu_weight
                                   197C  4396              ;     100 * multicast timer
                                   197C  4397              ;
                                   197C  4398              ;
                                   197C  4399              ;   where:  cpu_weight is 1 for a 790, 1/2 for 785, 3/8 for 780, etc.
                                   197C  4400              ;
                                   197C  4401              ;
                     E810 CF   C3  197C  4402              SUBL3   GHB_L_LASTCPU,@GHB_L_NULLCPU,R0  ; Calculate "idle time"
                50   E809 DF       1980
                     E804 DF   D0  1984  4403              MOVL    @GHB_L_NULLCPU,GHB_L_LASTCPU     ; Set new last cpu time
                     E805 CF       1988
                00000000'GF   D0  198B  4404              MOVL    G^EXE$GL_MP,R6                   ; Get ptr to MP code
                          56       1991
                          17   13  1992  4405              BEQL    164$                             ; Br if not there
                     E7FC CF   C3  1994  4406              SUBL3   GHB_L_NULLSEC,-                  ; Else, compute secondary idle
                57   0000'C6       1998  4407                MPS$GL_NULLCPU(R6),R7                  ;   time
                     0000'C6   D0  199C  4408              MOVL    MPS$GL_NULLCPU(R6),-             ; And set new secondary
                     E7F1 CF       19A0  4409                GHB_L_NULLSEC                          ;   null time
                  57   FF 8F   78  19A3  4410              ASHL    #-1,R7,R7                        ; Get 50% of secondary null time
                          57       19A7
                     50   57   C2  19A8  4411              SUBL    R7,R0                            ; Let secondary account for
                                   19AB  4412                                                       ;   only 50% additional power
                000000FF 8F   C4  19AB  4413  164$:       MULL    #255,R0                          ; Multiply by highest rating possibl
                          50       19B1
                     53   01   9A  19B2  4414              MOVZBL  #1,R3                            ; Start with 1
                                   19B5  4415              CPUDISP <<790, EIGHT>,-                  ; Give 790 the highest rating
                                   19B5  4416                      <785, FOUR>,-                    ; Rate 785 as four
                                   19B5  4417                      <780, THREE>,-                   ; Rate 780 as three
                                   19B5  4418                      <750, TWO>,-                     ; Rate 750 as two
;JAY0001                           19B5     1                      <UV2, TWO>,-                     ; Rate MicroVAX II as two
                                   19B5  4419                      <730, ONE>,-                     ; etc.
                                   19B5  4420                      <UV1, ONE>>                      ;
```

```
                        19E7  4421
      53   04   C0       19E7  4422 EIGHT:  ADDL    #4,R3                          ; Rating will be highest
           53   D6       19EA  4423 FOUR:   INCL    R3                             ; next rating
           53   D6       19EC  4424 THREE:  INCL    R3                             ;   etc.
           53   D6       19EE  4425 TWO:    INCL    R3                             ;
                         19F0  4426 ONE:
      50   53   C4       19F0  4427         MULL    R3,R0                          ; Compute adjusted rating
00000320   8F   C6       19F3  4428         DIVL    #800,R0                        ; Divide by highest rating + %
           50             19F9
53   E7A0   CF   9A       19FA  4429         MOVZBL  <XMT_B_MC_MULTIMR-XMT_B_MC_SET>- ; Get multicast timer
                         19FF  4430          +GHB_T_MC_DATA,R3
      50   53   C6       19FF  4431         DIVL    R3,R0                          ; Normalize rating by time
           0D   13       1A02  4432         BEQL    165$                           ; Br if zero rating
           50   D1       1A04  4433         CMPL    R0,#255                        ; Ridiculous rating ?
000000FF   8F             1A06
           06   15       1A0B  4434         BLEQ    167$                           ; Br if no, continue
 50    FE   8F   9A       1A0D  4435         MOVZBL  #254,R0                        ; Else, set highest rating
                         1A11  4436                                                ;  let INCL up it to 255
                         1A11  4437 165$:   ;
                         1A11  4438         ;  Don't let rating fall to zero, unless the joblimit is reached
                         1A11  4439         ;  or the jobcnt if greater than or eqal to the joblim.
                         1A11  4440         ;
           50   D6       1A11  4441         INCL    R0                             ; Else, set to minimum
00000000'GF   B1         1A13  4442 167$:   CMPW    G^SYS$GW_IJOBCNT,G^SYS$GW_IJOBLIM ; Are we maxed out?
00000000'GF               1A19
           02   1F       1A1E  4443         BLSSU   169$                           ; Br if no, continue
           50   D4       1A20  4444         CLRL    R0                             ; Else, set to mimimum
      61   50   91       1A22  4445 169$:   CMPB    R0,(R1)                        ; New second service rating?
           05   13       1A25  4446         BEQL    170$                           ; Br if no
      61   50   90       1A27  4447         MOVB    R0,(R1)                        ; Else, set new 2nd svc rating
           0A   11       1A2A  4448         BRB     175$
                         1A2C  4449
00000000'GF   91         1A2C  4450 170$:   CMPB    G^SYS$GW_IJOBCNT, 1(R4)        ; New number of interactive
      01   A4             1A32
                         1A34  4451                                                ;     users?
           13   13       1A34  4452         BEQL    180$                           ; Br if no
00000000'GF   90         1A36  4453 175$:   MOVB    G^SYS$GW_IJOBCNT, 1(R4)        ; Use the interactive count
      01   A4             1A3C
                         1A3E  4454                                                ;  as the second rating
      4E   A2   96       1A3E  4455         INCB    XMT_B_MC_INCARN(R2)            ; set new incarnation
    E754   CF   96       1A41  4456         INCB    GHB_B_INCARN                   ; no call LT$SETENTRY next time
 4F A2     08   8C       1A45  4457         XORB    #XMT_CHFLG_M_RATE,XMT_B_MC_CHG_FLAG(R2) ; flag rate change
                         1A49  4458
 54   E73B   CF   D0       1A49  4459 180$:   MOVL    GHB_L_FFI, R4                  ; Get FFI block address
                17   13       1A4E  4460         BEQL    300$                           ; Br if none
                         1A50  4461         SETBIT  #GHB_STS_V_MULTI,GHB_B_STATUS  ; Indicate that multicast is active
      53   52   D0       1A56  4462         MOVL    R2,R3                          ; Copy CXB address
    E988   CF   7D       1A59  4463         MOVQ    GHB_Q_MULTIADD,-               ; Copy the destination address
      28   A3             1A5D  4464         CXB$Q_STATION(R3)                      ;
                         1A5F  4465         ASSUME  CXB$W_BCNT EQ CXB$W_BOFF+2
14 A3    18   A3   D0       1A5F  4466         MOVL    CXB$W_BOFF(R3),CXB$L_T_SAVE(R3) ; Save BOFF and BCNT
                         1A64  4467
      10   B4   16       1A64  4468         JSB     @FFI$L_XMIT(R4)                ; Start XMIT operation
                         1A67  4469
    0FE0   8F   BA       1A67  4470 300$:   POPR    #^M<R5,R6,R7,R8,R9,R10,R11>    ; Restore the regs
                         1A68  4471         ENBINT                                 ; Restore priority
                05   1A6E  4472         RSB
```

