DR11-L/M

EXERCISER

MD-11-DZDRH-A

EP-DZDRH-A-DL-A

FICHE 1 OF 1

NOV 1976

digital

MADE IN USA

IDENTIFICATION
****

Product Code: MAINDEC-11-DZDRH-A-D
Product Name: DR11-L, DR11-M Exerciser
Date Created: Sept. 1, 1975
Maintainer: Diagnostic Engineering
Author: Edward C. Badger

TABLE OF CONTENTS
****

1.0 ABSTRACT
****

This program allows the user to checkout or debug the DR11-L or DR11-M module (UNIBUS 2 word input interface or UNIBUS 2 word output interface). To run the logic test, the user must have a maintenance loop-back cable. Logic tests are limited and cannot check the input or output data paths. To check input or output data paths, some means of generating or detecting data changes is needed.

2.0 REQUIREMENTS
****

2.1 Equipment

1. PDP-11/Family computer with 4K of core or more and I/O terminal.
2. DR11L or DR11M (To be tested).
3. For logic test, maintenance loop-back cable. The maintenance loop-back cable is made up of a Berg Connector with 2 wires. One wire connects pin "V" with pin "VV". The other wire connects pin "X" with pin "TT".
4. For auto-slave, a master PDP-11 with DR11-M and this program.
5. For auto-master, a slave CPU (PDP-8 or PDP-11) with input module (DR11-L).
6. For auto-master/slave, a DR11L cabled to a DR11M.
7. For all other tests, some means of generating or detecting data changes.

2.2 Storage

This program occupies core locations 000000-017474.

3.0 LOADING PROCEDURE
****

3.1 Method

Procedure for normal binary tapes should be followed:

1. Absolute Loader must be in memory.
2. Place binary tape in reader.
3. Load address *7500 (*Determined by location of loader).
4. Press "START" (Program will load).

4.0 STARTING PROCEDURE
****

4.1 Control Switch Settings

Starting at SA200 or 210 set all switches as desired. See Section 5.1.

4.2 Starting Addresses

1. Load and start at location 200 for initial questions about SPC modules.
2. Load and start at location 210 to avoid questions if you wish to run the program with the same parameters as set by a start at 200.

NOTE

After loading program into core you *must* initially start at location 200.

4.3 Program AND/OR Operator Action

1. Load program into core.
2. Set switch register to starting address.
3. Load address.
4. Set switches to desired settings - See Section 5.1.
5. Press start.
6. The program will type first question.

4.4 Program Questions

4.4.1 TERMINAL CHANGE REQUIRED (Y OR N)?

Respond by typing a Y or (no carriage return required) to represent YES or NO. If the program is started on the CPU's console TTY and you wish to change to a different TTY, type "Y", otherwise type "N".

4.4.2 KEYBOARD CSR ADDR. OF NEW TERMINAL?

Question asked only if YES was the response to 4.4.1. Respond by typing the keyboard CSR address of the new terminal.

4.4.3 ITS VECTOR ADDR.?

Question asked only if YES was the response to 4.4.1. Respond by typing the vector address of the new terminal's keyboard.

4.4.4 PRINTER CSR ADDR. OF NEW TERMINAL?

Question asked only if YES was the response to 4.4.1. Respond by typing the printer CSR address of the new terminal.

4.4.5 CSR ADDR. OF INPUT MODULE?

Respond by typing the CSR address of the word on the input module you wish to exercise. If you do not wish to exercise an input module, type only a carriage return.

4.4.6 CSR ADDR. OF OUTPUT MODULE?

Respond by typing the CSR address of the word on the output module you wish to exercise. If you do not wish to exercise an output module, type only a carriage return.

4.4.7 MODE?

Respond to this question by typing a character from "A" to "M" followed by carriage return. At any time (except logic tests) you may return to this question by typing "^C" (control and the letter "C" typed together).

MODE	MODULE AFFECTED	FUNCTION
A	output	Continuous output. Load data to the output module without checking for any flag.
B	output	Flag output. Check for output modules's "Ready" flag (bit7) and loads DBR with data.
C	output	Interrupt output. Program sets output module's interrupt enable. On interrupt it loads output modules DBR with data.
D	output	Delayed output. The program clears output modules CSR bit0; loads DBR with data; and then sets CSR bit0.
E	output	Start Auto Mode Master. This program may be used to generate data a a master to send to a slave CPU running slave mode. A table of data is outputed in modes A-D automatically. Mode "K" of program MD-11-DZDRH (forPDP-11's) or MD-08-DHDRH (for PDP-8's) must have been selected. Output word must be cabled to input word of DR11-M or DR8-ED. "Start Mode" is typed at the beginning of this mode. The program stalls waiting to link up with the input module its connected to. After link-up, "Immediate Send Mode" is typed. In this mode data from the table is sent directly out as soon as the input module

is ready to receive it. If the input module detects an error in the data it receives, it can request a re-transmit of the previous data by setting the stat out bit thus setting the input module's stat in bit. If this occurs "Resending data xxxxxx" will be typed. After this mode, "Burst Send Mode" is typed. In this mode the output module is allowed to interrupt for table data. "END PASS" is typed after completion.

Warning: Before starting the input or output module into "Start Mode", both CPU's should be running and waiting at "4.4.2 Mode?". At no time should the program be allowed to enter "Start Mode" and the other CPU initialized.

This mode was not designed to diagnose this module. Only data errors are checked. If any of the control signals fail the program could "hang" waiting for the signal. Maximum time between typeouts should not exceed 3 minutes - if it does, halt the CPU and determine the PC the program is at. Go to the listing for more information. Prints mentioned in the listing can be coordinated against the flow chart.

F	output	Output Module Logic Test. The program preforms a brief logic check of the outputs module's CSR and interrupt capacities. A Maintance loop-back must be installed in the word under test.
G	input	Continuous input. The program continously reads the input module's DBR.
H	input	Flag input. The program waits for the "Ready" flag (CSR bit7) and then does a DBR read when bit7=1.
I	input	Interrupt input. The program sets interrupt enable and waits until the input module interrupts to read the modules DBR.
J	input	Delayed input. The program clears CSR bit0; reads the DBR; and then sets CSR bit0, thus delaying Data accept from returning.
K	input	Start Auto Slave The program may be used to receive data as a slave to receive data incoming from master CPU running in Master Mode (see Mode E), as data is received it is compared against a table of data. Modes G-J are automatically executed. If data received does not compare to the table, an error is reported. Mode "E" of MD-11-DZDRH must be selected in Master CPU. Input word must be cabled to the output word of a DR11-L. "Start Mode" is typed at the beginning of this mode. The program stalls waiting to link up with the output module its connected to. After link-up, "Immediate Send Mode" is typed. Each time data is ready, the data is read and compared against table data. "Burst Send Mode" is next. Here the input module is allowed to interrupt each time it receives data. "End Pass" is typed at completion. Warning: Before starting the input or output module into "Start Mode" both CPU's should be

running and waiting at "4.4.2 Mode?". At no time should the program be allowed to enter "Start Mode" and the other CPU be initialized.

This mode was not designed to diagnose this module. Only data errors are checked. If any of the control signals fail the program could "hang" waiting for the signal. Maximum time between typeouts should not exceed 3 minutes - if it does, halt the CPU and determine the PC the program is at. Go to the listing for more information. Points mentioned in the listing can be coordinated against the flow chart.

L	input	Input Module Logic Test. The program preforms a brief logic check of the input modules CSR and interrupt capacities. A Maintance loop back connector must be installed in the word under test.
M	input-output	Start Auto Master-Slave. Preforms Modes E and K on an input and output mode located in the same CPU. The input word on the DR11-M must be cabled to the output word on the DR11-L.

4.4.8 SOURCE OF DATA?

This question is typed when an output Mode A-D is selected. What the program wants to know is where do you want to get the data that you want to output. Respond by typing a character from A-D followed by a carriage return.

A	Data from the Switch Register
B	Data from the TTY. In this case, any future octal digits typed on the Keyboard will be assembled as data and each time a carriage return is typed the new data typed will be sent out.
C	Program generated pattern. The program will automatically generate the data to be outputed in one of 4 ways specified by the operator by question 4.4.10.
D	Data is outputed from a table in core. Data to be outputed can be found by finding the address of TABLE and looking in the listing.

4.4.9 STARTING PATTERN? (If Source of Data "C" Selected)

Respond by typing the octal representation of the data you wish to start outputing. This data will be modified according to the pattern modifier you select in 4.4.10.

4.4.10 PATTERN MODIFIER? (If source of Data "C" Selected)

Respond by typing a character from "A" to "D" to represent how you want the data that is being outputed to be modified.

MODIFIER	FUNCTION
A	INCREMENT DATA
B	DECREMENT DATA
C	FLOAT ZERO-COMPLEMENT
D	COMPLEMENT

4.4.11 METHOD OF REPORTING DATA?

Respond by typing a character from "A" to "C" to represent the way you wish the data being received by the input module to be reported.

MEANS	METHOD
A	in data lights
B	on TTY-reported each time a change in input data is detected.
C	Error check. An input module is cabled to an output module/data beginning sent to the output module is compared to the data received by the input module-if data is different an error is reported. After each 100 transfers are made, "END PASS" is reported.
D	Error Check. An input module is cabled to an

output module in another
CPU. This method only
works when the "Source
of Data" for the output
module is that of the
table "D".
On the output module
side, you would select
"Mode" "B" or "C";
"Source of Data" "D".
On the input module
side, you would select
"Mode" "H" or "I"
ERRORS detected as
in modes "E" and "K".

5.0 OPERATING PROCEDURE LOGIC TEST
****

5.1 Definition - Switch Register

The program must, at various points in execution, determine what the operator desires to do at these points (such as "halt on error"). Standard procedure was to get this information from the hardware switch register. With the addition of CPU's to this family of computers that have no switch register, a hardware switch register can no longer be assumed. This program uses a software switch register; that is, a location in memory "SWREQ" as the source of "switch register" data.

5.1.1 Loading the Software Switch Register

The software switch register can be altered in two ways:

5.1.2 From the Hardware Switch Register

This way over-rides the other ways. Data loaded into the hardware switch register will be transferred to the software switch register directly each time "switch register" data is needed.

5.1.3 From the User's Terminal

A "↑S" (control and letter "S" keys typed simultaneously) will cause the current software switch register to be typed out and allow the user to type in the new value of the switch register.

If a hardware switch register exists, and the user desires to use the software switch register, the user should leave the hardware switch register clear (all switches down).

NOTE 1

When using a pure software switch register "halt on error" is re-defined to "hold on error". If the CPU were allowed to halt, then core data coult not be examined and the switch register data could not be altered. "Hold on error" sets in a loop testing software SWR bit 15, allowing TTY interrupts.

NOTE 2

Some means of examining CPU memory must be incorporated into the keyboard handler must be made so that the operator on a switch register - less CPU can examine core locations while in a "hold on error". The program uses a "↑E xxxxxx" where xxxxxx is the location to be examined.

5.1.4 Switch Register Function

SW15=1 or up Halt on error.
SW14=1 or up Loop on test.
SW13=1 or up Inhibit printout of error.
SW11=1 or up Inhibit subtest iterations.

5.2 Scope Loops

If an error occurs and the user wishes to scope the error, he (or she) should set SW15=1 to halt on error, then when the program halts on error, SW15=0, set SW14=1 to (loop on current test), SW13=1 (to inhibit error printout) and press continue on the CPU's CONSOLE.

5.3 Program AND/OR Operator Action

The first pass through the program will be made with iterations inhibited. Successive passes will enable iterations if SW11=0. "END PASS" is printed at the end of a pass. The vector and priority of each input or output under test will be typed when program is initially run or if the vector or priority is changed during a pass.

6.0 ERRORS

6.1 Error Typeout

Print out varies with the error detected. The error PC typed out is the actual location of the error call. Error typeouts are preceeded with THE ADDRESS of the error, if information is desired, reference the listing in respect to the PC (ADDRESS) typed out.

6.2 Non-Standard Errors

"TRAPPED TO LOCATION:XXXXXX FROM LOCATION:YYYYYY"

indicates:

1. An INPUT or OUTPUT module interrupted to a wrong vector.
2. Time-out or illegal instruction hareware trap. This is a fatal error that can not be continued past.

7.0 RESTRICTIONS

7.1 Starting Restriction

If the Vector address of any input or output is either 200 or 210-the program must be restricted at address 001000.

8.0 MISCELLANEOUS

8.1 Execution Time Logic Test

0,5 MIN. Iterations inhibited-no errors.
1,0 MIN. WITH ITERATIONS (FOR EACH CONNECTION)-NO ERRORS.