```
                        1A6F  4473
                        1A6F  4474          .DISABLE LOCAL_BLOCK
```

```
                     1A6F  4476          .SBTTL  LT$XMIT  - Transmit a message
                     1A6F  4477 ;++
                     1A6F  4478 ; LT$XMIT - Transmit a message
                     1A6F  4479 ;
                     1A6F  4480 ; FUNCTIONAL DESCRIPTION:
                     1A6F  4481 ;
                     1A6F  4482 ;       If the CXB is outstanding then record the fact that there is
                     1A6F  4483 ;       a waiting message.  If the CXB is not outstanding, then ready
                     1A6F  4484 ;       it for transmission and send it to the other driver as a write.
                     1A6F  4485 ;
                     1A6F  4486 ;       Transmissions must make progress.  That is the sequence and acks
                     1A6F  4487 ;       of the messages must change over time or we decide that the
                     1A6F  4488 ;       circuit is really deadlocked and we take it down.  We are generous
                     1A6F  4489 ;       with the counter for detecting this deadlock.
                     1A6F  4490 ;
                     1A6F  4491 ; Input:
                     1A6F  4492 ;       R4 = CSB address
                     1A6F  4493 ;
                     1A6F  4494 ; Output:
                     1A6F  4495 ;       R0-R5 clobbered
                     1A6F  4496 ;-
                     1A6F  4497
                     1A6F  4498          .ENABLE LOCAL_BLOCK
                     1A6F  4499
                     1A6F  4500 LT$XMIT:
53   34 B4    0F     1A6F  4501          REMQUE  @CSB_Q_XBUFQ(R4), R3    ; Remove the buffer from queue
        06    1C     1A73  4502          BVC     5$                      ; We had better have a buffer here.
                     1A75  4503          DEBUG                           ; Fatal error, no buffer
     59 A4    96     1A7B  4504 5$:      INCB    CSB_B_XMTCNT(R4)        ; One more buffer to transmit
     62 A4    96     1A7E  4505          INCB    CSB_B_XSEQ(R4)          ; Next message to send
     5C A4    7D     1A81  4506          MOVQ    CSB_T_CIRCHDR(R4),-     ; Set the circuit header for the
     48 A3           1A84  4507                  XMT_T_DATA(R3)          ; message
50   48 A3    9E     1A86  4508          MOVAB   XMT_T_DATA(R3), R0      ; Start of data to send
1C A3    50    C2     1A8A  4509          SUBL    R0, CXB$L_T_ENDADR(R3)  ; Make length from end address
     1C A3    B0     1A8E  4510          MOVW    CXB$L_T_ENDADR(R3),-    ; Copy length to CXB format
     1A A3           1A91  4511                  CXB$W_BCNT(R3)
40 B4    63    0E     1A93  4512          INSQUE  (R3), @CSB_Q_XWAITQ+4(R4) ; Queue to tail of waiting buffers
     58 A4    95     1A97  4513          TSTB    CSB_B_XMTBSY(R4)        ; Transmitter busy?
        3D    13     1A9A  4514          BEQL    60$                     ; Br if no, start transmitting
     58 A4    96     1A9C  4515 10$:     INCB    CSB_B_XMTBSY(R4)        ; Else, count another transmit
                     1A9F  4516                                          ;  we just placed one on the wait queue
              05     1A9F  4517 20$:     RSB
                     1AA0  4518
                     1AA0  4519 ;
                     1AA0  4520 ;       Transmit just the one buffer, not all the ones waiting.
                     1AA0  4521 ;
                     1AA0  4522 LT$XMITONE:
53   34 B4    0F     1AA0  4523          REMQUE  @CSB_Q_XBUFQ(R4), R3    ; Get the one buffer of interest
        06    1C     1AA4  4524          BVC     30$                     ; Got it
                     1AA6  4525          DEBUG                           ; No buffer for this transmit
     59 A4    96     1AAC  4526 30$:     INCB    CSB_B_XMTCNT(R4)        ; Count one more on wait queue
     62 A4    96     1AAF  4527          INCB    CSB_B_XSEQ(R4)          ; Next message to send
     5C A4    7D     1AB2  4528          MOVQ    CSB_T_CIRCHDR(R4),-     ; Set the circuit header for the
     48 A3           1AB5  4529                  XMT_T_DATA(R3)          ; message
50   48 A3    9E     1AB7  4530          MOVAB   XMT_T_DATA(R3), R0      ; Start of data to send
1C A3    50    C2     1ABB  4531          SUBL    R0, CXB$L_T_ENDADR(R3)  ; Make length from end address
     1C A3    B0     1ABF  4532          MOVW    CXB$L_T_ENDADR(R3),-    ; Copy length to CXB format
```

```
                1A A3        1AC2  4533                    CXBSW_BCNT(R3)
                58 A4   95   1AC4  4534            TSTB    CSB_B_XMTBSY(R4)        ; Are we busy??
                   05   12   1AC7  4535            BNEQ    40$                     ; Br if yes, so carry on
                58 A4   96   1AC9  4536            INCB    CSB_B_XMTBSY(R4)        ; Just one buffer
                   25   11   1ACC  4537            BRB     90$                     ; Go
                        1ACE  4538
             40 B4      63   0E   1ACE  4539 40$:   INSQUE  (R3), @CSB_Q_XWAITQ+4(R4) ; Place it on the queue
                   C8   11   1AD2  4540            BRB     10$                     ; And transmit later
                        1AD4  4541
                        1AD4  4542 LTSREXMIT:                                   ; Entry to start a retransmit
                58 A4   95   1AD4  4543            TSTB    CSB_B_XMTBSY(R4)        ; Is the transmit busy??
                   C6   12   1AD7  4544            BNEQ    20$                     ; Br if yes, just leave
          58 A4    59 A4   90   1AD9  4545 60$:    MOVB    CSB_B_XMTCNT(R4), -     ; Start a series of retransmits
                                1ADE  4546                    CSB_B_XMTBSY(R4)        ;
                        1ADE  4547
                        1ADE  4548
                        1ADE  4549 ;;**    debug code
                59 A4   91   1ADE  4550            CMPB    CSB_B_XMTCNT(R4),-      ; All buffers on the transmit queue?
                   02        1AE1  4551                    #LATSC_XMT_BUFFERS      ;
                   09   19   1AE2  4552            BLSS    80$                     ; Nope
                   06   13   1AE4  4553            BEQL    70$                     ; Yes all of them
                        1AE6  4554            DEBUG                           ; More than all of them
                        1AEC  4555 70$:     NOP
                        1AED  4556 80$:
                        1AED  4557 ;;**    debug code
                        1AED  4558
                        1AED  4559 LTSXMTGO:                                    ; Enter to transmit for waiters
             53    3C B4   0F   1AED  4560            REMQUE  @CSB_Q_XWAITQ(R4), R3   ; Obtain the next waiting buffer
                   AC   1D   1AF1  4561            BVS     20$                     ; Br if no buffer to transmit
                56 A4   B1   1AF3  4562 90$:     CMPW    CSB_W_PROGSEQ(R4),-     ; Are we making progress?
                62 A4        1AF6  4563                    CSB_W_XSEQ(R4)          ; are the seq and ack fields changing?
                   1F   12   1AF8  4564            BNEQ    100$                    ; Br if yes, splendid
                54 A4   B7   1AFA  4565            DECW    CSB_W_PROGRESS(R4)      ; Nope, give them some time then
                   23   14   1AFD  4566            BGTR    110$                    ; We have more time then
             34 A4      63   0E   1AFF  4567            INSQUE  (R3), CSB_Q_XBUFQ(R4)   ; Put the buffer back into the CSB
                        1B03  4568            INC_CTR GMB$L_CIRCDOWN+GMB_AREA ; Count a circuit timeout
                   58   DD   1B0D  4569            PUSHL   R8                      ; No more time.  Clobber the circuit
                58      54   D0   1B0F  4570            MOVL    R4, R8                  ; Since it appears to be deadlocked.
                   FC15      30   1B12  4571            BSBW    LTSCIRCDEAD             ; and clean it all up
                   58 8ED0   1B15  4572            POPL    R8                      ;
                        05   1B18  4573            RSB
                        1B19  4574
                62 A4   B0   1B19  4575 100$:    MOVW    CSB_W_XSEQ(R4),-        ; Reset the current sequence and ack
                56 A4        1B1C  4576                    CSB_W_PROGSEQ(R4)       ; to check against
                   14   B0   1B1E  4577            MOVW    #LATSC_PROGRESS,-       ; and reset the progress counter
                54 A4        1B20  4578                    CSB_W_PROGRESS(R4)      ;
                        1B22  4579
                        1B22  4580 110$:
                        1B22  4581 ;
                        1B22  4582 ;       Build the CXB to transmit
                        1B22  4583 ;
          55    E662 CF   D0   1B22  4584            MOVL    GMB_L_FFI, R5           ; Get the FFI block address
                   2A   13   1B27  4585            BEQL    150$                    ; Br if none
            51    5A A4   9A   1B29  4586            MOVZBL  CSB_B_XCXB_INX(R4), R1  ; Get XMIT CXB index slot
          44 A441      53   D0   1B2D  4587            MOVL    R3, CSB_L_XCXB(R4)[R1]  ; Save address of CXB
                   51   96   1B32  4588            INCB    R1                      ; Skip to next slot
            51    FE 8F   8A   1B34  4589            BICB    #^C<LATSC_XMT_BUFFERS-1>,R1 ; Modulo maximum
```

```
                                1B38  4590          MOVB    R1, CSB_B_XCXB_INX(R4)  ; Save for next time
                          54    1B3C  4591          PUSHL   R4                      ; Save R4
                          54    1B3E  4592          MOVL    R5,R4                   ; Else, copy the FFI block address
                                1B41  4593          ASSUME  CXBSW_BCNT EQ CXBSW_BOFF+2
                          18 A3 1B41  4594          MOVL    CXBSW_BOFF(R3),CXBSB_T_SAVE(R3) ; Save BOFF and BCNT
                                1B46  4595                                          ;  progress
                                1B46  4596          .IF DEFINED LT_HISTORY
                                1B46  4597 ;
                                1B46  4598 ;        Save a history of transmits
                                1B46  4599 ;
                          48 A3 1B46  4600          MOVAB   XMT_B_FLAG(R3), R0      ; Address of data to save
                             EBDA 1B4A  4601          BSBW    LT$HISTORY              ; Save the data
                                1B4D  4602
                                1B4D  4603          .ENDC   ;; DEFINED LT_HISTORY
                                1B4D  4604
                          10 B4 1B4D  4605          JSB     @FFISL_XMIT(R4)         ; Start the XMIT operation
                          54 8ED0 1B50  4606          POPL    R4                      ; Restore R4
                             05   1B53  4607 150$:    RSB                             ; Return to caller
                                1B54  4608
                                1B54  4609          .DISABLE LOCAL_BLOCK
                                1B54  4610
```

```
                     1B54  4612          .SBTTL  LT$XMT_FFIDONE  - FFI Transmit done routine
                     1B54  4613 ;++
                     1B54  4614 ; LT$XMT_FFIDONE - FFI Transmit completion routine
                     1B54  4615 ;
                     1B54  4616 ; FUNCTIONAL DESCRIPTION:
                     1B54  4617 ;
                     1B54  4618 ;       Post routine for a transmit.  Called directly from datalink
                     1B54  4619 ;       driver with CXB address in R3.  We have r0 - r5 to work with.
                     1B54  4620 ;
                     1B54  4621 ; INPUTS:
                     1B54  4622 ;       R0 =  Status of XMIT request
                     1B54  4623 ;       R3 = CXB address
                     1B54  4624 ;       R4 = FFI block address
                     1B54  4625 ;
                     1B54  4626 ;       IPL = SYNCH
                     1B54  4627 ;
                     1B54  4628 ; OUTPUTS:
                     1B54  4629 ;       R1,R2,R4,R5 clobbered
                     1B54  4630 ;--
                     1B54  4631          ASSUME  IPL$_SYNCH EQ LAT$C_IPL
                     1B54  4632 LT$XMT_FFIDONE:
           36   BB  1B54  4633          PUSHR   #^M<R1,R2,R4,R5>        ; Save registers
                1B56  4634          ASSUME  CXB$W_BCNT EQ CXB$W_BOFF+2
 18 A3   14 A3   D0  1B56  4635          MOVL    CXB$L_T_SAVE(R3),CXB$W_BOFF(R3) ; Restore BOFF and BCNT
                1B5B  4636          INC_CTR GHB$L_XCOUNT+GHB_AREA   ; Count one more buffer away
        0F 50   E8  1B65  4637          BLBS    R0, 20$                 ; Br if no error
  E5F3 CF   50   3C  1B68  4638          MOVZWL  R0,GHB$L_XERRCOD+GHB_AREA ; Save error code
                1B6D  4639          INC_CTR GHB$L_XERR+GHB_AREA     ; Count one more error, but ignore it
     4C 0B A3   E8  1B77  4640 20$:     BLBS    CXB$B_CODE(R3),110$     ; Br if multicast buffer
   54   0C A3   D0  1B7B  4641          MOVL    CXB$L_T_CSB(R3),R4      ; Get CSB address
           08   12  1B7F  4642          BNEQ    40$                     ; Got it, continue
        50   53   D0  1B81  4643          MOVL    R3,R0                   ; Toss this CXB then since the
          0052   30  1B84  4644          BSBW    LT$DEALPOOL             ;   CSB seems to be gone.
           38   11  1B87  4645          BRB     70$                     ; Leave quietly
                1B89  4646
        50   01   9A  1B89  4647 40$:     MOVZBL  #LAT$C_XMT_BUFFERS-1,R0 ; Get number of CXBs
 44 A440   53   D1  1B8C  4648 45$:     CMPL    R3,CSB_L_XCXB(R4)[R0]   ; Same as we sent?
           09   13  1B91  4649          BEQL    50$                     ; Br if yes, remove it
        F6 50   F4  1B93  4650          SOBGEQ  R0,45$                  ; Loop if more
                1B96  4651          DEBUG                           ;;&& oops
     44 A440   D4  1B9C  4652 50$:     CLRL    CSB_L_XCXB(R4)[R0]      ; Zero entry
                1BA0  4653          INC_CTR CSB_Z_LCB+LCB_L_MSG_XMT(R4) ; Count one more transmit
  40 B4   63   0E  1BA8  4654          INSQUE  (R3), @CSB_Q_XWAITQ+4(R4) ; Queue to tail for retransmit
        58 A4   97  1BAC  4655          DECB    CSB_B_XMTBSY(R4)        ; Any more to transmit??
           05   13  1BAF  4656          BEQL    60$                     ; Nope all done
        FF39   30  1BB1  4657          BSBW    LT$XMTGO                ; Transmit for waiter
           0E   11  1BB4  4658          BRB     70$                     ; Not done transmitting all yet
                1BB6  4659 60$:
                1BB6  4660 ;
                1BB6  4661 ; If all UCBs are gone and we have transmitted a stop message last,
                1BB6  4662 ; then we can make this CSB go away.
                1BB6  4663 ;
                1BB6  4664
        2D A4   95  1BB6  4665          TSTB    CSB_B_REFC(R4)          ; Are all the UCB's gone?
           09   12  1BB9  4666          BNEQ    70$                     ;
        5C A4   91  1BBB  4667          CMPB    CSB_B_FLAG(R4),-        ; Is the circuit dead?
           08       1BBE  4668                  #MTYP_C_HALT@FLAG_V_MTYPE
```

```
                1BBF  4669              BNEQ    70$                     ; Br if not
F85C    30      1BC1  4670              BSBW    LT$DEALOCSB             ; Get rid of the circuit block
  36    BA      1BC4  4671  70$:        POPR    #^M<R1,R2,R4,R5>        ; Restore registers
        05      1BC6  4672              RSB
                1BC7  4673              ;
                1BC7  4674              ; Multicast buffer completed
                1BC7  4675              ;
                1BC7  4676  110$:       CLRBIT  #GHB_STS_V_MULTI,-      ; Multicast is not busy now
                1BC7  4677                        GHB_B_STATUS
  F5    11      1BCD  4678              BRB     70$                     ; And exit
                1BCF  4679
```

```
                     1BCF  4681          .SBTTL  LT$FFIPOSTDONE  - Garbage transmit posting routine
                     1BCF  4682 ;++
                     1BCF  4683 ; LT$FFIPOSTDONE - Garbage transmit posting routine
                     1BCF  4684 ;
                     1BCF  4685 ; FUNCTIONAL DESCRIPTION:
                     1BCF  4686 ;
                     1BCF  4687 ;       Final post routine to dry up a CXB I/O when we are done
                     1BCF  4688 ;       with the channel.  We simply deallocate any transmit buffer returned.
                     1BCF  4689 ;
                     1BCF  4690 ; INPUTS:
                     1BCF  4691 ;       R3 = CXB address
                     1BCF  4692 ;       R4 = FFI block address
                     1BCF  4693 ;
                     1BCF  4694 ; OUTPUTS:
                     1BCF  4695 ;       R0 clobbered
                     1BCF  4696 ;--
                     1BCF  4697
                     1BCF  4698 LT$FFIPOSTDONE:
          36   BB    1BCF  4699          PUSHR   #^M<R1,R2,R4,R5>        ; Save registers
     50   53   D0    1BD1  4700          MOVL    R3,R0                   ; Copy CXB address
          03   10    1BD4  4701          BSBB    LT$DEALPOOL             ; Drop the buffer
          36   BA    1BD6  4702          POPR    #^M<R1,R2,R4,R5>        ; Restore registers
                     1BD8  4703 LT$DROPCXB:                             ; Drop all RECEIVE CXBs
               05    1BD8  4704          RSB
                     1BD9  4705
                     1BD9  4706 LT$DEALPOOL:
00000000'GF    17    1BD9  4707          JMP     G^EXE$DEANONPAGED       ; Deallocate the FFI block
                     1BDF  4708
                     1BDF  4709 LT$END:                                 ; End of driver
                     1BDF  4710
                     1BDF  4711          .END
```