Execution times are approximate, as the various PDP-11 CPU's have varied instruction execution times.


```
                        064000                  $SWR=   064000
                                                .TITLE  MAINDEC-11-DZDRH-A
                                                ;*COPYRIGHT (C) 1975
                                                ;*DIGITAL EQUIPMENT CORP.
```

```
                                        ;*MAYNARD, MASS. 01754
                                        ;*
                                        ;*PROGRAM BY EDWARD C. BADGER
                                        ;*
                                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                        ;*PACKAGE (MAINDEC-11-DZQAC-C2), SEPT 14, 1976.
                                        ;*
            000001                      $TN=1

                                        .SBTTL  TRAP    CATCHER

                                                .=0
            000000                      ;*ALL UNUSED LOCATIONS 0 FROM 4 TO 776 CONTAIN .+4. IOTT (TRAP CALL)
                                        ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS TO WRONG VECTORS AND
                                        ;*ROUTE THEM TO ROUTINE "IOTRP" FOR TYPEOUT.
                                        ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS.

            000040                              .=40
000040  000042                                  .WORD   42
000042  000000                                  .WORD   0
            000174                              .=174
000174  000000                      SWREG:  0
000176  000000                      DISPRE: 0
            000200                              .=200
000200  000137  001306                          JMP     @#START         ;GO TO STARTING ADDRESS OF PROGRAM.

            000210                              .=210
000210  000137  003060                          JMP     @#RSTART

            001000                              .=1000
001000  000137  001306                          JMP     @#START


                                        .SBTTL  BASIC DEFINITIONS

                                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
            001100                      STACK=  1100
                                        .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                                        .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL

                                        ;*MISCELLANEOUS DEFINITIONS
            000011                      HT=     11              ;;CODE FOR HORIZONTAL TAB
            000012                      LF=     12              ;;CODE FOR LINE FEED
            000015                      CR=     15              ;;CODE FOR CARRIAGE RETURN
            000200                      CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
            177776                      PS=     177776          ;;PROCESSOR STATUS WORD
                                        .EQUIV  PS,PSW
            177774                      STKLMT= 177774          ;;STACK LIMIT REGISTER
            177772                      PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
            177570                      DSWR=   177570          ;;HARDWARE SWITCH REGISTER
            177570                      DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
```

```
                                ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                          R0=     %0              ;;GENERAL REGISTER
000001                          R1=     %1              ;;GENERAL REGISTER
000002                          R2=     %2              ;;GENERAL REGISTER
000003                          R3=     %3              ;;GENERAL REGISTER
000004                          R4=     %4              ;;GENERAL REGISTER
000005                          R5=     %5              ;;GENERAL REGISTER
000006                          R6=     %6              ;;GENERAL REGISTER
000007                          R7=     %7              ;;GENERAL REGISTER
000006                          SP=     %6              ;;STACK POINTER
000007                          PC=     %7              ;;PROGRAM COUNTER

                                ;*PRIORITY LEVEL DEFINITIONS
000000                          PR0=    0               ;;PRIORITY LEVEL 0
000040                          PR1=    40              ;;PRIORITY LEVEL 1
000100                          PR2=    100             ;;PRIORITY LEVEL 2
000140                          PR3=    140             ;;PRIORITY LEVEL 3
000200                          PR4=    200             ;;PRIORITY LEVEL 4
000240                          PR5=    240             ;;PRIORITY LEVEL 5
000300                          PR6=    300             ;;PRIORITY LEVEL 6
000340                          PR7=    340             ;;PRIORITY LEVEL 7

                                ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                          SW15=   100000
040000                          SW14=   40000
020000                          SW13=   20000
010000                          SW12=   10000
004000                          SW11=   4000
002000                          SW10=   2000
001000                          SW09=   1000
000400                          SW08=   400
000200                          SW07=   200
000100                          SW06=   100
000040                          SW05=   40
000020                          SW04=   20
000010                          SW03=   10
000004                          SW02=   4
000002                          SW01=   2
000001                          SW00=   1
                                .EQUIV  SW09,SW9
                                .EQUIV  SW08,SW8
                                .EQUIV  SW07,SW7
                                .EQUIV  SW06,SW6
                                .EQUIV  SW05,SW5
                                .EQUIV  SW04,SW4
                                .EQUIV  SW03,SW3
                                .EQUIV  SW02,SW2
                                .EQUIV  SW01,SW1
                                .EQUIV  SW00,SW0

                                ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                          BIT15=  100000
040000                          BIT14=  40000
020000                          BIT13=  20000
010000                          BIT12=  10000
```

```
                        004000          BIT11=  4000
                        002000          BIT10=  2000
                        001000          BIT09=  1000
                        000400          BIT08=  400
                        000200          BIT07=  200
                        000100          BIT06=  100
                        000040          BIT05=  40
                        000020          BIT04=  20
                        000010          BIT03=  10
                        000004          BIT02=  4
                        000002          BIT01=  2
                        000001          BIT00=  1
                                        .EQUIV  BIT09,BIT9
                                        .EQUIV  BIT08,BIT8
                                        .EQUIV  BIT07,BIT7
                                        .EQUIV  BIT06,BIT6
                                        .EQUIV  BIT05,BIT5
                                        .EQUIV  BIT04,BIT4
                                        .EQUIV  BIT03,BIT3
                                        .EQUIV  BIT02,BIT2
                                        .EQUIV  BIT01,BIT1
                                        .EQUIV  BIT00,BIT0

                                        ;*BASIC "CPU" TRAP VECTOR ADDRESSES
                        000004          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
                        000010          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
                        000014          TBITVEC=14              ;;"T" BIT
                        000014          TRTVEC= 14              ;;TRACE TRAP
                        000014          BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
                        000020          IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
                        000024          PWRVEC= 24              ;;POWER FAIL
                        000030          EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
                        000034          TRAPVEC=34              ;;"TRAP" TRAP
                        000060          TKVEC=  60              ;;TTY KEYBOARD VECTOR
                        000064          TPVEC=  64              ;;TTY PRINTER VECTOR
                        000240          PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR

                        022626          POPSP2= 22626
```

```
                                .SBTTL  COMMON TAGS

                                ;;**********************************************************************
                                ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                ;*USED IN THE PROGRAM.

                        001100                  .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000          $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
001102  000             $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103  000             $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114  000             $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115  001             $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000          $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000          $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                  .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                  .WORD   0
001134  000             $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135  000             $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                  .WORD   0
001140  177570          SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570          DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560          $TKS:   177560                  ;;TTY KBD STATUS
001146  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154  000             $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155  002             $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156  012             $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157  000             $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000          $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
                                                        ;;WHICH  ($REGO) WAS OBTAINED
001162  000000          $REGO:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001164  000000          $TMPO:  .WORD   0               ;;USER DEFINED
001166  000000          $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001170  077             $QUES:  .ASCII  /?/             ;;QUESTION MARK
001171  015             $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001172  000012          $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;**********************************************************************

001174  000000          XCSRA:  .WORD   0               ;ADDRESS OF OUTPUT MODULE'S CSR.

001176  000000          XDBRA:  .WORD   0               ;ADDRESS OF OUTPUT MODULE'S DBR.

001200  000000          XVT:    .WORD   0               ;VECTOR ADDR. OF OUTPUT MODULE.
001202  000000          XVT2:   .WORD   0               ;VECTOR ADDR.+2 FOR INTR. PSW.

001204  000000          RCSRA:  .WORD   0               ;ADDRESS OF INPUT MODULE'S CSR.
001206  000000          RDBRA:  .WORD   0               ;ADDRESS OF INPUT MODULES DBR.
```

```
                                ;
001210  000000          RVT:    .WORD   0       ;VECTOR ADDR. OF INPUT MODULE.
001212  000000          RVT2:   .WORD   0       ;VECTOR ADDR.+2 OF INPUT MODULE
                                ;
001214  000060          TKV:    .WORD   60      ;I/O TERMINAL VECTOR ADDR.
001216  000062          TKV2:   .WORD   62      ;I/O TERMINAL VECTOR ADDR.+2.
                                ;
001220  000000          MODE:   .WORD   0       ;MODE OF OPERATION OF PROGRAM
001222  000000          MODEI:  .WORD   0       ;MODE OF OPERATION OF INPUT ROUTINES.
001224  000000          PATFRO: .WORD   0       ;INDICATES SOURCE OF DATA
                                ;               ;VALUES 'A TO 'D
001226  000000          PATTM:  .WORD   0       ;INDICATES PATTERN MODIFIER
001230  000000          PATTRN: .WORD   0       ;ACTUAL PATTERN TO BE OUTPUTTED.
001232  000000          TEXTP:  .WORD   0       ;POINTS TO TABLE OF PATTERNS FOR MASTER.
001234  000000          TABLEP: .WORD   0       ;POINTS TO TABLE OF PATTERNS
001236  000000          TTYPAT: .WORD   0       ;PATTERN INPUTTED ON TTY.
001240  000000          PATOUT: .WORD   0       ;METHOD OF REPORTING DATA.
001242  000000          PATTIN: .WORD   0       ;DATA READ FROM INPUT MODULE.
001244  000000          PATOLD: .WORD   0       ;OLD PATTERN IF OUTPUTTING ON TTY.
                                ;
001246  000000          PTRM:   .WORD   0       ;USED BY MASTER-SLAVE AUTO TO SAVE MASTER'S PC.
001250  000000          PTRS:   .WORD   0       ;USED BY MASTER-SLAVE AUTO TO SAVE SLAVE'S PC.
001252  000000          SMSST:  .WORD   0       ;USED BY MASTER-SLAVE AUTO TO SAVE MASTER'S STATUS.
001254  000000          SSLST:  .WORD   0       ;USED BY MASTER-SLAVE AUTO TO SAVE SLAVE'S STATUS.
                                ;
001256  000000          XNO:    .WORD   0       ;MASTER'S ITERATION COUNT
001260  000000          SXNO:   .WORD   0       ;SLAVE'S ITERATION COUNT.
                                ;
001262  000000          STABLP: .WORD   0       ;POINTS TO SLAVE'S BSM STORAGE AREA.
001264  000000          XVTP:   .WORD   0       ;PRIORITY OF OUTPUT.
001266  000000          RVTP:   .WORD   0       ;PRIORITY OF INPUT.
                                ;
001270  000000          CHAR:   .WORD   0       ;CHARACTER INPUTTED FROM TTY.
001272  000001          ST200:  .WORD   1       ;INDICATES STARTED AT 200 IF 0.
001274  000000          $TEMP0: .WORD   0       ;TEMP STORAGE.
001276  000000          SAVPC:  .WORD   0       ;SAVES PC IN ERROR ROUTINE.
001300  177570          BSWR:   .WORD   DSWR    ;ADDR. OF REAL HARDWARE SWITCH REGISTER.
001302  000000          SWREQ:  .WORD   0       ;SOFTWARE SWITCH REGISTER.
001304  000000          ERTYCN: .WORD   0       ;#OF ERRORS TYPED OUT.
                                ;
```

```
                                        .SBTTL  ERROR POINTER TABLE

                                        ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                        ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                        ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                        ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                        ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

                                        ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                        ;*      DH              ;;POINTS TO THE DATA HEADER
                                        ;*      DT              ;;POINTS TO THE DATA
                                        ;*      DF              ;;POINTS TO THE DATA FORMAT


001306                                  $ERRTB:

                                                .SBTTL  PROGRAM START-UP

001306                                  START:
                                        .SBTTL  INITIALIZE THE COMMON TAGS
                                        ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
001306  012706  001100                          MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001312  005026                                  CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001314  022706  001140                          CMP     #SWR,R6 ;;DONE?
001320  001374                                  BNE     .-6             ;;LOOP BACK IF NO
001322  012706  001100                          MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                        ;;INITIALIZE A FEW VECTORS
001326  012737  015524  000020                  MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001334  012737  000340  000022                  MOV     #340,@#IOTVEC+2 ;;LEVEL 7
001342  012737  015706  000030                  MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001350  012737  000340  000032                  MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001356  012737  017276  000034                  MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001364  012737  000340  000036                  MOV     #340,@#TRAPVEC+2;LEVEL 7
001372  012737  017372  000024                  MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
001400  012737  000340  000026                  MOV     #340,@#PWRVEC+2 ;;LEVEL 7
001406  005037  001166                          CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
001412  012737  001412  001106                  MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
                                        ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                        ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
001420  013746  000004                          MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
001424  012737  001460  000004                  MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
001432  012737  177570  001140                  MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
001440  012737  177570  001142                  MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
001446  022777  177777  177464                  CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
001454  001012                                  BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                        ;;AND  THE HARDWARE SWR IS NOT = -1
001456  000403                                  BR      65$             ;;BRANCH IF NO TIMEOUT
001460  012716  001466          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
001464  000002                          RTI
001466  012737  000174  001140  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
001474  012737  000176  001142                  MOV     #DISPREG,DISPLAY
001502  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR


                                        ;*  PROGRAM TEST TO DETERMINE WHAT CLASS OF COMPUTER WE ARE RUNNING IN.
```

```
001506  012737  001534  000010  ST1:    MOV     #ST2,@#10       ;TO DO THIS WE WILL DO AN INSTRUCTION THAT