```
$$$                             = 00000020 R      02
$$BASE                          = 00000001
$$DISPL                         = 0000000A
$$GENSW                         = 00000001
$$HIGH                          = 00000009
$$LIMIT                         = 00000008
$$LOW                           = 00000001
$$MNSW                          = 00000001
$$MXSW                          = 00000001
$$OP                            = 00000002
AT$_NULL                          ********    X   02
BIT...                          = 00000004
BUG$_UNSUPRTCPU                   ********    X   03
BUILD_TQE                         000008E2 R      03
CLASS_DDT                       = 00000010
CLASS_DISCONNECT                = 00000018
CLASS_GETNXT                    = 00000000
CLASS_PUTNXT                    = 00000004
CLASS_READERROR                 = 00000014
CLASS_SETUP_UCB                 = 00000008
CRB$L_INTD                      = 00000024
CRB$W_REFC                      = 0000000C
CREDIT_CHECK                      000010E2 R      03
CSB_B_FLAG                        0000005C  G
CSB_B_INX                         00000068  G
CSB_B_LOCIDN                      00000060  G
CSB_B_LOCIDS                      00000061  G
CSB_B_MAX_SLOTS                   00000069  G
CSB_B_NUM_SLOTS                   0000005D  G
CSB_B_RACK                        00000065  G
CSB_B_REFC                        0000002D  G
CSB_B_REMIDN                      0000005E  G
CSB_B_REMIDS                      0000005F  G
CSB_B_RSEQ                        00000064  G
CSB_B_SERVER                      0000001C  G
CSB_B_SPARE                       0000005B  G
CSB_B_STATE                       0000000B  G
CSB_B_TYPE                        0000000A  G
CSB_B_XACK                        00000063  G
CSB_B_XCXB_INX                    0000005A  G
CSB_B_XMTBSY                      00000058  G
CSB_B_XMTCNT                      00000059  G
CSB_B_XSEQ                        00000062  G
CSB_C_FIXLENGTH                   0000006C  G
CSB_C_STATE_HALT                = 00000000  G
CSB_C_STATE_RUN                 = 00000002  G
CSB_C_STATE_START               = 00000001  G
CSB_L_STOPREASON                  0000004C  G
CSB_L_UCBLST                      0000006C  G
CSB_L_XCXB                        00000044  G
CSB_Q_XBUFQ                       00000034  G
CSB_Q_XWAITQ                      0000003C  G
CSB_S_CIRCHDR                   = 00000008  G
CSB_T_CIRCHDR                     0000005C  G
CSB_T_SERVER                      0000001D  G
CSB_W_LOCID                       00000060  G
CSB_W_MAX_MSGSIZ                  0000006A  G
CSB_W_PROGRESS                    00000054  G
CSB_W_PROGSEQ                     00000056  G
CSB_W_REMID                       0000005E  G
CSB_W_RSEQ                        00000064  G
CSB_W_SIZE                        00000008  G
CSB_W_TIMEOUT                     00000050  G
CSB_W_TIMRESET                    00000066  G
CSB_W_XMTTMO                      00000052  G
CSB_W_XSEQ                        00000062  G
CSB_Z_DST                         0000002E  G
CSB_Z_LCB                         0000000C  G
CXB$B_CODE                      = 0000000B
CXB$B_TYPE                      = 0000000A
CXB$C_HEADER                    = 00000048
CXB$L_END_ACTION                = 0000001C
CXB$L_IRP                       = 00000014
CXB$L_T_CSB                       0000000C  G
CXB$L_T_ENDADR                    0000001C  G
CXB$L_T_SAVE                      00000014  G
CXB$Q_STATION                   = 00000028
CXB$W_BCNT                      = 0000001A
CXB$W_BOFF                      = 00000018
CXB$W_LENGTH                    = 0000000C
CXB$W_OFFSET                    = 0000000E
CXB$W_SIZE                      = 00000008
DC$_TERM                          ********    X   02
DDB$L_DDT                       = 0000000C
DDB$L_UCB                       = 00000004
DEFREASN                          00001752 R      03
DEV$M_AVL                       = 00040000
DEV$M_CCL                       = 00000002
DEV$M_IDV                       = 04000000
DEV$M_NNM                       = 00000200
DEV$M_ODV                       = 08000000
DEV$M_REC                       = 00000001
DEV$M_TRM                       = 00000004
DPT$B_FLAGS                     = 0000000D
DPT$C_LENGTH                    = 00000038
DPT$C_VERSION                   = 00000004
DPT$INITAB                        00000038 R      02
DPT$M_NOUNLOAD                  = 00000004
DPT$REINITAB                      000000D3 R      02
DPT$TAB                           00000000 R      02
DPT$W_VECTOR                    = 0000001E
DYN$C_BUFIO                     = 00000013
DYN$C_CDB                       = 00000033
DYN$C_CRB                       = 00000005
DYN$C_CXB                       = 0000001B
DYN$C_DDB                       = 00000006
DYN$C_DPT                       = 0000001E
DYN$C_ORB                       = 00000049
DYN$C_TQE                       = 0000000F
DYN$C_UCB                       = 00000010
ECHO_ERROR                        0000101A R      03
EIGHT                             000019E7 R      03
EXE$ALLOCTQE                      ********    X   03
EXE$ALONONPAGED                   ********    X   03
```

```
EXE$AL_TQENOREPT                ********  X   03
EXE$DEANONPAGED                 ********  X   03
EXE$FORK                        ********  X   03
EXE$GB_CPUDATA                  ********  X   03
EXE$GB_CPUTYPE                  ********  X   03
EXE$GL_MP                       ********  X   03
EXE$GQ_SYSTIME                  ********  X   03
EXE$INSTIMQ                     ********  X   03
FFI$B_SPARE                   = 0000000B
FFI$B_TYPE                    = 0000000A
FFI$C_LENGTH                  = 0000003E
FFI$L_BL                      = 00000004
FFI$L_CTX_DL                  = 0000000C
FFI$L_DL_UCB                  = 00000034
FFI$L_ERROR                   = 0000001C
FFI$L_FL                      = 00000000
FFI$L_PID                     = 00000038
FFI$L_RECV_DONE               = 00000018
FFI$L_SHUT_DONE               = 00000020
FFI$L_SPARE0                  = 00000024
FFI$L_SPARE1                  = 00000028
FFI$L_SPARE2                  = 0000002C
FFI$L_SPARE3                  = 00000030
FFI$L_XMIT                    = 00000010
FFI$L_XMIT_DONE               = 00000014
FFI$W_CHAN                    = 0000003C
FFI$W_SIZE                    = 00000008
FKB$B_FIPL                    = 0000000B
FLAG_M_MASTER                 = 00000002  G
FLAG_M_MTYPE                  = 000000FC  G
FLAG_M_RRF                    = 00000001  G
FLAG_S_MASTER                 = 00000001  G
FLAG_S_MTYPE                  = 00000006  G
FLAG_S_RRF                    = 00000001  G
FLAG_V_MASTER                 = 00000001  G
FLAG_V_MTYPE                  = 00000002  G
FLAG_V_RRF                    = 00000000  G
FOUR                            000019EA R      03
FUNCTAB_LEN                   = 00000000
GET_MORE                        00001196 R      03
GHB$B_LATECO                  = 00000003
GHB$B_LATVERSION              = 00000002
GHB$K_IDLEN                   = 00000040
GHB$K_NAMELEN                 = 00000010
GHB$L_CIRCDOWN                = 00000048
GHB$L_CSBLST                  = 00000020
GHB$L_DUPLMSG                 = 00000034
GHB$L_HISTORY                 = 00000008
GHB$L_NODE                    = 0000001C
GHB$L_NOXBFR                  = 00000058
GHB$L_PROTOCOL                = 0000004C
GHB$L_PROTOMASK               = 00000050
GHB$L_RCOUNT                  = 0000002C
GHB$L_RESOURCE                = 00000054
GHB$L_RETRANS                 = 00000044
GHB$L_SETENTRY                = 00000004
GHB$L_SHUTENTRY               = 0000000C
GHB$L_STRENTRY                = 00000010
GHB$L_TIM_ACT                 = 00000018
GHB$L_UCB                     = 00000014
GHB$L_XCOUNT                  = 00000038
GHB$L_XERR                    = 0000003C
GHB$L_XERRCOD                 = 00000040
GHB$Q_OLD_CSBS                = 00000024
GHB$S_COMM_AREA               = 00000060
GHB$T_CSBCTR                  = 0000000C
GHB$V_BADCREDITS              = 00000008
GHB$V_CSBINVALID              = 00000003
GHB$V_CSBRANGE                = 00000002
GHB$V_CSBSTALE                = 00000004
GHB$V_CSBZERO                 = 00000001
GHB$V_HALT                    = 00000005
GHB$V_INVALIDLOCID            = 00000007
GHB$V_INVALIDREMID            = 00000006
GHB$V_INVALIDSEQ              = 0000000A
GHB$V_REPCREATE               = 00000009
GHB$V_REPDISC                 = 0000000B
GHB$V_START                   = 00000000
GHB$W_VMSVERSION              = 00000000
GHB_AREA                        00000120 R      03
GHB_B_INCARN                    00000199 R      03
GHB_B_LOC_CCHK                  000000B8 R      03
GHB_B_MAX_SLOTS                 0000010C R      03
GHB_B_OLD_CSBCNT                0000019A R      03
GHB_B_RATE                      0000019B R      03
GHB_B_STATUS                    00000198 R      03
GHB_C_STRT_SLOTL              = 00000005
GHB_G_TQE                       000000DC R      03
GHB_L_AREA                      000004B8 R      03
GHB_L_CSBTABLE                  00000038 R      03
GHB_L_DEADLINK                  00000184 R      03
GHB_L_FFI                       00000188 R      03
GHB_L_LASTCPU                   00000190 R      03
GHB_L_MULTIBFR                  000003F0 R      03
GHB_L_NULLCPU                   0000018C R      03
GHB_L_NULLSEC                   00000194 R      03
GHB_L_UCBQ                      00000180 R      03
GHB_Q_MULTIADD                  000003E8 R      03
GHB_STS_M_ACTIVE              = 00000002  G
GHB_STS_M_MULTI               = 00000001  G
GHB_STS_M_SHUT                = 00000004  G
GHB_STS_M_TQE                 = 00000008  G
GHB_STS_S_ACTIVE              = 00000001  G
GHB_STS_S_MULTI               = 00000001  G
GHB_STS_S_SHUT                = 00000001  G
GHB_STS_S_TQE                 = 00000001  G
GHB_STS_V_ACTIVE              = 00000001  G
GHB_STS_V_MULTI               = 00000000  G
GHB_STS_V_SHUT                = 00000002  G
GHB_STS_V_TQE                 = 00000003  G
GHB_T_MAX_SLOT_TABLE            0000010D R      03
GHB_T_MC_DATA                   0000019E R      03
GHB_T_MC_MSG                    000003F4 R      03
GHB_T_STRT_MSG                  00000400 R      03
```

```
GHB_T_STRT_SLOT                   000004B1  R      03
GHB_T_STRT_VAR                    0000040C  R      03
GHB_W_MC_SIZE                     0000019C  R      03
GHB_W_MULTIMR                     000003EE  R      03
GHB_W_STRT_LEN                    000003FE  R      03
GOT_A_LOT                         000011A7  R      03
HBF_L_BUFEND                    =  00000004
HBF_L_NEXT                      =  00000000
HBF_Z_DATA                      =  0000000C
INB_B_SPARE                       0000000B  G
INB_B_TYPE                        0000000A  G
INB_C_LENGTH                      0000000C  G
INB_L_SPARE                       00000004  G
INB_W_BCNT                        00000002  G
INB_W_BOFF                        00000000  G
INB_W_SIZE                        00000008  G
IOC$CLONE_UCB                     ********    X    03
IOC$DELETE_UCB                    ********    X    03
IOC$GL_MUTEX                      ********    X    03
IOC$MNTVER                        ********    X    03
IOC$RETURN                        ********    X    03
IPL$_ASTDEL                     =  00000002
IPL$_QUEUEAST                   =  00000006
IPL$_SYNCH                      =  00000008
IPL$_TIMER                      =  00000008
LAT$C_CIR_TIMER                 =  00000050
LAT$C_CUR_ECO                   =  00000000
LAT$C_CUR_VER                   =  00000005
LAT$C_GRP_LEN                   =  00000020
LAT$C_HI_VER                    =  00000005
LAT$C_HOST_TIMER                =  00000002
LAT$C_IPL                       =  00000008
LAT$C_KEEP_CSB                  =  0000000A
LAT$C_LOC_LEN                   =  00000040
LAT$C_LO_VER                    =  00000005
LAT$C_MAX_CSBS                  =  00000020
LAT$C_MAX_MSGSIZ                =  000005DC
LAT$C_MAX_RCRED                 =  00000001
LAT$C_MAX_SERVICES              =  00000006
LAT$C_MAX_SLOTS                 =  00000040
LAT$C_MAX_SLOTSIZ               =  000000FF
LAT$C_MULTIADD1                 =  00000009
LAT$C_MULTIADD2                 =  0000002B
LAT$C_MULTIADD3                 =  00000F00
LAT$C_MULTI_TIMER               =  0000000A
LAT$C_PROGRESS                  =  00000014
LAT$C_SHUTDELAY                 =  004C4B40
LAT$C_SVC_LEN                   =  00000008
LAT$C_UCB_BUFSIZ                =  00000050
LAT$C_UCB_SLOTSIZ               =  00000004
LAT$C_UCB_TIMOUT                =  00000002
LAT$C_VMS                       =  00000003
LAT$C_VMS_VER                   =  00000004
LAT$C_XMT_BUFFERS               =  00000002
LAT_C_CRSN_DISC                 =  00000005
LAT_C_CRSN_DONE                 =  00000001
LAT_C_CRSN_IVMCI                =  00000004
LAT_C_CRSN_IVMSG                =  00000002
LAT_C_CRSN_IVSCI                =  00000003
LAT_C_CRSN_PROGRESS             =  00000006
LAT_C_CRSN_RESOURCE             =  00000009
LAT_C_CRSN_RETRANS              =  00000008
LAT_C_CRSN_TIMRANG              =  0000000A
LAT_C_CRSN_TMO                  =  00000007
LAT_C_RSN_XXX                   =  00000000
LCB_B_INV_MSG                   =  0000000E
LCB_B_INV_SLOT                  =  0000000F
LCB_C_LENGTH                    =  00000010
LCB_L_MSG_RCV                   =  00000004
LCB_L_MSG_REXMT                 =  00000008
LCB_L_MSG_XMT                   =  00000000
LCB_W_SEQ_ERR                   =  0000000C
LOAD_CREDITS                      00000F93  R      03
LOOP_ABORT                        00000F45  R      03
LOOP_EXIT                         0000123E  R      03
LT$ABORT                          00000C6A  R      03
LT$ALC_UCB                        00001547  R      03
LT$ASYNCERR                       00000964  R      03
LT$CIRCDEAD                       0000172A  R      03
LT$CREATECSB                      0000126D  R      03
LT$CREATEUCB                      0000150A  R      03
LT$CTRL_INIT                      00000658  R      03
LT$DDT                            00000000  RG     03
LT$DEALOCSB                       00001420  R      03
LT$DEALPOOL                       00001BD9  R      03
LT$DESTROYUCB                     0000151F  R      03
LT$DISCONNECT                     000016A7  R      03
LT$DPT                            00000000  RG     02
LT$DROPCXB                        00001BD8  R      03
LT$END                            00001BDF  R      03
LT$FFIPOSTDONE                    00001BCF  R      03
LT$FFI_RCV_MSG                    00000D1C  R      03
LT$FLOW_CHANGE                    00000C78  R      03
LT$FLUSH_DATA                     00000C59  R      03
LT$FORCE_HALT                     00000E4C  R      03
LT$GETFFI                         000007E0  R      03
LT$HANGUP_UCB                     00001690  R      03
LT$HANGUP_UCB_NOW                 00001683  R      03
LT$HISTORY                        00000727  R      03
LT$KILL_UCB                       000016D6  R      03
LT$NEW_CIRCUIT                    000012D0  R      03
LT$NULL                           000004F8  R      03
LT$RCV_HALT_MSG                   00000E62  R      03
LT$RCV_RUN_MSG                    00000E75  R      03
LT$RCV_START_MSG                  00000E68  R      03
LT$REPLY_ALT                      0000125A  R      03
LT$REPLY_DONE                     00001260  R      03
LT$REPLY_MSG                      00001256  R      03
LT$REPLY_REXMIT                   00001265  R      03
LT$REXMIT                         00001AD4  R      03
LT$SETENTRY                       00000965  R      03
LT$SET_TIMER                      00001818  R      03
LT$SHUTENTRY                      00000835  R      03
LT$SHUT_DONE                      00000953  RG     03
```