                                                                ;ONLY AN 11/40 OR GREATER MACHINE WILL RECOGNIZE.
                                                                ;WE'LL SET THE ILLEGAL INSTR. TRAP VECTOR IF
                                                                ;THE CPU WE'RE RUNNING ON CAN'T PREFORM AN
                                                                ;"XOR" INSTR. IT WILL TRAP.
001514  074101                          XOR     R1,R1           ;THIS "XOR" WILL CAUSE 11/20 OR
                                                                ;LOWER CLASS PDP11 CPU TO TRAP TO "ST2".

001516  012737  000006  006260          MOV     #6,INSRT1       ;WE DIDN'T TRAP!MUST BE ON 11/40 OR BETTER.
                                                                ;IN THAT CASE WE MUST REPLACE "RTI" INSTR. WITH
                                                                ;"RTT" INSTRUCTION WHERE WE RETURN FROM A TRACE TRAP CAL
001524  012737  000006  006310          MOV     #6,INSRT2       ;THERE ARE TWO LOCATIONS.

001532  000402                          BR      ST3             ;GOTO "ST3"

001534  062706  000004          ST2:    ADD     #4,R6           ;TRAPPED HERE IF ON 11/20 OR LESSER CPU.
                                                                ;FIX THE STACK POINTER.

001540  012737  000012  000010  ST3:    MOV     #12,@#10        ;RESTORE LOCATION 10 FOR FUTURE TRAPS.

                                        ;*END PROGRAM TEST FOR CLASS OF COMPUTER.

001546  012737  001302  001140          MOV     #SWREQ,SWR      ;POINT TO SOFTWARE SWR.

001554  005037  001174                  CLR     XCSRA           ;CLEAR ADDR. OF OUTPUT CSR.
001560  005037  001204                  CLR     RCSRA           ;CLEAR ADDR. OF INPUT CSR.
001564  013777  001202  177406          MOV     XVT2,@XVT       ;RESTORE VECTOR IN CASE OF ILLEGAL
001572  012777  104416  177402          MOV     #IOT†,@XVT2     ;INTERRUPT
001600  013777  001212  177402          MOV     RVT2,@RVT       ;RESTORE VECTOR IN CASE OF
001606  012777  104416  177376          MOV     #IOT†,@RVT2     ;ILLEGAL INTERRUPT.
001614  005037  001200                  CLR     XVT             ;CLEAR VECTOR ADDR.
001620  005037  001202                  CLR     XVT2            ;CLEAR VECTOR ADDR.
001624  005037  001210                  CLR     RVT             ;CLEAR VECTOR ADDR.
001630  005037  001212                  CLR     RVT2            ;CLEAR VECTOR ADDR.
001634  012777  000340  177354          MOV     #340,@TKV2      ;SET KEYBOARD STATUS VECTOR.
001642  042777  000100  177274          BIC     #BIT6,@$TKS     ;CLEAR INTERRUPT ENABLE.
001650  104401  001656                  TYPE    ,65$            ;;TYPE ASCIZ STRING
001654  000410                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>#MD-11-DZDRH-A#
001676                          64$:
                                1$:
001676  104401  001704                  TYPE    ,67$            ;;TYPE ASCIZ STRING
001702  000427                          BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  <15><12><12>#4.4.1 TERMINAL CHANGE REQUIRED (Y OR N)? #
001762                          66$:
001762  104406                  2$:     RDCHR                   ;GET OPER RESPONCE.
001764  012600                          MOV     (SP)+,R0        ;CHAR. ON STACK.
001766  120027  000116                  CMPB    R0,#'N          ;IS ANSWER NO?
001772  001007                          BNE     3$              ;IF NOT CONTINUE.
001774  104401  002002                  TYPE    ,69$            ;;TYPE ASCIZ STRING
002000  000402                          BR      68$             ;;GET OVER THE ASCIZ
                                ;;69$:  .ASCIZ  #NO#
002006                          68$:
002006  000137  002646                  JMP     INPASK          ;ASK FOR INPUT MODULE ADDRS.
```

```
                                3$:     CMPB    R0,#'Y          ;IS ANSWER "YES"?
                                        BNE     1$              ;IF NOT REASK QUESTIKN

                                        TYPE    ,71$            ;;TYPE ASCIZ STRING
                                        BR      70$             ;;GET OVER THE ASCIZ
                                ;;71$:  .ASCIZ  #YES#
                                70$:

                                4$:
                                        TYPE    ,73$            ;;TYPE ASCIZ STRING
                                        BR      72$             ;;GET OVER THE ASCIZ
                                ;;73$:  .ASCIZ  <15><12>#4.4.2 KEYBOARD CSR ADDR. OF NEW TERMINAL? #
                                72$:
                                        CLR     @$TKS
                                        RDOCT
                                        MOV     (SP)+,R0
                                        JSR     PC,CKADR        ;CHECK THAT ADDRESS IS LEGAL.
                                        BCS     4$              ;CARRY BIT SET ON RETURN MEANS NO.
                                        MOV     R0,R1           ;SET ADDR. OF NEW KEYBOARD CSR.
                                5$:
                                        TYPE    ,75$            ;;TYPE ASCIZ STRING
                                        BR      74$             ;;GET OVER THE ASCIZ
                                ;;75$:  .ASCIZ  <15><12>#4.4.3 ITS VECTOR ADDR.? #
                                74$:

                                        RDOCT                   ;GET VECTOR ADDR.
                                        MOV     (SP)+,R0        ;ADDR ON STACK.
                                        BIT     #BIT0,R0        ;ADD # ENTERED?
                                        BEQ     6$              ;NO-CONTINUE
                                        TYPE    ,77$            ;;TYPE ASCIZ STRING
                                        BR      76$             ;;GET OVER THE ASCIZ
                                ;;77$:  .ASCIZ  <15><12>#ODD NUMBER NOT EXCEPTED!#
                                76$:
                                        BR      5$
                                6$:     BIT     #176000,R0      ;NUMBER TOO LARGE?
                                        BEQ     7$              ;NO-CONTINUE
                                        TYPE    ,79$            ;;TYPE ASCIZ STRING
                                        BR      78$             ;;GET OVER THE ASCIZ
                                ;;79$:  .ASCIZ  <15><12>#NUMBER TOO LARGE!#
                                78$:
                                        BR      5$              ;REASK QUESTION

                                7$:     CMP     R0,#40          ;NUMBER TOO SMALL?
                                        BHIS    8$              ;NO-CONTINUE
                                        TYPE    ,81$            ;;TYPE ASCIZ STRING
                                        BR      80$             ;;GET OVER THE ASCIZ
                                ;;81$:  .ASCIZ  <15><12>#NUMBER TOO SMALL!#
                                80$:
                                        BR      5$              ;REASK QUESTION

                                8$:     MOV     TKV2,@TKV       ;RESTORE PREVIOUS TTY VECTOR.
                                        MOV     #IOTT,@TKV2
                                        MOV     R0,TKV          ;SAVE NEW TTY VECTOR.
                                        ADD     #2,R0
                                        MOV     R0,TKV2
                                        MOV     #340,@TKV2      ;PRIORITY 7 ON INTERRUPT
```

```
                                                10$:
002420  104401  002426                                  TYPE    ,83$            ;;TYPE ASCIZ STRING
002424  000426                                          BR      82$             ;;GET OVER THE ASCIZ
                                                ;;83$:  .ASCIZ  <15><12>#4.4.4 PRINTER CSR ADDR. OF NEW TERMINAL? #
002502                                          82$:
002502  104410                                          RDOCT                   ;GET ADDR.
002504  012600                                          MOV     (SP)+,R0        ;ADDR. ON STACK
002506  004737  013264                                  JSR     PC,CKADR        ;CHECK THAT ADDR. IS OK
002512  103742                                          BCS     10$             ;IF NOT REASK.

002514  104401  002522                                  TYPE    ,85$            ;;TYPE ASCIZ STRING
002520  000416                                          BR      84$             ;;GET OVER THE ASCIZ
                                                ;;85$:  .ASCIZ  <15><12>#SWITCHING TO NEW TERMINAL#
002556                                          84$:
002556  010137  001144                                  MOV     R1,$TKS
002562  062701  000002                                  ADD     #2,R1
002566  010137  001146                                  MOV     R1,$TKB

002572  010037  001150                                  MOV     R0,$TPS         ;STORE ADDR. OF PRINTER.
002576  062700  000002                                  ADD     #2,R0
002602  010037  001152                                  MOV     R0,$TPB

002606  104401  002614                                  TYPE    ,87$            ;;TYPE ASCIZ STRING
002612  000415                                          BR      86$             ;;GET OVER THE ASCIZ
                                                ;;87$:  .ASCIZ  <15><12>#NEW TERMINAL ACTIVE.#<15><12><12>
002646                                          86$:

002646  005077  176272                        INPASK: CLR     @$TKS
002652  104401  002660                                TYPE    ,65$            ;;TYPE ASCIZ STRING
002656  000422                                        BR      64$             ;;GET OVER THE ASCIZ
                                                ;;65$:  .ASCIZ  <15><12>#4.4.5 CSR ADDR. OF INPUT MODULE? #
002724                                          64$:
002724  104410                                          RDOCT                   ;GET ADDR.
002726  012600                                          MOV     (SP)+,R0        ;ADDR. ON STACK
002730  004737  013264                                  JSR     PC,CKADR        ;SEE IS ADDR. LEGAL.
002734  103744                                          BCS     INPASK          ;IF NOT-REASK.
002736  010037  001204                                  MOV     R0,RCSRA        ;STORE ADDR. OF CSR.
002742  062700  000002                                  ADD     #2,R0           ;MAKE ADDR. OF DBR
002746  010037  001206                                  MOV     R0,RDBRA        ;STORE DBR ADDR.

002752                                          OUTASK:
002752  104401  002760                                  TYPE    ,65$            ;;TYPE ASCIZ STRING
002756  000423                                          BR      64$             ;;GET OVER THE ASCIZ
                                                ;;65$:  .ASCIZ  <15><12>#4.4.6 CSR ADDR. OF OUTPUT MODULE? #
003026                                          64$:
003026  104410                                          RDOCT                   ;GET ADDR.
003030  012600                                          MOV     (SP)+,R0        ;ADDR ON STACK.
003032  004737  013264                                  JSR     PC,CKADR        ;SEE IF ADDR. LEGAL
003036  103745                                          BCS     OUTASK          ;IF NOT-REASK.
003040  010037  001174                                  MOV     R0,XCSRA        ;STORE ADDR. OF CSR
003044  062700  000002                                  ADD     #2,R0           ;FORM DBR ADDR.
003050  010037  001176                                  MOV     R0,XDBRA        ;STORE DBR ADDR.
003054  005037  001272                                  CLR     ST200           ;INDICATE HAD STARTED AT LOC 200.
                                                        .SBTTL  PROGRAM RESTART
```

```
   476

                ;*THIS SECTION ENTERED FROM START AT 210 OR ↑C


   482
   483  003060  012706  001100          RSTART: MOV     #STACK,SP
   484  003064  005037  001100                  CLR     $PASS
   485  003070  005037  001112                  CLR     $ERTTL          ;CLEAR ERROR COUNT.
   486  003074  005037  001304                  CLR     ERTYCN          ;CLEAR # OF ERRORS TYPED OUT.
   487  003100  012777  013434  176106          MOV     #KEYSRV,@TKV
   488  003106  000005                          RESET
   489  003110  012737  003130  000004          MOV     #1$,@#4         ;SET FOR TRAP IF NO SWR.
   490  003116  005037  000006                  CLR     @#6
   491  003122  005737  177570                  TST     @#177570        ;ADDRESS THE SWR IF TRAPS THEN NONE.
   492  003126  000407                          BR      2$              ;NO TRAP - CONTINUE.
   493
   494  003130  022626                  1$:     POPSP2                  ;WE TRAPPED-MUST BE NO HARDWARE SWR.
   495  003132  012737  001302  001300          MOV     #SWREQ,BSWR     ;SET SOFTWARE SWR AS ADDRESS OF SWR
   496  003140  012737  001162  001142          MOV     #$REGO,DISPLAY  ;MAKE SURE TO TAKE CARE OF DISPLAY.
   497
   498  003146  012737  000006  000004  2$:     MOV     #6,@#4          ;RESET THE TRAP VECTOR.
   499  003154  012737  104416  000006          MOV     #IOTT,@#6       ;TRAP ROUTINE IN CASE OF ILL TRAP.
   500  003162  005737  001272                  TST     ST200           ;WAS THIS PROG. ORIGINALLY STARTED AT 200?
   501  003166  001436                          BEQ     RST2            ;LOC "ST200" ZEROED BY START ROUTINE.
   502
   503  003170  104401  003176                  TYPE    ,65$            ;;TYPE ASCIZ STRING
   504  003174  000430                          BR      64$             ;;GET OVER THE ASCIZ
   505                                  ;;65$:  .ASCIZ  <15><12>#PROGRAM MUST BE STARTED AT LOC 200 INITAILLY#
   506  003256                          64$:
   507  003256  000000                          HALT
   508  003260  000137  000200                  JMP     @#200
   509
   510
   511  003264                          RST2:
   512  003264  104401  003272                  TYPE    ,65$            ;;TYPE ASCIZ STRING
   513  003270  000407                          BR      64$             ;;GET OVER THE ASCIZ
   514                                  ;;65$:  .ASCIZ  <15><12>#4.4.7 MODE?#
   515  003310                          64$:
   516  003310  005037  001104                  CLR     $ICNT
   517  003314  005037  001240                  CLR     PATOUT
   518  003320  012777  013434  175666          MOV     #KEYSRV,@TKV
   519  003326  042777  000100  175610          BIC     #BIT6,@$TKS     ;DISABLE TTY INTERRUPTS
   520  003334  104407                          RDLIN                   ;GET OPER TEXT.
   521  003336  052777  000100  175600          BIS     #BIT6,@$TKS
   522  003344  012600                          MOV     (SP)+,R0        ;ADDR. OF TEXT ON STACK.
   523  003346  111037  001220                  MOVB    (R0),MODE
   524  003352  121027  000113          1$:     CMPB    (R0),#'K        ;SLAVE AUTO MODE SELECTED "K"?
   525  003356  001002                          BNE     2$
   526  003360  000137  005770                  JMP     STARTS          ;YES-GOTO START SLAVE.
   527
   528  003364  121027  000105          2$:     CMPB    (R0),#'E        ;MASTER AUTO MODE SELECTED "E"?
   529  003370  001002                          BNE     3$
   530  003372  000137  006064                  JMP     STARTM          ;YES-GOTO START MASTER.
   531
```

```
                                                3$:     CMPB    (R0),#'L        ;INPUT MODULE LOGIC TEST "L"?
003402  001002                                          BNE     4$
003404  000137  010052                                  JMP     IMLT            ;YES-GOTO INPUT MOD. LOGIC TEST.

003410  121027  000106                        4$:     CMPB    (R0),#'F        ;OUTPUT MODULE LOGIC TEST?
003414  001002                                          BNE     5$
003416  000137  011052                                  JMP     OMLT            ;YES-DO OUTPUT MODULE LOGIC TEST

003422  121027  000115                        5$:     CMPB    (R0),#'M        ;MASTER-SLAVE AUTO MODE?
003426  001002                                          BNE     QE
003430  000137  006162                                  JMP     STRTMS          ;YES-DO MASTER-SLAVE AUTO.

003434  121027  000107                        QE:     CMPB    (R0),#'G        ;MODE > G?
003440  002410                                          BLT     1$              ;NO-MUST BE OUTPUT MODE.
003442  121027  000112                                  CMPB    (R0),#'J        ;MODE < = J?
003446  003005                                          BGT     1$              ;NO-GOT TO BE OUTPUT MODE.
003450  012737  014270  001234                          MOV     #TABLE,TABLEP
003456  000137  004662                                  JMP     INPM            ;GOTO INPUT MODE.

003462  121027  000101                        1$:     CMPB    (R0),#'A        ;MODE < A?
003466  002403                                          BLT     2$              ;YES-UNKNOWN MODE.
003470  121027  000104                                  CMPB    (R0),#'D        ;MODE > D?
003474  003401                                          BLE     OUTM            ;NO-OUTPUT MODE


003476                                          2$:
003476  000672                                  BR      RST2            ;REASK QUESTION
003500  005737  001174                OUTM:   TST     XCSRA           ;OUTPUT MODULE ENTERED?
003504  001527                                  BEQ     5$
003506  104401  003514                          TYPE    ,65$            ;;TYPE ASCIZ STRING
003512  000414                                  BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <15><12>#4.4.8 SOURCE OF DATA?#
003544                                  64$:

003544  104407                                  RDLIN                   ;GET ADDR. RESPONSE
003546  012600                                  MOV     (SP)+,R0        ;ADDR. OF RESPONSE ON STACK.
003550  121027  000101                          CMPB    (R0),#'A        ;GREATER THAN SOURCE "A"?
003554  002751                                  BLT     OUTM            ;YES-REASK QUESTION
003556  121027  000104                          CMPB    (R0),#'D        ;LESS THAN SOURCE "D"?
003562  003346                                  BGT     OUTM            ;NO-REASK QUESTION

003564  111037  001224                1$:     MOVB    (R0),PATFRO     ;STORE SOURCE OF DATA.
003570  123727  001224  000103                CMPB    PATFRO,#'C      ;SOURCE FROM PROGRAM GENERATION?
003576  001057                                  BNE     4$              ;NO-CONTINUE
                                                                        ;YES-ASK FOR PATTERN.