```
LT$SIDELINEUCB                  00001709 R      03
LT$STARTIO                      00000ADE R      03
LT$STATE_ENTER_RUN              00000EA6 R      03
LT$STATE_HALT                   00000E5C R      03
LT$STATE_RUN                    00000EAE R      03
LT$STATE_START                  00000E82 R      03
LT$STOPCIRC                     0000176B R      03
LT$STRENTRY                     0000075D R      03
LT$TICK                         0000188C R      03
LT$UCB_INIT                     00000A9B R      03
LT$UNIT_INIT                    00000A2F R      03
LT$VEC                          000004BC R      03
LT$VECEND                       000004F4 R      03
LT$VECTOR                       000004BC R      03
LT$XMIT                         00001A6F R      03
LT$XMITONE                      00001AA0 R      03
LT$XMTGO                        00001AED R      03
LT$XMT_FFIDONE                  00001B54 R      03
LT$XOFF                         00000CCE R      03
LT$XON                          00000C7E R      03
LT_HISTORY                    = 00000001
LT_RSN_BADCREDITS               000005E9 R      03
LT_RSN_CSBINVALID               00000561 R      03
LT_RSN_CSBRANGE                 00000541 R      03
LT_RSN_CSBSTALE                 0000057C R      03
LT_RSN_CSBZERO                  00000529 R      03
LT_RSN_HALT                     00000598 R      03
LT_RSN_INVALIDLOCID             000005B3 R      03
LT_RSN_INVALIDREMID             000005CE R      03
LT_RSN_REPCREATE                0000060C R      03
LT_RSN_REPDISC                  00000630 R      03
LT_RSN_START                    000004F9 R      03
MAX_PR_SID                    = 00000010
MMG$GL_MAXPFN                   ********    X   03
MPS$GL_NULLCPU                  ********    X   03
MTX$W_OWNCNT                  = 00000000
MTYP_C_CONF                   = 0000000A  G
MTYP_C_HALT                   = 00000002  G
MTYP_C_RUN                    = 00000000  G
MTYP_C_START                  = 00000001  G
NEXT_SLOT                       00001007 R      03
NEXT_SLOT0                      00000F90 R      03
ONE                             000019F0 R      03
ONE_LESS                        0000104C R      03
ORB$B_FLAGS                   = 00000008
ORB$L_OWNER                   = 00000000
ORB$M_PROT_16                 = 00000001
ORB$W_PROT                    = 00000018
PCB$L_PHD                     = 0000006C
PHD$L_CPUTIM                  = 00000038
PORT_ABORT                    = 00000020
PORT_DISCONNECT               = 00000004
PORT_LENGTH                   = 00000038
PORT_SET_LINE                 = 00000008
PORT_STARTIO                  = 000C0000
PORT_XOFF                     = 00000014
PORT_XON                      = 00000010
PR$_IPL                       = 00000012
PR$_SID_TYP730                = 00000003
PR$_SID_TYP750                = 00000002
PR$_SID_TYP780                = 00000001
PR$_SID_TYP785                = 00000009
PR$_SID_TYP790                = 00000004
PR$_SID_TYPMAX                = 00000008
PR$_SID_TYPUV1                = 00000007
PR$_SID_TYPUV2                = 00000008
RCV_B_ACK                       00000007  G
RCV_B_DSTIDN                    00000002  G
RCV_B_DSTIDS                    00000003  G
RCV_B_FLAG                      00000000  G
RCV_B_NUM_SLOTS                 00000001  G
RCV_B_SEQ                       00000006  G
RCV_B_SRCIDN                    00000004  G
RCV_B_SRCIDS                    00000005  G
RCV_T_DATA                      00000000  G
RCV_T_MDATA                     00000008  G
RCV_W_DSTID                     00000002  G
RCV_W_SRCID                     00000004  G
RSN$_ASTWAIT                  = 00000001
SCH$GL_CURPCB                   ********    X   03
SCH$GL_NULLPCB                  ********    X   03
SCH$RAVAIL                      ********    X   03
SCH$RWAIT                       ********    X   03
SEND_STOP_SLOT                  00001020 R      03
SHUT_TIMER                      00000936 R      03
SIZ...                        = 00000001
SLOT_LOOP_IN                    00000ED9 R      03
SLOT_LOOP_OUT                   00001044 R      03
SLOT_SIZE                     = 00000020
SLT_ATT_M_ABORT               = 00000020  G
SLT_ATT_S_ABORT               = 00000001  G
SLT_ATT_V_ABORT               = 00000005  G
SLT_B_ATT_CONTROL               00000004  G
SLT_B_AT_SLTSIZ                 00000005  G
SLT_B_COUNT                     00000002  G
SLT_B_CRED                      00000003  G
SLT_B_DB_CONTROL                00000004  G
SLT_B_DB_IFOF                   00000007  G
SLT_B_DB_IFON                   00000008  G
SLT_B_DB_OFOF                   00000005  G
SLT_B_DB_OFON                   00000006  G
SLT_B_DSTID                     00000000  G
SLT_B_DST_NAMLEN                00000007  G
SLT_B_DT_SLTSIZ                 00000006  G
SLT_B_PARAMS                    00000029  G
SLT_B_REAS                      00000003  G
SLT_B_REJ_RLEN                  00000004  G
SLT_B_SRCID                     00000001  G
SLT_B_SRC_NAME                  00000019  G
SLT_B_SRC_NAMLEN                00000018  G
SLT_B_STP_RLEN                  00000004  G
SLT_B_SVCC                      00000004  G
SLT_B_TYPE                      00000003  G
SLT_C_ATT_SLOT                = 0000000B  G
```

```
SLT_C_DA_SLOT                   = 00000000  G
SLT_C_DB_SLOT                   = 0000000A  G
SLT_C_INVSLOT                   = 00000003  G
SLT_C_INVSVC                    = 00000004  G
SLT_C_REJ_SLOT                  = 0000000C  G
SLT_C_RESOURCE                  = 00000005  G
SLT_C_START_FLAG                = 00000001  G
SLT_C_STP_SLOT                  = 0000000D  G
SLT_C_STR_SLOT                  = 00000009  G
SLT_C_SVC_INTT                  = 00000001  G
SLT_C_SYS                       = 00000002  G
SLT_C_USD                       = 00000001  G
SLT_DB_M_BREAK                  = 00000010  G
SLT_DB_M_IFDIS                  = 00000002  G
SLT_DB_M_IFENA                  = 00000001  G
SLT_DB_M_OFDIS                  = 00000008  G
SLT_DB_M_OFENA                  = 00000004  G
SLT_DB_S_BREAK                  = 00000001  G
SLT_DB_S_IFDIS                  = 00000001  G
SLT_DB_S_IFENA                  = 00000001  G
SLT_DB_S_OFDIS                  = 00000001  G
SLT_DB_S_OFENA                  = 00000001  G
SLT_DB_V_BREAK                  = 00000004  G
SLT_DB_V_IFDIS                  = 00000001  G
SLT_DB_V_IFENA                  = 00000000  G
SLT_DB_V_OFDIS                  = 00000003  G
SLT_DB_V_OFENA                  = 00000002  G
SLT_FLAG_M_BIND                 = 00000004  G
SLT_FLAG_M_LOGIN                = 00000002  G
SLT_FLAG_M_REMOTE               = 00000001  G
SLT_FLAG_S_BIND                 = 00000001  G
SLT_FLAG_S_LOGIN                = 00000001  G
SLT_FLAG_S_REMOTE               = 00000001  G
SLT_FLAG_V_BIND                 = 00000002  G
SLT_FLAG_V_LOGIN                = 00000001  G
SLT_FLAG_V_REMOTE               = 00000000  G
SLT_S_ATT_LEN                   = 00000001  G
SLT_S_ATT_SLOT                  = 00000005  G
SLT_S_DB_LEN                    = 00000005  G
SLT_S_DB_SLOT                   = 00000009  G
SLT_S_REJ_LEN                   = 00000000  G
SLT_S_REJ_SLOT                  = 00000004  G
SLT_S_START_SLOT                = 00000025  G
SLT_S_STP_LEN                   = 00000001  G
SLT_S_STP_SLOT                  = 00000005  G
SLT_T_DATA                      = 00000004  G
SLT_T_DST_NAME                  = 00000008  G
SLT_T_REJ_REAS                  = 00000004  G
SLT_T_START                     = 00000000  G
SLT_T_STP_REAS                  = 00000005  G
SS$_INSFMEM                     = ********    X  03
SS$_NORMAL                      = ********    X  03
STOP_B_RCODE                    = 00000000  G
STOP_B_RLEN                     = 00000001  G
STOP_TQE                        = 0000187C  R    03
STOP_T_REAS                     = 00000002  G
STRT_B_CIR_TIMR                 = 00000006  G
STRT_B_DL_BFRS                  = 00000005  G
STRT_B_ID_LEN                   = 0000002E  G
STRT_B_KPA_TIMR                 = 00000007  G
STRT_B_LOC_LEN                  = 0000006F  G
STRT_B_MAXSLOTS                 = 00000004  G
STRT_B_NODE_LEN                 = 0000000C  G
STRT_B_PARAM                    = 000000B0  G
STRT_B_PECO                     = 00000003  G
STRT_B_PVER                     = 00000002  G
STRT_B_SYS_LEN                  = 0000001D  G
STRT_B_VAR                      = 0000000C  G
STRT_C_LENGTH                   = 0000000C  G
STRT_C_VAR_LEN                  = 000000A5  G
STRT_T_ID                       = 0000002F  G
STRT_T_LOC                      = 00000070  G
STRT_T_NODE                     = 0000000D  G
STRT_T_SYS                      = 0000001E  G
STRT_W_FAC_NUM                  = 00000008  G
STRT_W_MSGSIZ                   = 00000000  G
STRT_W_PROD_TYP                 = 0000000A  G
SYS$GW_IJOBCNT                  = ********    X  03
SYS$GW_IJOBLIM                  = ********    X  03
THREE                           = 000019EC  R    03
TQE$B_RQTYPE                    = 0000000B
TQE$B_TYPE                      = 0000000A
TQE$C_LENGTH                    = 00000030
TQE$C_SSREPT                    = 00000005
TQE$C_SSSNGL                    = 00000001
TQE$L_FPC                       = 0000000C
TQE$M_REPEAT                    = 00000004
TQE$Q_DELTA                     = 00000020
TQE$W_SIZE                      = 00000008
TT$V_HOSTSYNC                   = 00000004
TT$V_PASSALL                    = 00000000
TT$V_REMOTE                     = 0000000D
TT$V_TTSYNC                     = 00000005
TT$_UNKNOWN                     = 00000000
TT2$V_AUTOBAUD                  = 00000001
TT2$V_HANGUP                    = 00000002
TTY$GB_DEFSPEED                 = ********    X  02
TTY$GB_PARITY                   = ********    X  02
TTY$GB_RSPEED                   = ********    X  02
TTY$GL_DEFCHAR                  = ********    X  02
TTY$GL_DEFCHAR2                 = ********    X  02
TTY$GL_DELTA                    = ********    X  03
TTY$GL_DPT                      = ********    X  03
TTY$GL_OWNUIC                   = ********    X  02
TTY$GW_DEFBUF                   = ********    X  02
TTY$GW_PROT                     = ********    X  02
TTY$V_PC_NOTIME                 = 00000000
TTY_C_BELL                      = 00000007
TWO                             = 000019EE  R    03
UCB$B_DEVCLASS                  = 00000040
UCB$B_DEVTYPE                   = 00000041
UCB$B_DIPL                      = 0000005E
UCB$B_ERTCNT                    = 00000080
UCB$B_FIPL                      = 0000000B
```

```
UCB$B_LT_CBUF                 0000014C  G
UCB$B_LT_CURC                 00000148  G
UCB$B_LT_DATAW                00000142  G
UCB$B_LT_EBUF                 0000019C  G
UCB$B_LT_LATSTS               00000143  G
UCB$B_LT_LOCID                00000141  G
UCB$B_LT_MAXC                 0000014A  G
UCB$B_LT_RCRED                00000148  G
UCB$B_LT_REASON               00000145  G
UCB$B_LT_REMID                00000140  G
UCB$B_LT_SLOTSZ               00000149  G
UCB$B_LT_STATE                00000144  G
UCB$B_LT_TCRED                00000146  G
UCB$B_LT_XCRED                00000147  G
UCB$B_TT_DEPARI             = 000000EC
UCB$B_TT_DETYPE             = 000000F0
UCB$B_TT_OUTYPE             = 00000108
UCB$C_LT_LENGTH               0000019C  G
UCB$C_LT_STATE_KILL         = 00000004  G
UCB$C_LT_STATE_RUN          = 00000002  G
UCB$C_LT_STATE_START        = 00000001  G
UCB$C_LT_STATE_STOP         = 00000003  G
UCB$C_TT_LENGTH             = 00000134
UCB$L_CRB                   = 00000024
UCB$L_DDB                   = 00000028
UCB$L_DDT                   = 00000088
UCB$L_DEVCHAR               = 00000038
UCB$L_DEVCHAR2              = 0000003C
UCB$L_DEVDEPEND             = 00000044
UCB$L_LT_CSB                  00000134  G
UCB$L_LT_DEADLINK             00000138  G
UCB$L_LT_INPBUF               0000013C  G
UCB$L_STS                   = 00000064
UCB$L_TT_CLASS              = 00000114
UCB$L_TT_DECHA1             = 000000C8
UCB$L_TT_DECHAR             = 000000C4
UCB$L_TT_DEVDP1             = 00000048
UCB$L_TT_GETNXT             = 0000010C
UCB$L_TT_LOGUCB             = 000000C0
UCB$L_TT_OUTADR             = 0000011C
UCB$L_TT_PORT               = 00000118
UCB$L_TT_PUTNXT             = 00000110
UCB$L_TT_RTIMOU             = 000000B4
UCB$L_TT_WBLINK             = 000000D0
UCB$L_TT_WFLINK             = 000000CC
UCB$M_INT                   = 00000002
UCB$M_LT_ABORT              = 00000004  G
UCB$M_LT_BOUND              = 00000008  G
UCB$M_LT_DATA               = 00000001  G
UCB$M_LT_DEAD               = 00000002  G
UCB$M_LT_FLOW               = 00000002  G
UCB$M_LT_HANGUP             = 00000001  G
UCB$M_LT_INPUT              = 00000010  G
UCB$M_LT_OVFLOW             = 00000004  G
UCB$M_ONLINE                = 00000010
UCB$S_LT_ABORT              = 00000001  G
UCB$S_LT_BOUND              = 00000001  G
UCB$S_LT_DATA               = 00000001  G
UCB$S_LT_DEAD               = 00000001  G
UCB$S_LT_FLOW               = 00000001  G
UCB$S_LT_HANGUP             = 00000001  G
UCB$S_LT_INPUT              = 00000001  G
UCB$S_LT_OVFLOW             = 00000001  G
UCB$V_DELETEUCB             = 00000010
UCB$V_LT_ABORT              = 00000002  G
UCB$V_LT_BOUND              = 00000003  G
UCB$V_LT_DATA               = 00000000  G
UCB$V_LT_DEAD               = 00000001  G
UCB$V_LT_FLOW               = 00000001  G
UCB$V_LT_HANGUP             = 00000000  G
UCB$V_LT_INPUT              = 00000004  G
UCB$V_LT_OVFLOW             = 00000002  G
UCB$V_POWER                 = 00000005
UCB$V_TEMPLATE              = 0000000D
UCB$W_DEVBUFSIZ             = 00000042
UCB$W_DEVSTS                = 00000068
UCB$W_ERRCNT                = 00000082
UCB$W_MB_SEED               = 00000000
UCB$W_REFC                  = 0000005C
UCB$W_TT_DESIZE             = 000000F1
UCB$W_TT_DESPEE             = 000000E8
UCB$W_TT_OUTLEN             = 00000120
UCB$W_TT_PRTCTL             = 00000122
UCB$W_TT_SPEED              = 000000F4
UCB$W_UNIT                  = 00000054
USE_CREDIT                    000011D6  R      03
USE_CREDIT1                   000011DC  R      03
VEC$L_INITIAL               = 0000000C
VEC$L_START                 = 0000001C
VEC$L_UNITINIT              = 00000018
XMT_B_ACK                     0000004F  G
XMT_B_DSTIDN                  0000004A  G
XMT_B_DSTIDS                  0000004B  G
XMT_B_FLAG                    00000048  G
XMT_B_MC_CHG_FLAG             0000004F  G
XMT_B_MC_CIR_TIMER            00000049  G
XMT_B_MC_CUR_ECO              0000004D  G
XMT_B_MC_CUR_VER              0000004C  G
XMT_B_MC_GRP_LEN              00000054  G
XMT_B_MC_HI_VER               0000004A  G
XMT_B_MC_ID_LEN               00000099  G
XMT_B_MC_INCARN               0000004E  G
XMT_B_MC_LO_VER               00000048  G
XMT_B_MC_MULTIMR              00000052  G
XMT_B_MC_NAME_LEN             00000088  G
XMT_B_MC_NODE_LEN             00000075  G
XMT_B_MC_NUM_SVCS             00000086  G
XMT_B_MC_RATE                 00000087  G
XMT_B_MC_SET                  00000052  G
XMT_B_MC_STATUS               00000053  G
XMT_B_MC_SVC                  0000027A  G
XMT_B_MC_SVC_LEN              00000279  G
XMT_B_NUM_SLOTS               00000049  G
XMT_B_SEQ                     0000004E  G
```