003600                                          2$:
003600  104401  003606                          TYPE    ,67$            ;;TYPE ASCIZ STRING
003604  000416                                  BR      66$             ;;GET OVER THE ASCIZ
                                        ;;67$:  .ASCIZ  <15><12>#4.4.9 STARTING PATTERN? #
003642                                  66$:
003642  104410                                  RDOCT                   ;GET FROM OPER.
003644  012637  001230                          MOV     (SP)+,PATTRN    ;STORE

003650                                          3$:
003650  104401  003656                          TYPE    ,69$            ;;TYPE ASCIZ STRING
003654  000416                                  BR      68$             ;;GET OVER THE ASCIZ
```

```
                                        ;;69$:  .ASCIZ  <15><12>#4.4.10 PATTERN MODIFIER? #
                                        68$:
003712                                          RDLIN                   ;GET OPER RESPONSE
003712  104407                                  MOV     (SP)+,R0        ;ADDR. OF RESPONSE ON STACK
003714  012600
003716  111037  001226                          MOVB    (R0),PATTM      ;STORE MODIFIER.
003722  121027  000101                          CMPB    (R0),#'A        ;GREATER THAN 'A?
003726  002750                                  BLT     3$              ;IF SO-CONTINUE
003730  121027  000104                          CMPB    (R0),#'D        ;LESS THAN, EQUAL TO 'D?
003734  003345                                  BGT     3$              ;IF NOT REASK.

003736  012737  014270  001232          4$:     MOV     #TABLE,TEXTP    ;POINT TO BEGINNING OF TABLE.
003744  005037  001236                          CLR     TTYPAT          ;CLEAR TTY CHANGE PATTERN.
003750  012777  000000  175216                  MOV     #0,@XCSRA
003756  005737  001174                          TST     XCSRA           ;IS AN ADDRESS PRESENT?
003762  001026                                  BNE     OUTMWK          ;YES-CONTINUE.

003764                                  5$:
003764  104401  003772                          TYPE    ,71$            ;;TYPE ASCIZ STRING
003770  000421                                  BR      70$             ;;GET OVER THE ASCIZ
                                        ;;71$:  .ASCIZ  <15><12>#NO OUTPUT MODULE ADDR. ENTERED!#
004034                                  70$:
004034  000137  002752                          JMP     OUTASK          ;ASK FOR ADDR.

                                                .SBTTL  OUTPUT MODULE ROUTINES


                ;*ROUTINE TO OUTPUT TO AN OUTPUT MODULE
                ;*USES ROUTINES "MODPAT" TO GET A PATTERN
                ;*AND "SNDPAT" TO DO ACTUAL SENDING.

004040  000410                  OUTMWK: BR      MODPAT          ;MODIFY OUTPUT PATTERN.
004042  000137  004322          OUTWR:  JMP     SNDPAT          ;SEND DATA

004046  023727  001240  000103  OUTWR2: CMP     PATOUT,#103     ;DID WE COME FROM INPUT MOD ROUTINE?
004054  001371                          BNE     OUTMWK          ;NO-LOOP
004056  000137  005100                  JMP     INPMWK          ;YES-RETURN.


                ;*MODPAT ROUTINE USED TO MODIFY OUTPUT PATTERN
                ;*      PATFRO=         FUNCTION
                ;*      -------         --------
                ;*
                ;*      101             PATTERN FROM SWITCH REGISTER
                ;*      102             PATTERN FROM TTY.
                ;*      103             PROGRAM GENERATED PATTERN.
                ;*      104             PATTERN FROM TABLE
                ;*

004062  123727  001224  000101  MODPAT: CMPB    PATFRO,#'A      ;PATTERN FROM SWR?
004070  001434                          BEQ     PSWR            ;YES-GO READ IT.

004072  123727  001224  000103  1$:     CMPB    PATFRO,#'C      ;PATTERN FROM PROGRAM GEN?
004100  001002                          BNE     2$              ;NO-CONTINUE.
004102  000137  004202                  JMP     PPM             ;YES-GENERATE
```

```
                                                    ;PATTERN FROM TTY?
004106  123727  001224  000102  2$:     CMPB    PATFRO,#'B
004114  001002                          BNE     3$          ;NO-CONTINUE.
004116  000137  004042                  JMP     OUTWR       ;YES-DONE AUTOMATICALLY.
                                                    ;PATTERN FROM THE TABLE
004122  017737  175104  001230  3$:     MOV     @TEXTP,PATTRN   ;LOAD NEW PATTERN
004130  062737  000002  001232          ADD     #2,TEXTP    ;UPDATE TABLE POINTER.
004136  023727  001232  014642  4$:     CMP     TEXTP,#TEXTE    ;DONE WHOLE TABLE?
004144  001402                          BEQ     6$          ;YES-
004146  000137  004042          5$:     JMP     OUTWR       ;NO-SEND PATTERN

004152  012737  014270  001232  6$:     MOV     #TEXT,TEXTP ;POINT TO BEGINNING OF TABLE.
004160  000772                          BR      5$

004162  017777  175112  174750  PSWR:   MOV     @BSWR,@SWR  ;READ SWITCH REGISTER
004170  017737  174744  001230          MOV     @SWR,PATTRN
004176  000137  004042                  JMP     OUTWR

                ;*PPM PROGRAM GENERATED PATTERN ROUTINE.
                ;*
                ;*      PATTM=          FUNCTION
                ;*      ------          --------
                ;*      101             INCREMENT
                ;*      102             DECREMENT
                ;*      103             ;FLOAT 0 - COMPLEMENT
                ;*      104             ;COMPLEMENT
                ;*

004202  123727  001226  000101  PPM:    CMPB    PATTM,#'A   ;INCREMENT PATTERN?
004210  001433                          BEQ     INCP        ;YES-GO DO IT

004212  123727  001226  000102  1$:     CMPB    PATTM,#'B   ;DECREMENT PATTERN?
004220  001432                          BEQ     DECP        ;YES-GO DO IT

004222  123727  001226  000103  2$:     CMPB    PATTM,#'C   ;FLOAT 0-COMPLEMENT?
004230  001031                          BNE     COMP        ;NO-MUST BE STRAIGHT COMPLEMENT.

004232  005737  001230          3$:     TST     PATTRN      ;PATTERN=0
004236  001003                          BNE     4$          ;NO-CONTINUE.
004240  005237  001230                  INC     PATTRN      ;YES-SET TO A ONE.
004244  000413                          BR      PPMEXT      ;EXIT

004246  105737  001231          4$:     TSTB    PATTRN+1    ;HIGH BYTE=0 (INDICATE COMPLEMENT)?
004252  001403                          BEQ     6$          ;

004254  105737  001230          5$:     TSTB    PATTRN      ;LOW BYTE=0 (INDICATES COMPLEMENT)?
004260  001015                          BNE     COMP        ;NO-THEN COMPLEMENT

004262  000241          6$:     CLC                         ;FLOAT COMP.
004264  006137  001230                  ROL     PATTRN
004270  005137  001230                  COM     PATTRN

004274  000137  004042  PPMEXT: JMP     OUTWR
```

```
                                INCP:   INC     PATTRN          ;INCREMENT.
004304  000773                          BR      PPMEXT
                                DECP:   DEC     PATTRN          ;DECREMENT
004312  000770                          BR      PPMEXT
                                COMP:   COM     PATTRN          ;COMPLEMENT
004320  000765                          BR      PPMEXT

                ;*ROUTINE TO SEND PATTERN TO OUTPUT MODULE
                ;*
                ;*      MODE            METHOD
                ;*      A               CONTINUOUS OUTPUT
                ;*      B               FLAG OUTPUT
                ;*      C               INTERRUPT OUTPUT.
                ;*      D               ;DELAYED OUTPUT.
                ;*
004322  005037  177776          SNDPAT: CLR     PS
004326  052777  000001  174640          BIS     #BIT00,@XCSRA
004334  123727  001220  000101          CMPB    MODE,#'A        ;CONTINUOUS OUTPUT.
004342  001002                          BNE     1$
004344  000137  004404                  JMP     AMODE           ;YES-GO OUTPUT
004350  123727  001220  000102  1$:     CMPB    MODE,#'B        ;FLAG OUTPUT?
004356  001002                          BNE     2$
004360  000137  004416                  JMP     BMODE           ;YES-GO OUTPUT.
004364  123727  001220  000103  2$:     CMPB    MODE,#'C        ;INTERRUPT OUTPUT?
004372  001002                          BNE     3$
004374  000137  004444                  JMP     CMODE           ;YES-GO OUTPUT.
                                                                ;DELAYED MODE.
004400  000137  004554          3$:     JMP     DMODE           ;GO-OUTPUT

                                        ;*ROUTINE TO DO A STRAIGHT OUTPUT.

004404  013777  001230  174564  AMODE:  MOV     PATTRN,@XDBRA   ;LOAD OUTPUT.
004412  000137  004046                  JMP     OUTWR2

                                        ;*FLAG MODE OUTPUT.

004416  004737  004614          BMODE:  JSR     PC,TSTFLG
004422  032777  000200  174544          BIT     #BIT7,@XCSRA    ;DONE SET?
004430  001772                          BEQ     BMODE           ;NO-WAIT.
004432  013777  001230  174536          MOV     PATTRN,@XDBRA   ;LOAD OUTPUT.
004440  000137  004046                  JMP     OUTWR2

                                        ;*INTERRUPT MODE OUTPUT

004444  104412                  CMODE:  GETXC
004446  012777  004512  174524          MOV     #2$,@XVT        ;SET VECTOR FOR INTERRUPT
004454  012777  000340  174520          MOV     #340,@XVT2      ;PRIORITY 7 ON INTERRUPT
004462  052777  000100  174504          BIS     #BIT6,@XCSRA    ;ALLOW INTERRUPT.
004470  000240                  1$:     NOP
004472  012737  000340  177776          MOV     #340,PS         ;LOCK OUT INTRS.
```

```
                                        JSR     PC,TSTFLG       ;CHECK REXMIT REQ.

                                        CLR     PS      ;NOW ALLOW INTRS. AGAIN.
                                        BR      1$

                                        ;*WILL INTERRUPT TO HERE

                                2$:     POPSP2                          ;RESET STACK.
                                        MOV     PATTRN,@XDBRA   ;OUTPUT PATTERN.
                                        BIC     #BIT6,@XCSRA    ;DISABLE ENABLE.
                                        MOV     XVT2,@XVT2
                                        MOV     #IOTT,@XVT2
                                        CLR     PS              ;ALLOW INTERRUPTS.
                                        JMP     OUTWR2

                                        ;*DELAYED OUTPUT MODE.

                                DMODE:  BIS     #BIT0,@XCSRA    ;DISABLE DRE.
                                1$:     JSR     PC,TSTFLG
                                        TSTB    @XCSRA
                                        BPL     1$
                                        BIC     #BIT00,@XCSRA
                                        MOV     PATTRN,@XDBRA   ;LOAD OUTPUT.
                                        JMP     OUTWR2

                                TSTFLG:
                                        TST     @XCSRA          ;IS RECEIVER REQUESTING REXMIT
                                        BPL     1$              ;OF DATA??
                                        SUB     #2,TEXTP
                                        JSR     PC,RXMIT        ;YES-DO THE PROTOCAL.
                                        CMP     PATFRO,#'D      ;PATTERN FROM TABLE?
                                        BNE     1$
                                        MOV     @TEXTP,PATTRN   ;YES RESET PATTRN.
                                        ADD     #2,TEXTP
                                1$:
                                        RTS PC                  ;EXIT BACK TO WHENCE YOU CAME.


                ;*INPUT MODULE ROUTINE MODES G-J

                                INPM:   TST     RCSRA           ;ANY ADDR ENTERED?
                                        BNE     1$

                                        TYPE    ,65$            ;;TYPE ASCIZ STRING
                                        BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>#NO INPUT MODULE ADDR. ENTERED!#
                                64$:
                                        JMP     INPASK

                                1$:
                                        TYPE    ,67$            ;;TYPE ASCIZ STRING
                                        BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  <15><12>#4.4.11 METHOD OF REPORTING DATA ? #
```

```
005020                          66$:

                                        RDLIN                   ;GET RESPONSE
005020  104407                          MOV     (SP)+,R0        ;ADDR OF RESPONSE ON STACK
005022  012600                          MOV     (R0),PATOUT     ;SAVE
005024  011037  001240                  CMPB    (R0),#'A        ;> OR= TO 'A?
005030  121027  000101                  BLT     1$
005034  002743                          CMPB    (R0),#'D        ;< OR= TO 'D?
005036  121027  000104                  BGT     1$
005042  003340                          TST     @RDBRA
005044  005777  174136                  MOV     MODE,MODEI
005050  013737  001220  001222          SUB     #6,MODE
005056  162737  000006  001220

005064  023727  001240  000103          CMP     PATOUT,#'C      ;CONNECTED TO OUTPUT?
005072  001002                          BNE     INPMWK          ;NO-CONTINUE
005074  000137  003500                  JMP     OUTM            ;YES-MORE QUESTIONS

                                        .SBTTL  INPUT MODULE ROUTINES


                ;*THIS ROUTINE INPMWK TAKES CARE OF MODES G-J
                ;*OF READING DATA.

005100  000402                  INPMWK: BR      REDMOD          ;GO READ DATA FROM MODULE.

005102  000137  005300          INPRD:  JMP     REPPAT          ;REPORT DATA.


                ;*REDMOD THIS ROUTINE DETERMINES HOW WE ARE
                ;*       GOING TO READ THE INPUT MODULE
                ;*
                ;*       MODE            METHOD
                ;*       ----            ------
                ;=       G               CONTINUOUS
                ;*       H               FLAG
                ;*       I               INTERRUPT
                ;*       J               DELAYED
                ;*

005106  123727  001222  000107  REDMOD: CMPB    MODEI,#'G       ;CONTINUOUS READ 'G?
005114  001411                          BEQ     GMODE

005116  123727  001222  000110  1$:     CMPB    MODEI,#'H               ;FLAG MODE READ 'H?
005124  001412                          BEQ     HMODE

005126  123727  001222  000111  2$:     CMPB    MODEI,#'I       ;INTERRUPT MODE READ 'I?
005134  001416                          BEQ     IMODE

005136  000444                  3$:     BR      JMODE           ;DELAYED MODE.

                                        ;*ROUTINE TO CONTINUOUS READ
```

```
                                GMODE:  MOV     @RDBRA,PATTIN   ;READ INPUT MODULE.
                                        JMP     INPRD

                                ;*ROUTINE TO READ WHEN FLAG IS SET.
                                ;*>>WARNING<<IF DISPLAYING RESULTS IN DATA LIGHTS, A RESET
                                ;*(SYSTEM INITIALIZE) IS USED AND COULD CLEAR THE FLAG.

                                HMODE:  TSTB@RCSRA              ;FLAG SET?
                                        BPL     HMODE           ;NO-WAIT FOR IT
                                        MOV     @RDBRA,PATTIN   ;YES-READ
                                        JMP     INPRD

                                ;*IMODE ROUTINE TO READ THE INPUT MODULE ON INTERRUPT.
                                ;*>>WARNING<< IF DISPLAYING RESULTS IN DATA LIGHTS, A RESET
                                ;*(SYSTEM INITIALIZE IS USED AND COULD CLEAR MODULES "READY" FLAG (CSR
                                ;*BIT 7) AND NO INTERRUPT WOULD OCCUR.

                                IMODE:  GETRC                           ;GET INTERRUPT VECTOR.
                                        MOV     #2$,@RVT        ;SET UP INTERRUPT VECTOR.
                                        MOV     #340,@RVT2      ;PRIORITY INTERRUPT VECTOR
                                        BIS     #BIT6,@RCSRA    ;SET INTERRUPT ENABLE
                                        CLR     PS
                                1$:     WAIT                    ;WAIT HERE FOR INTERRUPT MODULE
                                        BR      1$              ;SHOUD INTERRUPT WHEN IT RECEIVES DAATA

                                2$:     POPSP2
                                        BIT     #BIT6,@RCSRA    ;DISABLE ENABLE
                                        MOV     @RDBRA,PATTIN   ;READ MODULE.
                                        JMP     INPRD

                                ;*ROUTINE TO READ DELAYED MODE.

                                JMODE:  BIC     #BIT0,@RCSRA    ;CLEAR DRE
                                        MOV     @RDBRA,PATTIN   ;READ MODULE.
                                        NOP                     ;STALL
                                        BIS     #BIT0,@RCSRA    ;SET DRE
                                        JMP     INPRD

        ;*ROUTINE USED BY INPMWK TO DISPLAY DATA READ
        ;*FROM INPUT MODULE. XFERS CONTROL TO RIGHT ROUTINE.
        ;*
        ;*      PATOUT=         METHOD
        ;*      -------         ------
        ;*      101             DATA LIGHTS
        ;*      102             TTY OUTPUT ON CHANGE
        ;*      103             ERROR CHECK OUTPUT > INPUT
        ;*      104             ERROR CHECK OUTPUT(OTHER CPU)>>INPUT (THIS CPU)
        ;*

                                REPPAT: CMPB    PATOUT,#'A      ;DISPLAY IN LIGHTS 'A?
                                        BNE     1$
                                        JMP     DISLGH          ;YES-GO DISPLAY

                                1$:     CMPB    PATOUT,#'B      ;REPORT ON TTY 'B?
```

```
                                                BNE     11$
                                                JMP     DISTTY          ;YES-GO REPORT

005330  123727  001240  000104  11$:  CMPB    PATOUT,#'D
005336  001002                        BNE     2$
005340  000137  005700                JMP     CIO
                                                                        ;OUTPUT > INPUT
005344  005237  001104          2$:   INC     $ICNT
005350  104414                        SWCAL
005352  023727  001104  030000        CMP     $ICNT,#30000
005360  001003                        BNE     3$
005362  005037  001104                CLR     $ICNT
005366  104413                        EOPCAL
005370  023737  001230  001242  3$:   CMP     PATTRN,PATTIN   ;DATA OUTPUTTED=DATA READ
005376  001002                        BNE     5$              ;NO-REPORT ERROR
005400  000137  005536          4$:   JMP     7$              ;YES-EXIT
005404  005237  001112          5$:   INC     $ERTTL
005410  104414                        SWCAL
005412  032777  020000  173520        BIT     #BIT13,@SWR     ;INHIBIT ERROR TYPEOUT?
005420  001040                        BNE     6$

005422  104401  005430                TYPE    ,65$            ;;TYPE ASCIZ STRING
005426  000413                        BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>#GOOD DATA  BAD DATA#
005456                          64$:
005456  104401  005464                TYPE    ,67$            ;;TYPE ASCIZ STRING
005462  000403                        BR      66$             ;;GET OVER THE ASCIZ
                                ;;67$:  .ASCIZ  <15><12>#  #
005472                          66$:
005472  013746  001230                MOV     PATTRN,-(SP)    ;;SAVE PATTRN FOR TYPEOUT
005476  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
005500  104401  005506                TYPE    ,69$            ;;TYPE ASCIZ STRING
005504  000403                        BR      68$             ;;GET OVER THE ASCIZ
                                ;;69$:  .ASCIZ  #    #
005514                          68$:
005514  013746  001242                MOV     PATTIN,-(SP)    ;;SAVE PATTIN FOR TYPEOUT
005520  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)

005522  104414                  6$:   SWCAL
005524  032777  100000  173406        BIT     #BIT15,@SWR     ;HALT ON ERROR?
005532  001401                        BEQ     7$              ;NO CONTINUE
005534  104415                        HALTCL                  ;HALTED ON DATA ERROR
                                                                ;DATA SENT TO OUTPUT MODULE
                                                                ;DOES NOT MATCH DATA READ
                                                                ;IN INPUT MODULE
                                                                ;DATA OUTPUTTED IN PATTRN
                                                                ;DATA READ IN PATTIN
005536  104414                  7$:   SWCAL
005540  032777  040000  173372        BIT     #BIT14,@SWR     ;LOOP ON CURRENT PATTERN (IF FROM TABLE)
005546  001002                        BNE     9$              ;YES-7$
005550  000137  004040          8$:   JMP     OUTMWK

005554  022737  014270  001234  9$:   CMP     #TABLE,TABLEP   ;BEGINNING OF TABLE?
005562  001004                        BNE     10$             ;NO-GO 9$
005564  012737  014266  001234        MOV     #TABLE-2,TABLEP ;YES POINT TO END OF TABLE.
005572  000766                        BR      8$              ;EXIT
```

```
                                                ;*DISLGH ROUTINE TO DISPLAY DATA
                                                ;
005604  013700  001242          DISLGH: MOV     PATTIN,R0       ;DISPLAY USING DATA LIGHTS
005610  104417                          RESETC                  ;DISPLAY
005612  052777  000100  173324          BIS     #BIT6,@$TKS     ;REENABLE TTY INTERRUPTS.
005620  000137  005100                  JMP     INPMWK          ;EXIT
                                                ;
                                                ;*ROUTINE TO TYPE INPUT DATA CHANGE ON TTY.
                                                ;
005624  023737  001244  001242  DISTTY: CMP     PATOLD,PATTIN   ;HAS INPUT DATA CHANGED?
005632  001415                          BEQ     1$              ;NO-GO EXIT
005634  104414                          SWCAL
005636  032777  020000  173274          BIT     #BIT13,@SWR     ;INHIBIT TYPEOUT?
005644  001010                          BNE     1$
005646  104401  005654                  TYPE    ,65$            ;;TYPE ASCIZ STRING
005652  000402                          BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <15><12>
005660                                  64$:
005660  013746  001242                  MOV     PATTIN,-(SP)    ;;SAVE PATTIN FOR TYPEOUT
005664  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                                ;
005666  013737  001242  001244  1$:     MOV     PATTIN,PATOLD   ;OLD PATTERN=NEW PATTERN
005674  000137  005100                  JMP     INPMWK
                                                ;
                                                ;*
                                                ;*ROUTINE USED BY REPPAT TO COMPARE DATA IN TO
                                                ;*TABLE DATA.
                                                ;*REPORT DATA MEATHOD ""D""
                                                ;*
                                                ;
005700  023777  001242  173326  CIO:    CMP     PATTIN,@TABLEP  ;DATA RECEIVED=TABLE DATA?
005706  001403                          BEQ     1$              ;YES-GOTO END.
                                                ;
005710  004737  006450                  JSR     PC,SLERR        ;NO-REPORT ERROR.
005714  000403                          BR      2$
005716  062737  000002  001234  1$:     ADD     #2,TABLEP       ;UPDATE TAPBLE POINT FOR NEXT DATA.
005724  023727  001234  014642  2$:     CMP     TABLEP,#TABLEE  ;DONE WHOLE TABLE?
005732  001014                          BNE     3$
005734  012737  014270  001234          MOV     #TABLE,TABLEP   ;YES-RESET TABLE POINTER.
005742  005237  001104                  INC     $ICNT           ;UPDATE ITERATION COUNT.
005746  023727  001104  001000          CMP     $ICNT,#1000     ;DONE TABLE 1000 TIMES?
005754  001003                          BNE     3$              ;NO-EXIT.
                                                ;
005756  104413                          EOPCAL                  ;YES REPOR END O' PASS.
005760  005037  001104                  CLR     $ICNT
                                                ;
005764  000137  005100          3$:     JMP     INPMWK          ;EXIT.
                                                ;
                                        .SBTTL  SLAVE-MASTER AUTO SEND MODES
```

```
1036
1037
                        ;*STARTS ROUTINE TO START SLAVE AUTO MODE.
1041
1042  005770  005737  001204        STARTS: TST     RCSRA           ;ANY INPUT ADDR. ENTERED?
1043  005774  001402                        BEQ     1$              ;NO-REPORT ERROR
1044  005776  000137  006312                JMP     SMST            ;YES-START SLAVE
1045
1046  006002  005037  177776        1$:     CLR     PS              ;NO CLEAR TRAP BIT IN CASE OF ENTRY FROM STRTMS
1047  006006  104401  006014                TYPE    ,65$            ;;TYPE ASCIZ STRING
1048  006012  000420                        BR      64$             ;;GET OVER THE ASCIZ
1049                                ;;65$:  .ASCIZ  <15><12>#NO INPUT MODULE ADDR ENTERED!#
1050  006054                        64$:
1051  006054  012706  001100                MOV     #STACK,SP       ;RESET STACK
1052  006060  000137  002646                JMP     INPASK          ;GO GET ADDR.
1053
                        ;*STARTM ROUTINE USED TO START MASTER AUTO MODE
1057
1058  006064  005737  001174        STARTM: TST     XCSRA           ;ANY OUTPUT MODULE ADDR. ENTERED?
1059  006070  001402                        BEQ     1$              ;NO REPORT ERROR
1060  006072  000137  007304                JMP     MST             ;YES-START TEST.
1061
1062  006076  005037  177776        1$:     CLR     PS              ;CLEAR T BIT TRAPS IF RUNNING
1063                                                                ;MASTER-SLAVE AUTO
1064  006102  104401  006110                TYPE    ,65$            ;;TYPE ASCIZ STRING
1065  006106  000421                        BR      64$             ;;GET OVER THE ASCIZ
1066                                ;;65$:  .ASCIZ  <15><12>#NO OUTPUT MODULE ADDR. ENTERED#
1067  006152                        64$:
1068  006152  012706  001100                MOV     #STACK,SP       ;RESET STACK POINTER
1069  006156  000137  002752                JMP     OUTASK          ;GET ADDR.
1070
                        ;*STRTMS USED TO START MASTER (STARTM) AND SLAVE
                        ;*(STARTS) ROUTINE IF BOTH INPUT AND OUTPUT RESIDE ON SAME CPU.
1075
1076  006162  012737  006064  001246  STRTMS: MOV     #STARTM,PTRM    ;POINT TO BEGINNING OF MASTER ROUTINE
1077  006170  012737  005770  001250          MOV     #STARTS,PTRS    ;POINT TO BEGINNING OF SLAVE ROUTINE.
1078
1079  006176  012746  000020                 MOV     #20,-(SP)       ;SET T BIT ON STACK
1080  006202  013746  001246                 MOV     PTRM,-(SP)      ;GO TO MASTER ROUTINE FIRST
1081  006206  012737  006232  000014         MOV     #MTTRP,@#14     ;SET FOR T BIT TRAP.
1082  006214  012737  000340  000016         MOV     #340,@#16       ;PRIORITY 7 ON TRAP
1083  006222  012737  000020  001254         MOV     #20,$SLST       ;T BIT IN SLAVE STATUS
1084  006230  000002                         RTI                     ;EXECUTE FIRST INSTRUCTION OF
1085                                                                 ;MASTER ROUTINE
1086
1087                                ;*MTTRAP ENTER HERE ON TRAPS FROM MASTER ROUTINE
1088
1089  006232                        MTTRP:
1090                                                                 ;YES MUST BE 11/40 OR 11/45 DO RTT
1091
```

```
1092  006232  012737  006262  000014  1$:     MOV     #STTRP,@#14     ;SET TO TRAP TO SLAVE HANDLER
1093  006240  012637  001246                  MOV     (SP)+,PTRM      ;SAVE MASTER PC.
1094  006244  012637  001252                  MOV     (SP)+,SMSST     ;SAVE MASTER STATUS
1095  006250  013746  001254                  MOV     SSLST,-(SP)     ;USE SLAVE STATUS
1096  006254  013746  001250                  MOV     PTRS,-(SP)      ;USE SLAVE PC.
1097  006260  000002                  INSRT1: RTI                     ;WILL BE REPLACE TO RTT(000006) IF 11/40 OR GREATER.
1098
1099                                          ;* STTRP ENTER HERE ON T BIT TRAPS FROM MASTER ROUTINE
1100
1101  006262                          STTRP:
1102
1103  006262  012737  006232  000014  1$:     MOV     #MTTRP,@#14     ;SET TO TRAP TO MASTER HANDLER.
1104  006270  012637  001250                  MOV     (SP)+,PTRS      ;SAVE SLAVE'S PC
1105  006274  012637  001254                  MOV     (SP)+,SSLST     ;SAVE SLAVE'S STATUS
1106  006300  013746  001252                  MOV     SMSST,-(SP)     ;USE MASTER'S STATUS
1107  006304  013746  001246                  MOV     PTRM,-(SP)      ;USE MASTER'S PC.
1108  006310  000002                  INSRT2: RTI                     ;WILL VE REPLACED TO RTT(000006) IF 11/40 OR GREATER.
1109
1110                                          .SBTTL  SLAVE AUTO RECIEVE MODE
1111
1112

                ;*ACTUAL SLAVE FLOWS

1116
1117  006312  104401  015226          SMST:   TYPE    ,MSM            ;TYPE "START MODE"
1118
1119                                          ;*INITIATED INTER-CPU-DIALOGUE
1120  006316  005777  172662          SP1:    TST     @RCSRA          ;STAT BIT SET?
1121  006322  100375                          BPL     SP1             ;NO WAIT FOR IT. NOTE: STAT BIT SETS
1122                                                                  ;WHEN MASTER SETS ITS CONTR BIT (BIT 8)
1123
1124  006324  052777  000400  172652  SP2:    BIS     #BIT8,@RCSRA    ;SET CONTR. BIT (SETS MASTER'S STAT BIT)
1125
1126  006332  005777  172646          SP3:    TST     @RCSRA          ;STAT BIT CLEAR?
1127  006336  100775                          BMI     SP3             ;NO WAIT FOR IT. NOTE: STAT BIT
1128                                                                  ;CLEARS WHEN CLEARS IT CONTR BIT
1129
1130  006340  005777  172642          SP3A:   TST     @RDBRA                  ;DUMB READ.
1131  006344  042777  000400  172632          BIC     #BIT8,@RCSRA    ;CLEAR CONTR BIT
1132
1133
1134                                          .SBTTL  SLAVE IMMED. SEND MODE
1135
1136                                          ;*IMMEDIATE SEND MODE
1137
1138  006352  012737  001000  001260  SISM:   MOV     #1000,SXNO              ;SET ITERATION COUNT
1139
1140  006360  104401  015243          SP13:   TYPE    ,MIM            ;TYPE "IMMED. SEND MODE"
1141
1142  006364  012737  014270  001234  SP14:   MOV     #TABLE,TABLEP   ;POINT TO BEGINNING OF TEXT.
1143
1144  006372  022737  014642  001234  SP15:   CMP     #TABLEE,TABLEP  ;DONE ALL TEXT?
1145  006400  001521                          BEQ     SP22            ;YES POINT 22
1146
1147  006402  105777  172576          SP18:   TSTB    @RCSRA          ;DATA AVAILABLE?
```

```
1148  006406  100375                          BPL     SP18            ;NO-WAIT FOR IT
1149
1150  006410  017737  172572  001242  SP16:   MOV     @RDBRA,PATTIN   ;YES-READ DATA
1151  006416  023777  001242  172610          CMP     PATTIN,@TABLEP  ;DATA READ=TABLE?
1152  006424  001005                          BNE     SP19            ;NO-REPORT ERROR
1153  006426  062737  000002  001234          ADD     #2,TABLEP       ;YES-UPDATE TABLE POINTER
1154  006434  000137  006372                  JMP     SP15            ;LOOP
1155
1156  006440  004737  006450          SP19:   JSR     PC,SLERR        ;REPORT ERROR HANDLER.
1157  006444  000137  006372                  JMP     SP15
1158
1159  006450  104414                  SLERR:  SWCAL
1160  006452  032777  020000  172460          BIT     #BIT13,@SWR     ;INHIBIT ERROR TYPEOUT?
1161  006460  001042                          BNE     1$              ;YES-SKIP IT
1162
1163  006462  104401  006470                  TYPE    ,65$            ;;TYPE ASCIZ STRING
1164  006466  000414                          BR      64$             ;;GET OVER THE ASCIZ
1165                                  ;;65$:  .ASCIZ  <15><12>#ERROR DATA  GOOD DATA#
1166  006520                          64$:
1167  006520  104401  006526                  TYPE    ,67$            ;;TYPE ASCIZ STRING
1168  006524  000403                          BR      66$             ;;GET OVER THE ASCIZ
1169                                  ;;67$:  .ASCIZ  <15><12>#  #
1170  006534                          66$:
1171  006534  013746  001242                  MOV     PATTIN,-(SP)    ;;SAVE PATTIN FOR TYPEOUT
1172  006540  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1173  006542  104401  006550                  TYPE    ,69$            ;;TYPE ASCIZ STRING
1174  006546  000404                          BR      68$             ;;GET OVER THE ASCIZ
1175                                  ;;69$:  .ASCIZ  #      #
1176  006560                          68$:
1177  006560  017746  172450                  MOV     @TABLEP,-(SP)   ;;SAVE @TABLEP FOR TYPEOUT
1178  006564  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1179
1180  006566  005237  001112          1$:     INC     $ERTTL
1181  006572  104414                          SWCAL
1182  006574  032777  004000  172336          BIT     #BIT11,@SWR     ;CONTINUE PAST ERROR?
1183  006602  001404                          BEQ     SP17            ;NO-POINT 17
1184  006604  062737  000002  001234          ADD     #2,TABLEP       ;YES-UPDATE POINTER
1185  006612  000207                          RTS PC
1186
1187  006614  052777  000400  172362  SP17:   BIS     #BIT8,@RCSRA    ;SET CONTR BIT-REQUEST REXMIT OF DATA
1188
1189  006622  005777  172356          SP20:   TST     @RCSRA          ;STAT BIT SET? SET WHEN MASTER
1190  006626  100375                          BPL     SP20            ;NO-WAIT FOR IT
1191
1192  006630  005777  172352          SP21:   TST     @RDBRA  ;DUMB READ.
1193  006634  042777  000400  172342          BIC     #BIT8,@RCSRA    ;YES-CLEAR CONTROL BIT
1194  006642  000207                          RTS PC
1195
1196  006644  005337  001260          SP22:   DEC     $XNO            ;XNO=XNO-1 DECREMENT ITERATION COUNT
1197
1198  006650  001245                          BNE     SP14            ;OTHERWISE ITERATE
1199  006652  104414                          SWCAL
1200  006654  032777  040000  172256          BIT     #BIT14,@SWR
1201  006662  001233                          BNE     SISM
1202  006664                          1$:
1203
```

```
1204                                            .SBTTL  SLAVE BURST SEND MODE
1205
1206
1207                                            ;*BURST SEND MODE
1208
1209  006664  104401  015267          SBSM:   TYPE    ,MBM            ;TYPE "BURST SEND MODE"
1210
1211  006670  012737  001000  001260  SP24:   MOV     #1000,SXNO              ;SET ITERATION COUNT
1212
1213  006676  005737  001260          SP25:   TST     SXNO            ;DONE 5 TIMES?
1214  006702  001010                          BNE     SP26            ;NO-CONTINUE POINT 26.
1215  006704  104414                          SWCAL
1216
1217  006706  032777  040000  172224          BIT     #BIT14,@SWR
1218  006714  001363                          BNE     SBSM
1219  006716                          SP32:
1220  006716  104413                          EOPCAL
1221  006720  000137  006312                  JMP     SMST
1222
1223  006724  005777  172254          SP26:   TST     @RCSRA          ;STAT BIT SET?
1224  006730  100375                          BPL     SP26            ;NO-WAIT FOR IT
1225
1226  006732  052777  000400  172244  SP27:   BIS     #BIT8,@RCSRA    ;YES-SET CONTROL BIT
1227
1228  006740  005777  172240          SP28:   TST     @RCSRA          ;STAT BIT CLEAR?
1229  006744  100775                          BMI     SP28            ;NO-WAIT TILL CLEAR
1230
1231  006746  042777  000400  172230  SP29:   BIC     #BIT8,@RCSRA    ;CLEAR CONTROL BIT
1232  006754  012737  014644  001262          MOV     #STABL,STABLP   ;POINT TO BEGINNING OF TABLE
1233  006762  012737  014270  001234          MOV     #TABLE,TABLEP
1234  006770  104411                          GETRC                   ;GFET VECTOR ADDR.
1235  006772  012777  007260  172210          MOV     #BISR,@RVT      ;SET FOR INTERRUPT
1236  007000  012777  000340  172204          MOV     #340,@RVT2      ;PRIORITY 7 ON INTR.
1237  007006  052777  000100  172170          BIS     #BIT6,@RCSRA    ;SET INTR. ENABLE
1238
1239  007014  042737  000340  177776          BIC     #340,PS
1240  007022  000240                  SP30:   NOP                     ;RECEIVE INTERRUPTS FOR EACH DATA XFER
1241
1242  007024  023727  001234  014644  SP31:   CMP     TABLEP,#TABLEE+2        ;RECEIVED WHOLE TEXT?
1243  007032  001373                          BNE     SP30            ;NO-WAIT FOR THEM
1244  007034  042777  000100  172142          BIC     #BIT6,@RCSRA    ;DON'T ALLOW INPUT TO INTERRUPT.
1245
1246  007042  012737  014644  001262  SP33:   MOV     #STABL,STABLP   ;POINT TO BEGINNING OF STORAGE TABLE
1247  007050  012737  014270  001234          MOV     #TABLE,TABLEP   ;POINT TO TEXT TABLE
1248
1249  007056  027777  172200  172150  SP34:   CMP     @STABLP,@TABLEP ;DATA RECEIVED=TEXT?
1250  007064  001016                          BNE     SP37            ;NO REPORT ERROR
1251