```
XMT_B_SRCIDN                    0000004C  G
XMT_B_SRCIDS                    0000004D  G
XMT_CHFLG_M_ALL               = 000000FF  G
XMT_CHFLG_M_CLASS             = 00000020  G
XMT_CHFLG_M_GROUP             = 00000001  G
XMT_CHFLG_M_NDESC             = 00000002  G
XMT_CHFLG_M_OTHER             = 00000080  G
XMT_CHFLG_M_RATE              = 00000008  G
XMT_CHFLG_M_SDESC             = 00000010  G
XMT_CHFLG_M_SVCNAM            = 00000004  G
XMT_CHFLG_S_CLASS             = 00000001  G
XMT_CHFLG_S_GROUP             = 00000001  G
XMT_CHFLG_S_NDESC             = 00000001  G
XMT_CHFLG_S_OTHER             = 00000001  G
XMT_CHFLG_S_RATE              = 00000001  G
XMT_CHFLG_S_SDESC             = 00000001  G
XMT_CHFLG_S_SVCNAM            = 00000001  G
XMT_CHFLG_V_CLASS             = 00000005  G
XMT_CHFLG_V_GROUP             = 00000000  G
XMT_CHFLG_V_NDESC             = 00000001  G
XMT_CHFLG_V_OTHER             = 00000007  G
XMT_CHFLG_V_RATE              = 00000003  G
XMT_CHFLG_V_SDESC             = 00000004  G
XMT_CHFLG_V_SVCNAM            = 00000002  G
XMT_C_HDRLEN                  = 00000008  G
XMT_C_MAXDATA                 = 000005D4  G
XMT_C_MAXLEN                  = 00000624  G
XMT_C_MC_LENGTH               = 0000000A  G
XMT_C_MC_SVC_SIZE             = 00000053  G
XMT_C_MC_TOTAL                = 0000023A  G
XMT_G_DST                       0000003A  G
XMT_T_DATA                      00000048  G
XMT_T_MC_GROUP                  00000055  G
XMT_T_MC_ID                     0000009A  G
XMT_T_MC_MORE_SVC               000000DA  G
XMT_T_MC_NAME                   00000089  G
XMT_T_MC_NODE                   00000076  G
XMT_T_MDATA                     00000050  G
XMT_W_DSTID                     0000004A  G
XMT_W_MC_MSG_SIZ                00000050  G
XMT_W_SRCID                     0000004C  G
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000282 (642.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$105_PROLOGUE	000000DE (222.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	00001BDF (7135.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	96	00:00:00.19	00:00:02.34
Command processing	137	00:00:00.50	00:00:03.20
Pass 1	1002	00:00:31.72	00:01:40.19
Symbol table sort	2	00:00:03.08	00:00:07.20
Pass 2	428	00:00:09.41	00:00:29.42
Symbol table output	2	00:00:00.39	00:00:00.75
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1671	00:00:45.31	00:02:23.10

The working set limit was 3000 pages.
233724 bytes (457 pages) of virtual memory were used to buffer the intermediate code.
There were 160 pages of symbol table space allocated to hold 2613 non-local and 352 local symbols.
4681 source lines were read in Pass 1, producing 49 object records in Pass 2.
105 pages of virtual memory were used to define 97 macros.

Macro library statistics

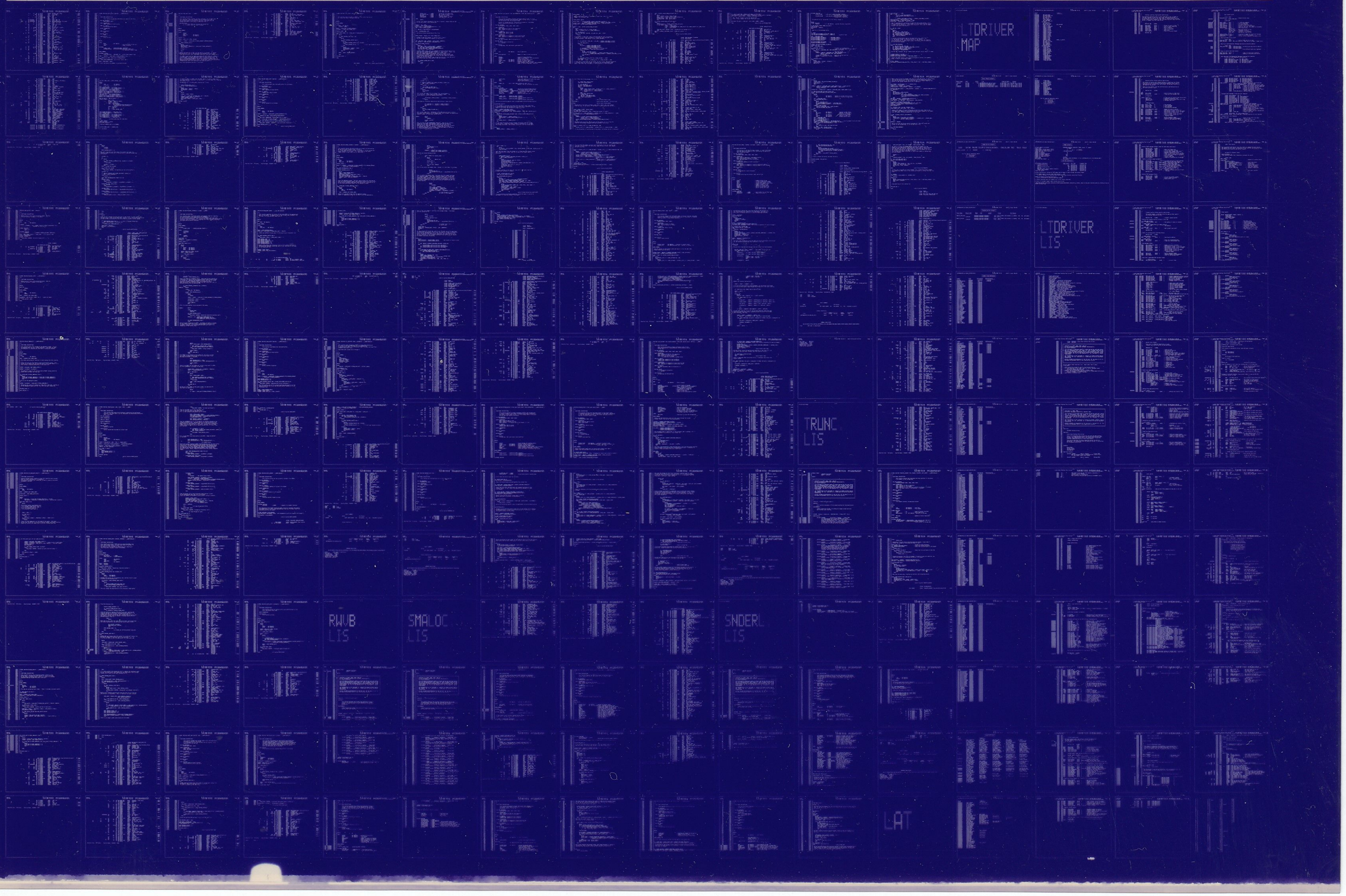
Macro library name	Macros defined
_255DUA18:[LAT.OBJ]LAT.MLB;2	1
_255DUA18:[SYS.OBJ]LIB.MLB;1	39
_255DUA18:[SYSLIB]STARLET.MLB;3	10
TOTALS (all libraries)	50

2819 GETS were required to define 50 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:LTDRIVER/OBJ=OBJ$:LTDRIVER MSRC$:LTDRIVER/UPDATE=(BUG$:LTDRIVER)+EXECML$/LIB+LIB$:LAT/LIB
```

0444 AH-EF71A-SE
VAX/VMS V4.1 SRC LST MCRF UPD



0445 AH-EF71A-SE
VAX/VMS V4.1 SRC LST MCRF UPD

SMGDISLIN
LIS

SMGRTL

SMGSHR
MAP