1252  007066  062737  000002  001262  SP35:   ADD     #2,STABLP       ;UPDATE TABLE POINTERS
1253  007074  062737  000002  001234          ADD     #2,TABLEP
1254
1255  007102  022737  014642  001234  SP36:   CMP     #TABLEE,TABLEP  ;COMPARED WHOLE TEXT?
1256  007110  001362                          BNE     SP34            ;NO-DO NEXT COMPARE
1257
1258  007112  005337  001260          SP36A:  DEC     SXNO            ;YES UP ITERATIN COUNT
1259  007116  000137  006676                  JMP     SP25            ;LOOP
```

```
1260
1261  007122  104414                  SP37:   SWCAL
1262  007124  032777  020000  172006          BIT     #BIT13,@SWR     ;INHIBIT ERROR TYPEOUT?
1263  007132  001042                          BNE     1$              ;YES-THEN SKIP
1264
1265  007134  104401  007142                  TYPE    ,65$            ;;TYPE ASCIZ STRING
1266  007140  000414                          BR      64$             ;;GET OVER THE ASCIZ
1267                                  ;;65$:  .ASCIZ  <15><12>#ERROR DATA  GOOD DATA#
1268  007172                          64$:
1269  007172  104401  007200                  TYPE    ,67$            ;;TYPE ASCIZ STRING
1270  007176  000403                          BR      66$             ;;GET OVER THE ASCIZ
1271                                  ;;67$:  .ASCIZ  <15><12>#  #
1272  007206                          66$:
1273  007206  017746  172022                  MOV     @TABLEP,-(SP)   ;;SAVE @TABLEP FOR TYPEOUT
1274  007212  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1275  007214  104401  007222                  TYPE    ,69$            ;;TYPE ASCIZ STRING
1276  007220  000404                          BR      68$             ;;GET OVER THE ASCIZ
1277                                  ;;69$:  .ASCIZ  #       #
1278  007232                          68$:
1279  007232  017746  172024                  MOV     @STABLP,-(SP)   ;;SAVE @STABLP FOR TYPEOUT
1280  007236  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1281
1282  007240  005237  001112          1$:     INC     $ERTTL
1283  007244  104414                          SWCAL
1284  007246  032777  004000  171664          BIT     #BIT11,@SWR     ;CONTINUE?
1285  007254  001316                          BNE     SP36A           ;YES
1286  007256  000703                          BR      SP35            ;NO-LOOP
1287
1288                                  ;*INTERRUPT HANDLER FOR SBSM
1289
1290  007260  017777  171722  171774  BISR:   MOV     @RDBRA,@STABLP  ;READ AND STORE DATA
1291  007266  062737  000002  001262          ADD     #2,STABLP       ;UPDATE STORE POINTER
1292  007274  062737  000002  001234          ADD     #2,TABLEP
1293  007302  000002                          RTI                     ;EXIT THIS INTERRUPT
1294                                  .SBTTL  MASTER AUTO SEND MODES-- START MODE
1295

                ;*INITIAL INTER-CPU DIALOGE
```

```
1299
1300  007304  104401  015226          MST:   TYPE   ,MSM            ;TYPE "START MODE"
1301
1302  007310  052777  000400  171656  P1:    BIS    #BIT8,  @XCSRA  ;SET CONTROL BIT.
1303                                                                 ;(SETS SLAVE'S STAT BIT).
1304
1305  007316  005777  171652          P2:    TST    @XCSRA          ;NO MASTER'S STAT BIT SET?
1306  007322  100375                        BPL    P2              ;SET WHEN SLAVE SETS IT'S
1307                                                                 ;CONTR. BIT.
1308                                                                 ;WAIT TILL SET.
1309
1310  007324  042777  000400  171642  P3:    BIC    #BIT8,  @XCSRA  ;CLEAR CONTROL BIT.
1311                                                                 ;(CLEARS SLAVE'S STAT BIT).
1312  007332  005777  171636          TST    @XCSRA
1313  007336  100772                        BMI    P3
1314
1315                                        .SBTTL  MASTER  IMMEDIATE SEND MODE
1316


              ;*ISM   IMMEDIATE MODE SEND


1322
1323  007340  052777  000001  171626  ISM:   BIS    #BIT0,  @XCSRA  ;SET BIT 0.
1324  007346  012737  001000  001256        MOV    #1000,  XNO     ;SET XMITT LOOP TO 5
1325
1326  007354  104401  015243          P13:   TYPE   ,MIM            ;TYPE "IMMED. SEND MODE"
1327
1328  007360  012737  014270  001232  P14:   MOV    #TEXT,  TEXTP   ;POINT TO BEGINNING OF TEXT.
1329
1330  007366  023727  001232  014642  P15:   CMP    TEXTP,  #TEXTE  ;TRANSMITTED WHOLE TEXT?
1331  007374  001016                        BNE    P16             ;NO-CONTINUE.
1332
1333  007376  162737  000001  001256  P22:   SUB    #1,     XNO     ;XNO=XNO-1
1334
1335  007404  001365                  P23:   BNE    P14             ;IF NOT DONE 5 TIMES REPEAT.
1336  007406  104414                        SWCAL
1337  007410  032777  040000  171522        BIT    #BIT14,@SWR
1338  007416  001350                        BNE    ISM
1339  007420                          1$:
1340
1341  007420  000137  007574                JMP    BSM             ;IF 5 THEN BUST MODE.
1342
1343
1344  007424  005777  171544          P17:   TST            @XCSRA          ;STAT BIT SET? SET BY SLAVE WHEN
1345                                                                 ;SLAVE WANTS LAST DATA
1346                                                                 ;RETRANSMITTED.
1347  007430  100413                        BMI    P19             ;IF SO GO DO IT.
1348
1349  007432  032777  000200  171534  P16:   BIT    #BIT7,@XCSRA    ;DATA REQUEST SET?
1350  007440  001771                        BEQ    P17             ;IF NOT WAIT FOR IT.
1351
1352  007442  017777  171564  171526  P18:   MOV    @TEXTP, @XDBRA  ;LOAD DATA.
1353
1354  007450  062737  000002  001232        ADD    #2,     TEXTP   ;UPDATE TEXT POINTER.
```

```
1355  007456  000743                              BR      P15             ;LOOP BACK
1356
1357  007460  004737  007470          P19:        JSR     PC,RXMIT
1358  007464  000137  007366                      JMP     P15
1359  007470  162737  000004  001232  RXMIT:      SUB     #4,     TEXTP   ;POINT TO DATA TO BE RE-SENT.
1360  007476  104414                              SWCAL
1361  007500  032777  020000  171432              BIT     #BIT13, @SWR    ;INHIBIT ERROR MESSAGE?
1362  007506  001020                              BNE     P20             ;IF SO-SKIP TYPEOUT
1363
1364  007510  104401  007516                      TYPE    ,65$            ;;TYPE ASCIZ STRING
1365  007514  000412                              BR      64$             ;;GET OVER THE ASCIZ
1366                                      ;;65$:  .ASCIZ  <15><12>#RESENDING DATA: #
1367  007542                              64$:
1368  007542  017746  171464                      MOV     @TEXTP,-(SP)    ;;SAVE @TEXTP FOR TYPEOUT
1369  007546  104402                              TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1370
1371  007550  052777  000400  171416  P20:        BIS     #BIT8,  @XCSRA  ;SET CONTR. BIT TO ACKNOWLEDGE
1372                                                                      ;SLAVE'S CONTR. BIT. (SETS SLAVE'S
1373                                                                      ;STAT BIT.
1374
1375  007556  005777  171412          P21:        TST             @XCSRA          ;STAT BIT CLEAR? CLEAR WHEN
1376                                                                      ;SLAVE SEES MASTERS CONTR SET.
1377  007562  100775                              BMI     P21             ;NO THEN WAIT.
1378
1379  007564  042777  000400  171402              BIC     #BIT8,  @XCSRA  ;YES - CLEAR CONTROL (CLEARS
1380                                                                      ;SLAVE'S STAT BIT).
1381
1382  007572  000207                              RTS PC
1383
1384                                              .SBTTL  MASTER BURST MODE SEND
1385
1386
                ;*BURST MODE SENT

1392
1393  007574  104401  015267          BSM:        TYPE    ,MBM            ;TYPE "BURST SEND MODE"
1394
1395  007600  012737  001000  001256  P24:        MOV     #1000,  XNO     ;SET XMITT NO. TO 5
1396
1397  007606  023727  001256  000000  P25:        CMP     XNO,    #0      ;DONE 1000 TIMES?
1398  007614  001024                              BNE     P26             ;NO - CONTINUE.
1399
1400  007616  104414                              SWCAL
1401  007620  032777  040000  171312              BIT     #BIT14,@SWR
1402  007626  001362                              BNE     BSM
1403
1404  007630  104413                      P32:    EOPCAL
1405  007632  023727  001220  000115              CMP     MODE,#'M
1406  007640  001007                              BNE     2$
1407  007642  005337  001100                      DEC     $PASS
1408  007646  005037  007664                      CLR     3$
1409  007652  005237  007664          1$:         INC     3$
1410  007656  001375                              BNE     1$
```

```
1411  007660                          2$:
1412
1413  007660  000137  007304                JMP     MST
1414
1415  007664  000000                  3$:     .WORD   0
1416
1417
1418  007666  052777  000400  171300  P26:    BIS     #BIT8,  @XCSRA  ;SET CONTR. BIT. (SET SLAVE
1419                                                                  ;SEES MASTER'S CONTR AND SETS ITS
1420                                                                  ;CONTR BIT.
1421  007674  005777  171274          P27:    TST     @XCSRA          ;STAT IN BIT SET YET?
1422
1423  007700  100375                        BPL     P27             ;IF NOT SET - WAIT FOR IT.
1424
1425
1426  007702  042777  000400  171264  P28:    BIC     #BIT8,  @XCSRA  ;CLEAR CONTROL BIT (CLEARS
1427                                                                  ;SLAVE'S STAT BIT.
1428  007710  104412                        GETXC                   ;GET INTERRUPT VECTOR.
1429
1430  007712  012777  010016  171260        MOV     #ISR,   @XVT    ;SET UP INTERRUPT VECTOR.
1431  007720  012777  000340  171254        MOV     #340,   @XVT2   ;PRIORITY 7 ON INTERRUPT.
1432
1433  007726  042777  000001  171240        BIC     #BIT0,  @XCSRA  ;CLEAR ACCEPT.
1434  007734  012737  014270  001232        MOV     #TEXT,  TEXTP   ;POINT TO BEGINNING OF TEXT.
1435
1436  007742  017777  171264  171226  P29:  MOV     @TEXTP, @XDBRA  ;FORCE 1ST XFERR.
1437  007750  062737  000002  001232        ADD     #2,     TEXTP   ;POINT TO NEXT WORD ON INTR.
1438  007756  052777  000101  171210        BIS     #BIT6!BIT0,@XCSRA ;SET INTR ENABLE, DATA EXCEPT.
1439  007764  042737  000340  177776        BIC     #340,PS
1440  007772  000240                  P30:  NOP                     ;DATA REQUEST INTERRUPTS
1441                                                                  ;OCCUR HERE.
1442  007774  023727  001232  014644  P31:  CMP     TEXTP,  #TEXTE+2        ;TRANSFERRED WHOLE TEXT YET?
1443  010002  001373                        BNE     P30             ;NO GO BACK AND WAIT.
1444
1445  010004  162737  000001  001256        SUB     #1,     XNO     ;YES, XNO=XNO -1
1446  010012  000137  007606                JMP     P25             ;GO BACK TO MAIN LINE
1447


                ;*INTERRUPT SERVICE ROUTINE FOR BSM1
                ;*LOADS NEW DATA CHECKS FOR END OF TEXT.
                ;*IF END IS REACHED, CLEARS INTR. ENABLE.


1455
1456  010016  017777  171210  171152  ISR:    MOV     @TEXTP, @XDBRA  ;LOAD NEW DATA.
1457  010024  023727  001232  014642          CMP     TEXTP,  #TEXTE  ;XFERRED WHOLE TEXT?
1458  010032  001003                        BNE     1$              ;NO - EXIT
1459  010034  042777  000100  171132        BIC     #BIT6,  @XCSRA  ;YES - CLEAR INTR. ENABLE.
1460
1461  010042  062737  000002  001232  1$:   ADD     #2,TEXTP
1462  010050  000002                        RTI                     ;RETURN FROM INTR.
1463
1464                                        .SBTTL  INPUT MODULE'S CSR/DBR TESTS.
```

```
                        ;*INPUT MODULE'S CSR, DBR TEST
                        ;*ENTER WITH:   RCSRT=ADDR. OF CSR UNDER TEST
                        ;*              RDBRT=ADDR. OF DBR UNDER TEST

 1471
 1472  010052  013737  001204  001120  IMLT:   MOV     RCSRA,  $GDADR  ;SET CSR ADDR FOR ERROR TYPOUT
 1473
 1474                                   ;;*****************************************************************
 1475                                   ;*TEST 1        *TEST CSR ADDRESS ABILITY
 1476                                   ;;*****************************************************************
 1477  010060  000240                   TST1:   NOP
 1478  010062  012737  010070  001106           MOV     #3$,$LPADR      ;;SET SCOPE LOOP ADDRESS
 1479
 1480  010070  013746  000004           3$:     MOV     @#4,    -(SP)   ;SAVE CONTENTS OF LOCATION 4 ON STACK.
 1481  010074  012737  010110  000004           MOV     #1$,    @#4     ;SET LOC. IN CASE OF "TRAP" IF CSR DOES NOT RESPOND
 1482  010102  005777  171076                   TST     @RCSRA          ;ADDRESS CSR
 1483  010106  000402                           BR      2$              ;NO-TRAP IF HERE - NEXT TEST.
 1484
 1485  010110                           1$:                             ;TRAP HERE IS CSR DOES NOT RESPOND BY RETURNING SLAVE-SYNC.
 1486  010110  022626                           POPSP2                  ;RESET STACK POINTER (R6).
 1487  010112  104000                           ERROR                   ;NO RESPONSE FROM CSR UNDER TEST.
 1488                                                                   ;CSR ADDR. IN RCSRT.
 1489  010114  012637  000004           2$:     MOV     (SP)+,  @#4     ;RESTORE CONTENTS OF LOCATION 4 FROM STACK.
 1490
 1491                                   ;;*****************************************************************
 1492                                   ;*TEST 2        *TEST DBR ADDRESS ABILITY
 1493                                   ;;*****************************************************************
 1494  010120  000004                   TST2:   SCOPE
 1495
 1496  010122  013746  000004                   MOV     @#4,    -(SP)   ;SAVE CONTENTS OF LOCATION 4 ON STACK.
 1497  010126  012737  010142  000004           MOV     #1$,    @#4     ;SET LOC. 4 IN CASE OF TRAP IF DBR DOES NOT RESPOND
 1498  010134  005777  171046                   TST     @RDBRA          ;ADDRESS DBR
 1499  010140  000402                           BR      2$              ;NO TRAP IF HERE - NEXT TEST.
 1500
 1501  010142                           1$:                             ;TRAP HERE IF DBR DOES NOT RESPOND BY RETURNING SLAVE-SYNC.
 1502
 1503  010142  022626                           POPSP2                  ;RESET STACK POINTER (R6).
 1504  010144  104000                           ERROR                   ;NO RESPONSE FROM DBR UNDER TEST.
 1505
 1506  010146  012637  000004           2$:     MOV     (SP)+,  @#4     ;RESTORE CONTENTS OF LOC. 4 FROM STACK.
 1507
 1508                                   ;;*****************************************************************
 1509                                   ;*TEST 3        *TEST THAT CSR BIT 0  IS SET ON INIT
 1510                                   ;;*****************************************************************
 1511  010152  000004                   TST3:   SCOPE
 1512  010154  012737  000002  001166           MOV     #2,$TIMES       ;;DO 2 ITERATIONS
 1513
 1514  010162  005737  001100                   TST     $PASS           ;ONLY DO THIS TEST ON PASS 1.
 1515  010166  001014                           BNE     1$
 1516  010170  104417                           RESETC
 1517  010172  017737  171006  001126           MOV     @RCSRA, $BDDAT  ;READ CSR.
 1518  010200  042737  177776  001126           BIC     #177776,$BDDAT  ;ONLY WANT BIT 0 .
 1519  010206  022737  000001  001126           CMP     #1,     $BDDAT  ;IS IT SET?
 1520  010214  001401                           BEQ     1$              ;YES - NEXT TEST.
 1521
 1522  010216  104000                           ERROR                   ;CSR BIT 0  NOT SET ON INIT.
```

```
                                                        ;CSR DATA IN $BDDAT.
010220                                  1$:

                                        ;;*****************************************************************
                                        ;*TEST 4         *TEST THAT CSR BITS 6,7,AND 8 ARE CLEAR ON INIT.
                                        ;;*****************************************************************
010220  000004                          TST4:   SCOPE
010222  012737  000002  001166                  MOV     #2,$TIMES       ;;DO 2 ITERATIONS
010230  005737  001100                          TST     $PASS           ;ONLY DO THIS TEST ON FIRST PASS.
010234  001006                                  BNE     1$
010236  104417                                  RESETC
010240  032777  000700  170736                  BIT     #BIT6!BIT7!BIT8,@RCSRA  ;SEE IF BITS 6,7,8,AND 15 ARE CLEAR.
010246  001401                                  BEQ     1$

                                                        ;CSR BIT(S) 6,7,8, OR 15 NOT CLEAR ON
010250  104000                                  ERROR                   ;SYSTEM INITIALIZE.

010252                                  1$:

                                        ;;*****************************************************************
                                        ;*TEST 5         *TEST THAT CSR BITS 0,6, AND 8 CAN BE CLEARED
                                        ;;*****************************************************************
010252  000004                          TST5:   SCOPE
010254  012737  000005  001166                  MOV     #5,$TIMES       ;;DO 5 ITERATIONS

010262  005737  001100                          TST     $PASS
010266  001022                                  BNE     1$
010270  104417                                  RESETC                  ;SET BITS 0 AND 1.
010272  012737  000340  177776                  MOV     #340,   PS
010300  052777  000500  170676                  BIS     #BIT6!BIT8,@RCSRA  ;SET BITS 6 AND 8
010306  042777  000501  170670                  BIC     #BIT0!BIT6!BIT8,@RCSRA
010314  017737  170664  001126                  MOV     @RCSRA, $BDDAT  ;READ CSR.
010322  032737  000501  001126                  BIT     #BIT0!BIT6!BIT8,$BDDAT  ;SEE IF ANY ARE SET.
010330  001401                                  BEQ     1$              ;NO - CONTINUE.

010332  104000                                  ERROR                   ;CSR BIT(S) 0 AND/OR 1 AND/OR 6 AND/OR 8
                                                        ;COULD NOT BE BIT CLEAR. CSR RESULTS
                                                        ;IN $BDDAT.

010334  052777  000100  170602  1$:     BIS     #BIT6,@$TKS     ;RE-ENABLE TTY INTERRUPTS.
010342  005077  170636                  CLR     @RCSRA
010346  005037  177776                  CLR     PS

                                        ;;*****************************************************************
                                        ;*TEST 6         *TEST THAT CSR BITS 6, AND 8 CAN BE BIT SET
                                        ;;*****************************************************************
010352  000004                          TST6:   SCOPE

010354  012737  000340  177776                  MOV     #340,   PS
010362  042777  000500  170614                  BIC     #BIT6!BIT8,@RCSRA  ;MAKE SURE THEY ARE CLEAR
010370  052777  000500  170606                  BIS     #BIT6!BIT8,@RCSRA  ;THEN SET THEM.

010376  017737  170602  001126                  MOV     @RCSRA, $BDDAT  ;READ CSR.
010404  042737  177277  001126                  BIC     #177277,$BDDAT  ;STRIP AWAY OTHER BITS.

010412  023727  001126  000500                  CMP     $BDDAT,#BIT6!BIT8  ;DID THEY ARE SET?
```

```
1579  010420  001401                          BEQ     1$
1580
1581  010422  104000                          ERROR                   ;CSR BIT(S) 0 AND/OR 6 AND/OR 8
1582                                                                  ;DID NOT BIT SET. CSR RESULTS IN
1583                                                                  ;$BDDAT.
1584
1585  010424                          1$:
1586  010424  005077  170554                 CLR     @RCSRA
1587  010430  005037  177776                 CLR     PS
1588
1589
1590                                  ;;*****************************************************************
1591                                  ;*TEST 7         *TEST THAT BIT 7 WILL SET WHEN BIT 8 IS SET
1592                                  ;;*****************************************************************
1593  010434  000004                  TST7:   SCOPE
1594
1595  010436  042777  000400  170540          BIC     #BIT8,@RCSRA    ;MAKE SURE BIT 8 IS CLEAR.
1596  010444  005777  170536                 TST     @RDBRA          ;THEN MAKE SURE BIT 7 IS CLEAR.
1597  010450  052777  000400  170526          BIS     #BIT8,@RCSRA    ;NOW SET CSR BIT 8 SHOULD SET BIT 7
1598                                                                  ;VIA THE MAINTANCE LOOP BACK CABLE.
1599  010456  105777  170522                 TSTB            @RCSRA  ;DID BIT 7 SET?
1600  010462  100401                          BMI     1$              ;YES-NEXT TEST.
1601
1602  010464  104000                          ERROR           ;SETTING CSR BIT 8 FAILED TO SET CST BIT 7 VIA CABLE.
1603
1604  010466                          1$:
1605
1606                                  ;;*****************************************************************
1607                                  ;*TEST 10        *TEST LOW BYTE OPERATION OF INPUT CSR.
1608                                  ;;*****************************************************************
1609  010466  000004                  TST10:  SCOPE
1610
1611  010470  005077  170510                  CLR     @RCSRA          ;MAKE SURE CSR DAE,STAT OUT CLEAR.
1612  010474  112777  000401  170502          MOVB    #BIT8!BIT0,     @RCSRA  ;SET DAE-SHOULD NOT SET STAT OUT'
1613  010502  032777  000400  170474          BIT     #BIT08, @RCSRA  ;DID STAT OUT SET?
1614  010510  001401                          BEQ     1$              ;NO-CONTINUE.
1615
1616  010512  104000                          ERROR                   ;C0=1;C1=1;BUS ADDR. 00=0; BUT WROTE
1617                                                                  ;INTO HIGH BYTE OF MODULE CSR WORD.
1618
1619  010514  032777  000001  170462  1$:     BIT     #BIT00, @RCSRA  ;MAKE SURE WE DID SET BIT 0.
1620  010522  001001                          BNE     2$              ;YES-NEXT TEST.
1621
1622  010524  104000                          ERROR                   ;FAILED TO SET BIT 00 IN CSR ON A
1623                                                                  ;MOVE BYTE INST. LOW BYTE OPERATION.
1624  010526                          2$:
1625                                  ;;*****************************************************************
1626                                  ;*TEST 11        *TEST HIGH BYTE OPERATION OF CSR
1627                                  ;;*****************************************************************
1628  010526  000004                  TST11:  SCOPE
1629
1630  010530  005077  170450                  CLR     @RCSRA          ;MAKE SURE DAE AND STAT OUT =0.
1631  010534  013737  001204  001274          MOV     RCSRA,$TEMP0    ;SET BUS ADDR. BIT 00 OF ADDRESS.
1632  010542  005237  001274                  INC     $TEMP0
1633  010546  112777  000401  170520          MOVB    #BIT8!BIT0,@$TEMP0      ;SEND WHOLE WORD TO CSR ONLY STAT
1634                                                                  ;OUT SHOULD SET (HIGH BYTE OPERATION)
```

```
1635  010554  032777  000001  170422          BIT     #BIT00,@RCSRA   ;DID BIT 00 SET?
1636  010562  001401                          BEQ     1$              ;NO-CONTINUE.
1637
1638  010564  104000                          ERROR                   ;C0=1;C1=1;A00=1 BUT WROTE INTO
1639                                                                  ;LOW BYTE OF MODULE CSR WORD.
1640
1641  010566  032777  000400  170410  1$:     BIT     #BIT08, @RCSRA  ;MAKE SURE STAT OUT HAD SET.
1642  010574  001001                          BNE     2$              ;IF SO-NEXT TEST.
1643
1644  010576  104000                          ERROR                   ;FAILED TO SET BIT 08 IN CSR ON A
1645                                                                  ;MOVE BYTE INSTR. HIGH BYTE OPERATION.
1646  010600                          2$:
1647
1648
1649                                  ;;**************************************************************
1650                                  ;*TEST 12          NPUT MODULE INTERRUPT TEST
1651
1652                                  ;
1653                                  ;***>>>     WARNING     <<<***
1654                                  ; A MAINTANCE LOOP BACK CONNECTOR IS NEEDED FOR THIS
1655                                  ;TEST(S) OR THE TESTS WILL FAIL.
1656                                  ;***>>> END OF WARNING <<<***
1657                                  ;
1658                                  ;;**************************************************************
1659  010600  000004                  TST12:  SCOPE
1660  010602  012737  000001  001166          MOV     #1,$TIMES       ;;DO 1 ITERATION
1661
1662  010610  104411                          GETRC                   ;GET INTR. VECTOR + PRIORITY.
1663  010612  042777  000400  170364          BIC     #BIT8,@RCSRA
1664  010620  005777  170362                  TST     @RDBRA          ;RESET ALL CONDITIONS IN INPUT + OUTPUT CSR'S.
1665  010624  013700  001210                  MOV     RVT,    R0      ;GET VECTOR ADDR. OF RECEIVER.
1666  010630  012737  000340  177776          MOV     #340,   PS      ;LOCK OUT INTERRUPTS.
1667
1668  010636  012710  010676                  MOV     #1$,    (0)     ;SET UP INTERRUPT VECTOR.
1669  010642  016037  000002  001274          MOV     2(R0),  $TEMP0  ;SAVE CONTENTS OF VECTOR X2.
1670  010650  012760  000340  000002          MOV     #340,   2(R0)   ;SET PRIORITY ON INTERRUPT.
1671
1672  010656  052777  000500  170320          BIS     #BIT6!BIT8,@RCSRA  ;SET INTERRUPT ENABLE, AND CONTROL BIT.
1673  010664  005037  177776                  CLR     PS              ;ALLOW INTERRUPTS.
1674
1675  010670  000240                          NOP                     ;SHOULD INTERRUPT FROM HERE.
1676
1677  010672  104000                          ERROR                   ;NO - INTERRUPT FROM INPUT MODULE.
1678
1679  010674  000406                          BR      2$
1680
1681                                          ;MODULE SHOULD INTERRUPT TO HERE.
1682
1683  010676  022626                  1$:     POPSP2                  ;RESET STACK PRINTER (R6)
1684  010700  042777  000500  170276          BIC     #BIT6!BIT8,@RCSRA  ;CLEAR INTERRUPT ENABLE.
1685  010706  005037  177776                  CLR     PS              ;ALLOW INTERRUPTS.
1686  010712  005777  170270          2$:     TST     @RDBRA          ;RESET ALL CSR BITS.
1687
1688                                  ;;**************************************************************
1689                                  ;*TEST 13          *THAT THE INPUT MODULE WON'T INTERRUPT IF CPU IS AT SAME PRIORITY
1690
```

```
                                        ;
                                        ;***>>>     WARNING     <<<***
                                        ; A MAINTANCE LOOP BACK CONNECTOR IS NEEDED FOR THIS
                                        ;TEST(S) OR THE TESTS WILL FAIL.
                                        ;***>>> END OF WARNING <<<***
                                        ;
                                        ;;*****************************************************************
010716 000004                           TST13:  SCOPE
010720 012737 000001 001166                     MOV     #1,$TIMES       ;;DO 1 ITERATION
010726 005777 170254                            TST     @RDBRA          ;RESET ALL CSR BITS
010732 104411                                   GETRC                   ;GET INTR. VECTOR + PRIORITY
010734 013700 001210                            MOV     RVT,    R0      ;GET VECTOR ADDR. OF INPUT MOD.
010740 012737 000340 177776                     MOV     #340,   PS      ;LOCK OUT ALL INTRS.
010746 012710 011012                            MOV     #1$,    (0)     ;SET UP INTR. VECTOR FOR INTR.
010752 012760 000340 000002                     MOV     #340,   2(0)    ;SET PRIORITY ON INTR.

010760 052777 000500 170216                     BIS     #BIT6!BIT8,@RCSRA  ;SET INTERRUPT ENABLE.
010766 013737 001266 177776                     MOV     RVTP,   PS      ;SET PROCESSOR PRIORITY TO THAT OF
                                                                        ;INPUT MODULE.
010774 000240                                   NOP                     ;WILL INTERRUPT FROM HERE IF ANY.

010776 042777 000500 170200                     BIC     #BIT6!BIT8,@RCSRA  ;NO INTR. CLEAR INTR. ENABLE.
011004 005037 177776                            CLR     PS              ;ALLOW INTERRUPTS
011010 000407                                   BR      2$

                                                ;MODULE WILL INTERRUPT TO HERE IF ANY.

011012 022626                           1$:     POPSP2                  ;RESET STACK POINTER (R6).
011014 042777 000500 170162                     BIC     #BIT6!BIT8,@RCSRA  ;CLEAR INTERRUPT ENABLE.
011022 005037 177776                            CLR     PS              ;ALLOW INTERRUPTS.

011026 104000                                   ERROR                   ;REPORT ERROR - INPUT MODULE.
                                                                        ;INTERRUPTED WHEN CPU WAS SET AT
                                                                        ;SAME PRIORITY AS IT WAS.
011030 010001                           2$:     MOV     R0,R1
011032 005721                                   TST     (1)+
011034 010110                                   MOV     R1,(R0)
011036 013760 001164 000002                     MOV     $TMP0,  2(R0)   ;RESTORE VECTOR +2.

011044 104413                                   EOPCAL

011046 000137 010052                    3$:     JMP     IMLT

                                                .SBTTL  OMLT OUTPUT MODULE CSR/DBR TEST.


                ;*OUTPUT MODULES CSR, DBR TEST.
                ;*ENTER WITH:    XCSRA=ADDR. OF CSR UNDER TEST
                ;*               XDBRA=ADDR. OF DBR UNDER TEST


011052 013737 001174 001120     OMLT:   MOV     XCSRA,$GDADR    ;SET CSR ADDR. FOR ERROR TYPOUT
```

```
                                ;;*****************************************************************
                                ;*TEST 14       *TEST CSR ADDRESSABILITY
                                ;;*****************************************************************
011060  000240                  TST14:  NOP
011062  012737  011070  001106          MOV     #3$,$LPADR      ;;SET SCOPE LOOP ADDRESS

011070  013746  000004          3$:     MOV     @#4,    -(SP)   ;SAVE CONTENTS OF LOC. 4 ON STACK
011074  012737  011110  000004          MOV     #1$,    @#4     ;SET LOC. 4 IN CASE OF TRAP IF CSR DOESN'T RESPOND.
011102  005777  170066                  TST     @XCSRA          ;ADDRESS CSR.
011106  000402                          BR      2$              ;NO TRAP IF HERE - NEXT TEST.

011110                          1$:             ;TRAP HERE IF CSR DOES NOT RESPOND BY RETURNING SLAVE-SYNC.
011110  022626                          POPSP2                  ;RESET STACK POINTER (R6).
011112  104000                          ERROR                   ;NO RESPONSE FROM CSR UNDER TEST.

011114  012637  000004          2$:     MOV     (SP)+,  @#4     ;RESTORE CONTENTS OF LOC 4 FROM STACK.

                                ;;*****************************************************************
                                ;*TEST 15       *TEST DBR ADDRESSABILITY
                                ;;*****************************************************************
011120  000004                  TST15:  SCOPE

011122  013746  000004                  MOV     @#4,    -(SP)   ;SAVE CONTENTS OF LOCATION 4 ON STACK
011126  012737  011142  000004          MOV     #1$,    @#4     ;SET LOC. 4 IN CASE OF TRAP IF DBR DOESN'T RESPOND.
011134  005777  170036                  TST     @XDBRA          ;ADDRESS DBR.
011140  000402                          BR      2$              ;NO TRAP IF HERE - NEXT TEST.

011142                          1$:             ;TRAP HERE IF DBR DOESN'T RESPOND BY RETURNING SLAVE-SYNC.

011142  022626                          POPSP2                  ;RESET STACK POINTER (R6).
011144  104000                          ERROR                   ;NO RESPONSE FROM DBR UNDER TEST.

011146  012637  000004          2$:     MOV     (SP)+,  @#4     ;RESTOER CONTENTS OF LOCATION 4 FROM STACK.

                                ;;*****************************************************************
                                ;*TEST 16       *TEST THAT XMITTER CSR BIT 0 IS SET ON INIT
                                ;;*****************************************************************
011152  000004                  TST16:  SCOPE
011154  012737  000002  001166          MOV     #2,$TIMES       ;;DO 2 ITERATIONS
011162  005737  001100                  TST     $PASS           ;ONLY DO THIS TEST ON FIRST PASS.
011166  001011                          BNE     1$

011170  042777  000001  167776          BIC     #BIT0,  @XCSRA  ;MAKE SURE CSR BIT 0 IS CLEAR.

011176  104417                          RESETC                  ;ISSUE SYSTEM INITIALIZE, THIS
                                                                ;SHOULD CAUSE BIT 0 TO SET.
011200  032777  000001  167766          BIT     #BIT0,  @XCSRA  ;IS CSR BIT 0 SET?
011206  001001                          BNE     1$              ;YES - NEXT TEST.

011210  104000                          ERROR                   ;OUTPUT MODULES CSR BIT 0 FAILED TO
                                                                ;SET ON INIT.

011212                          1$:

                                ;;*****************************************************************
                                ;*TEST 17       *TEST THAT OUTPUT MOD. CSR BITS 1,6 AND 8 ARE CLEAR ON INIT.
```

```
1803                                    ;;*****************************************************************
1804  011212  000004                    TST17:  SCOPE
1805  011214  012737  000002  001166            MOV     #2,$TIMES       ;;DO 2 ITERATIONS
1806  011222  005737  001100                    TST     $PASS           ;ONLY DO THIS TEST FIRST PASS.
1807  011226  001017                            BNE     1$
1808
1809  011230  012737  000340  177776            MOV     #340,PS
1810  011236  052777  000500  167730            BIS     #BIT6!BIT8,@XCSRA  ;PRESET CSR BITS 6 AND 8.
1811
1812  011244  104417                            RESETC                  ;SYSTEM INITIALIZE, SHOULD CLEAR
1813                                                                    ;BITS 1,6, AND 8
1814  011246  032777  000502  167720            BIT     #BIT1!BIT6!BIT8,@XCSRA  ;ARE CSR BITS 1,6, AND 8 CLEAR?
1815  011254  001404                            BEQ     1$              ;YES - CONTINUE.
1816  011256  017737  167712  001126            MOV     @XCSRA,$BDDAT   ;NO - STORE CSR FOR OPERATOR EXAMINATION.
1817
1818  011264  104000                            ERROR                   ;OUTPUT MODULES CSR BIT 1 AND/OR BIT 6-
1819                                                                    ;AND/OR BIT 8 FAILED TO CLEAR ON INIT.
1820
1821  011266                          1$:
1822  011266  005077  167702                    CLR     @XCSRA
1823  011272  005037  177776                    CLR     PS
1824
1825                                    ;;*****************************************************************
1826                                    ;*TEST 20       *TEST THAT DBR IS ZERO ON INIT
1827                                    ;;*****************************************************************
1828  011276  000004                    TST20:  SCOPE
1829  011300  012737  000001  001166            MOV     #1,$TIMES       ;;DO 1 ITERATION
1830
1831  011306  005737  001100                    TST     $PASS           ;ONLY DO THIS TEST FIRST PASS.
1832  011312  001006                            BNE     1$
1833  011314  104417                            RESETC                  ;ISSUE SYSTEM INITIALIZE.
1834
1835  011316  017737  167654  001126            MOV     @XDBRA,$BDDAT   ;READ DBR.
1836  011324  001401                            BEQ     1$              ;IF ZERO - NEXT TEST
1837
1838  011326  104000                            ERROR                   ;DBR NOT ZERO AFTER INITIALIZE
1839                                                                    ;DBR ADDR. IN XDBRA.
1840
1841  011330                          1$:
1842
1843                                    ;;*****************************************************************
1844                                    ;*TEST 21       *TEST THAT CSR BITS 0,1,6, AND 8 CAN BE BIT CLEARED
1845                                    ;;*****************************************************************
1846  011330  000004                    TST21:  SCOPE
1847  011332  012737  000005  001166            MOV     #5,$TIMES       ;;DO 5 ITERATIONS
1848
1849  011340  005737  001100                    TST     $PASS           ;ONLY DO THIS TEST FIRST PASS.
1850  011344  001022                            BNE     1$
1851  011346  104417                            RESETC                  ;SET BITS 0 AND 1.
1852  011350  012737  000340  177776            MOV     #340,PS
1853  011356  052777  000500  167610            BIS     #BIT6!BIT8,@XCSRA  ;SET BITS 6 AND 8
1854  011364  042777  000503  167602            BIC     #BIT0!BIT1!BIT6!BIT8,@XCSRA     ;NOW CLEAR THE BITS.
1855  011372  017737  167576  001126            MOV     @XCSRA,$BDDAT   ;READ CSR.
1856  011400  032737  000503  001126            BIT     #BIT0!BIT1!BIT6!BIT8,$BDDAT  ;SEE IF ANY ARE SET.
1857  011406  001401                            BEQ     1$              ;NO - CONTINUE.
1858
```

```
1858  011410  104000                          ERROR                   ;CSR BIT(S) 0 AND/OR 1 AND/OR6 AND/OR 8
1860                                                                  ;COULD NOT BE BIT CLEAR. CSR RESULTS
1861                                                                  ;IN $BDDAT.
1862  011412  052777  000100  167524  1$:     BIS     #BIT6,@$TKS     ;RE-ENABLE TTY INTERRUPTS.
1863  011420  005077  167550                  CLR     @XCSRA
1864  011424  005037  177776                  CLR     PS
1865
1866                                  ;;**********************************************************************
1867                                  ;*TEST 22       *TEST THAT CSR BITS 0,6, AND 8 CAN BE BIT SET
1868                                  ;;**********************************************************************
1869  011430  000004                  TST22:  SCOPE
1870
1871  011432  012737  000340  177776          MOV     #340,PS
1872  011440  042777  000501  167526          BIC     #BIT0!BIT6!BIT8,@XCSRA  ;MAKE SURE THEY ARE CLEAR
1873  011446  052777  000501  167520          BIS     #BIT0!BIT6!BIT8,@XCSRA  ;THEN SET THEM.
1874
1875  011454  017737  167514  001126          MOV     @XCSRA,$BDDAT   ;READ CSR.
1876  011462  042737  177274  001126          BIC     #177274,$BDDAT  ;STRIP AWAY OTHER BITS.
1877
1878  011470  023727  001126  000501          CMP     $BDDAT,#BIT0!BIT6!BIT8  ;DID THEY ARE SET?
1879  011476  001401                          BEQ     1$
1880
1881  011500  104000                          ERROR                   ;CSR BIT(S) 0 AND/OR 6 AND/OR 8
1882                                                                  ;DID NOT BIT SET. CSR RESULTS IN
1883                                                                  ;$BDDAT
1884
1885  011502                          1$:
1886  011502  005077  167466                  CLR     @XCSRA
1887  011506  005037  177776                  CLR     PS
1888
1889                                  ;;**********************************************************************
1890                                  ;*TEST 23       *TEST THAT CSR BIT 7 WILL SET WHEN BIT 8 IS SET
1891
1892                                  ;
1893                                  ;***>>>    WARNING     <<<***
1894                                  ; A MAINTANCE LOOP BACK CONNECTOR IS NEEDED FOR THIS
1895                                  ;TEST(S) OR THE TESTS WILL FAIL.
1896                                  ;***>>> END OF WARNING <<<***
1897                                  ;
1898                                  ;;**********************************************************************
1899  011512  000004                  TST23:  SCOPE
1900
1901  011514  005077  167454                  CLR     @XCSRA          ;CLEAR CSR BITS.
1902  011520  052777  000400  167446          BIS     #BIT8,@XCSRA    ;SET CSR BIT8. SHOULD SET CSR BIT 7
1903  011526  105777  167442                  TSTB    @XCSRA  ;DID BIT7 SET?
1904  011532  001001                          BNE     1$              ;YES - NEXT TEST.
1905
1906  011534  104000                          ERROR                   ;CSR BIT 7 DID NOT SET WHEN
1907                                                                  ;CSR BIT 8 WAS SET.
1908
1909                                  ;
1910                                  ;***>>>    WARNING     <<<***
1911                                  ; A MAINTANCE LOOP BACK CONNECTOR IS NEEDED FOR THIS
1912                                  ;TEST(S) OR THE TESTS WILL FAIL.
1913                                  ;***>>> END OF WARNING <<<***
1914                                  ;
```

```
1915
1916  011536                                1$:
1917
1918                                        ;;*****************************************************************
1919                                        ;*TEST 24       *TEST LOW BYTE OPERATION OF OUTPUT MODS. CSR
1920                                        ;;*****************************************************************
1921  011536  000004                        TST24:  SCOPE
1922
1923  011540  005077  167430                        CLR     @XCSRA          ;MAKE SURE BIT 00 AND BIT 08 ARE CLEAR.
1924  011544  112777  000401  167422        1$:     MOVB    #BIT8!BIT0,@XCSRA       ;SEND ALL ONES TO CSR-ONLY LOW BYTE
1925                                                                        ;SHOULD GET THROUGH.
1926  011552  032777  000400  167414              BIT     #BIT08,@XCSRA   ;DID BIT 08 GET SET?(ITS IN HIGH BYTE).
1927  011560  001401                              BEQ     1$              ;NO-CONTINUE.
1928
1929  011562  104000                              ERROR                   ;C0=1;C1=1;BUS A00=0 BUT WE WROTE INTO
1930                                                                        ;HIGH BYTE OF MODULES CSR WORD.
1931
1932  011564  032777  000001  167402  1$:     BIT     #BIT00, @XCSRA  ;MAKE SURE WE SET BIT 00.
1933  011572  001001                              BNE     2$              ;IF SO-NEXT TEST.
1934
1935  011574  104000                              ERROR                   ;FAILED TO SET BIT 00 IN CSR ON A
1936                                                                        ;LOW BYTE MOV OPERATION.
1937
1938  011576                                2$:
1939
1940                                        ;;*****************************************************************
1941                                        ;*TEST 25       *TEST HIGH BYTE OPERATION OF OUTPUT CSR.
1942                                        ;;*****************************************************************
1943  011576  000004                        TST25:  SCOPE
1944
1945  011600  005077  167370                        CLR     @XCSRA          ;MAKE SURE CSR BITS 00 AND 08 ARE CLEAR.
1946  011604  013737  001174  001274              MOV     XCSRA,$TEMP0    ;NOW FORM HIGH BYTE ADDRESS SET BUS-
1947  011612  005237  001274                      INC     $TEMP0          ;ADDRESS BIT 00.
1948  011616  112777  000401  167450              MOVB    #BIT8!BIT0,@$TEMP0      ;WRITE WHOLE WORD OUT MODULE SHOULD DECODE-
1949                                                                        ;ONLY WANTING TO WRITE INTO HIGH BYTE.
1950  011624  032777  000001  167342              BIT     #BIT00, @XCSRA  ;DID WE SET A BIT IN THE LOW WORD?
1951  011632  001401                              BEQ     1$              ;NO-CONTINUE.
1952
1953  011634  104000                              ERROR                   ;C0=1;C1=1;A00=1 BUT WORTE INTO LOW
1954                                                                        ;BYTE OF MODULES CSR WORD.
1955
1956  011636  032777  000400  167330  1$:     BIT     #BIT08, @XCSRA  ;MAKE SURE WE HAD SET BIT 08 IN HIGH BYTE.
1957  011644  001001                              BNE     2$
1958
1959  011646  104000                              ERROR                   ;FAILED TO SET BIT 08 IN CSR ON A
1960                                                                        ;MOV BYTE HIGH BYTE OPERATION.
1961
1962  011650                                2$:
1963
1964                                        ;;*****************************************************************
1965                                        ;*TEST 26       *TEST THE LOW BYTE OPERATION OF THE OUTPUT MODS. DBR.
1966                                        ;;*****************************************************************
1967  011650  000004                        TST26:  SCOPE
1968
1969  011652  005077  167320                        CLR     @XDBRA          ;MAKE SURE DBR IS CLEAR.
1970  011656  112777  177777  167312              MOVB    #-1,@XDBRA      ;SEND ALL ONES TO DBR--BUT SENCE THIS IS
```

```
1971
1972  011664  032777  000400  167304          BIT     #BIT08,@XDBRA   ;A BYTE INSTR. ONLY THE LOW BYTE SHOLD BE WRITTEN.
1973  011672  001401                          BEQ     1$              ;DID HIGH BYTE GET WRITTEN INTO?
1974                                                                  ;NO-CONTINUE.
1975  011674  104000                          ERROR                   ;HIGH BYTE OF DBR GOT WRITTEN INTO
1976
1977                                                                  ;ON A MOV TO LOW BYTE OPERATION.
1978  011676  032777  000001  167272  1$:     BIT     #BIT00, @XDBRA  ;DID LOW BYTE GET WRITTEN?
1979  011704  001001                          BNE     2$              ;YES-NEXT TEST.
1980
1981  011706  104000                          ERROR                   ;LOW BYTE OF DBR FAILED TO GET WRITTEN
1982                                                                  ;INTO ON A LOW BYTE DATAOB INSTR.
1983  011710                          2$:
1984
1985                                  ;;**************************************************************
1986                                  ;*TEST 27        *TEST THE HIGH BYTE OPERATION OF THE OUTPUT MODS. DBR.
1987                                  ;;**************************************************************
1988  011710  000004                  TST27:  SCOPE
1989
1990  011712  005077  167260                  CLR     @XDBRA          ;MAKE SURE DBR IS CLEAR.
1991  011716  013737  001176  001274          MOV     XDBRA,$TEMPO    ;NOW FORM HIGH BYTE ADDRESS OF DBR.
1992  011724  005237  001274                  INC     $TEMPO          ;BY SETTING BUS ADDR. BIT 00.
1993  011730  112777  177777  167336          MOVB    #-1,@$TEMPO     ;WEN WHOLE WORD TO DBR ONLY HIGH BYTE
1994                                                                  ;OF DBR SHOULD BE WRITTEN INTO.
1995  011736  032777  000001  167232          BIT     #BIT00,@XDBRA   ;DID LOW BYTE BE WRITTEN INTO?
1996
1997  011744  001401                          BEQ     1$              ;NO-CONTINUE.
1998
1999  011746  104000                          ERROR                   ;ERROR-WROTE INTO LOW BYTE OF DBR
2000                                                                  ;ON A MOV HIGH BYTE OPERATION.
2001                                                                  ;C0=1;C1=1;A00=1.
2002
2003  011750  032777  000400  167220  1$:     BIT     #BIT08,@XDBRA   ;MAKE SURE HIGH BYTE GOT WRITTEN INTO.
2004  011756  001001                          BNE     2$              ;YES-NEXT TEST.
2005
2006  011760  104000                          ERROR                   ;FAILED TO WRITE INTO HIGH BYTE
2007                                                                  ;OF DBR IN A MOV HIGH BYTE WRITE OPERATION.
2008
2009  011762                          2$:
2010                                  ;;**************************************************************
2011                                  ;*TEST 30        *OUTPUT MODULE INTERRUPT TEST
2012
2013                                  ;
2014                                  ;***>>>    WARNING     <<<***
2015                                  ; A MAINTANCE LOOP BACK CONNECTOR IS NEEDED FOR THIS
2016                                  ;TEST(S) OR THE TESTS WILL FAIL.
2017                                  ;***>>> END OF WARNING <<<***
2018                                  ;
2019                                  ;;**************************************************************
2020  011762  000004                  TST30:  SCOPE
2021  011764  012737  000001  001166          MOV     #1,$TIMES       ;;DO 1 ITERATION
2022
2023  011772  104412                          GETXC                   ;GET INTR. VECTOR + PRIORITY.
2024  011774  042777  000400  167172          BIC     #BIT8,@XCSRA
2025  012002  005777  167170                  TST     @XDBRA          ;RESET ALL CONDITIONS IN OUTPUT CSR.
2026  012006  013700  001200                  MOV     XVT,    R0      ;GET VECTOR ADDR. OF RECEIVER.
```

```
2027  012012  012737  000340  177776          MOV     #340,   PS      ;LOCK OUT INTERRUPTS.
2028
2029  012020  012710  012060                  MOV     #1$,    (0)     ;SET UP INTERRUPT VECTOR.
2030  012024  016037  000002  001164          MOV     2(R0),  $TMP0   ;SAVE CONTENTS OF VECTOR X2.
2031  012032  012760  000340  000002          MOV     #340,   2(R0)   ;SET PRIORITY ON INTERRUPT.
2032
2033  012040  052777  000500  167126          BIS     #BIT6!BIT8,@XCSRA  ;SET INTERRUPT ENABLE, AND CONTROL BIT.
2034  012046  005037  177776                  CLR     PS              ;ALLOW INTERRUPTS.
2035
2036  012052  000240                          NOP                     ;SHOULD INTERRUPT FROM HERE.
2037
2038  012054  104000                          ERROR                   ;NO - INTERRUPT FROM OUTPUT MODULE.
2039
2040  012056  000406                          BR      2$
2041
2042                                          ;MODULE SHOULD INTERRUPT TO HERE.
2043
2044  012060  022626                  1$:     POPSP2                  ;RESET STACK PRINTER (R6).
2045  012062  042777  000500  167104          BIC     #BIT6!BIT8,@XCSRA  ;CLEAR INTERRUPT ENABLE.
2046  012070  005037  177776                  CLR     PS              ;ALLOW INTERRUPTS.
2047
2048  012074                          2$:
2049
2050                                  ;;**********************************************************************
2051                                  ;*TEST 31        *THAT THE OUTPUT MODULE WON'T INTERRUPT IF CPU IS AT SAME PRIORITY
2052
2053                                  ;
2054                                  ;***>>>     WARNING     <<<***
2055                                  ; A MAINTANCE LOOP BACK CONNECTOR IS NEEDED FOR THIS
2056                                  ;TEST(S) OR THE TESTS WILL FAIL.
2057                                  ;***>>> END OF WARNING <<<***
2058                                  ;
2059                                  ;;**********************************************************************
2060  012074  000004                  TST31:  SCOPE
2061  012076  012737  000001  001166          MOV     #1,$TIMES       ;;DO 1 ITERATION
2062
2063  012104  005777  167066                  TST     @XDBRA          ;RESET ALL CSR BITS.
2064  012110  104412                          GETXC                   ;GET INTR. VECTOR + PRIORITY
2065  012112  013700  001200                  MOV     XVT,    R0      ;GET VECTOR ADDR. OF OUTPUT MOD.
2066  012116  012737  000340  177776          MOV     #340,   PS      ;LOCK OUT ALL INTRS.
2067  012124  012710  012170                  MOV     #1$,    (0)     ;SET UP INTR. VECTOR FOR INTR.
2068  012130  012760  000340  000002          MOV     #340,   2(0)    ;SET PRIORITY ON INTR.
2069
2070  012136  052777  000500  167030          BIS     #BIT6!BIT8,@XCSRA  ;SET INTERRUPT ENABLE.
2071  012144  013737  001264  177776          MOV     XVTP,   PS      ;SET PROCESSOR PRIORITY TO THT OF
2072                                                                  ;INPUT MODULE.
2073  012152  000240                          NOP                     ;WILL INTERRUPT FROM HERE IF ANY.
2074
2075  012154  042777  000500  167012          BIC     #BIT6!BIT8,@XCSRA  ;NO INTR. CLEAR INTR. ENABLE.
2076  012162  005037  177776                  CLR     PS              ;ALLOW INTERRUPTS.
2077  012166  000407                          BR      2$
2078
2079                                          ;MODULE WILL INTERRUPT TO HERE IF ANY.
2080
2081  012170  022626                  1$:     POPSP2                  ;RESET STACK POINTER (R6).
2082  012172  042777  000500  166774          BIC     #BIT6!BIT8,@XCSRA  ;CLEAR INTERRUPT ENABLE.
```

```
2083  012200  005037  177776                    CLR     PS              ;ALLOW INTERRUPTS.
2084
2085  012204  104000                            ERROR                   ;REPORT ERROR - OUTPUT MODULE.
2086                                                                    ;INTERRUPTED WHEN CPU WAS SET AT
2087                                                                    ;SAME PRIORITY AS IT WAS.
2088  012206  010001                  2$:       MOV     R0,R1
2089  012210  005721                            TST     (R1)+
2090  012212  010110                            MOV     R1,(R0)
2091  012214  013760  001164  000002            MOV     $TMP0,  2(R0)   ;RESTORE VECTOR +2.
2092  012222  104417                            RESETC                  ;CLEAR WORLD.
2093
2094  012224  104413                            EOPCAL
2095  012226  000137  011052                    JMP     OMLT
2096

                ;*XCVP AND RCVP - ROUTINES USE TO READ AND STORE
                ;*INPUT AND OUTPUT MODULES VECTOR + PRIORITIES.
                ;*CALL= JSR     PC,XCVP FOR OUTPUT MODULE'S CSR.
                ;*                      REQUIRED: XCSRA
                ;*                                XVT
                ;*                                XVTP
                ;*      CALL=   JSR  PC,RCVP FOR INPUT MODULES CSR.
                ;*              REQUIRED: RCSRA
                ;*                        RVT
                ;*                        RVTP

2109
2110  012232  017737  166736  001124  XCVP:   MOV     @XCSRA,$GDDAT   ;READ XMITTER CSR FOR PRIORITY
2111  012240  042737  177717  001124          BIC     #177717,$GDDAT  ;PRIORITY IN BITS 4+5
2112  012246  013701  001124                  MOV     $GDDAT,R1       ;PSW IN R1
2113  012252  006237  001124                  ASR     $GDDAT          ;STATE5 STATE4  PRIORITY
2114  012256  006237  001124                  ASR     $GDDAT          ;   0       0       4
2115  012262  006237  001124                  ASR     $GDDAT          ;   0       1       5
2116  012266  006237  001124                  ASR     $GDDAT          ;   1       0       6
2117  012272  006301                          ASL     R1              ;   1       1       7
2118  012274  052701  000200                  BIS     #BIT7,R1        ;R0 TO KEEP COUNT.
2119  012300  017737  166670  001126          MOV     @XCSRA,$BDDAT   ;READ XMITTER'S CSR FOR VECTOR.
2120  012306  042737  100777  001126          BIC     #100777,$BDDAT  ;VECTOR BITS 9-14 IN CSR.
2121  012314  000337  001126                  SWAB    $BDDAT          ;CSR BITS 9-14 REPRESENT ADDR BITS 03-08.
2122  012320  006337  001126                  ASL     $BDDAT
2123  012324  006337  001126                  ASL     $BDDAT
2124  012330  017700  166640                  MOV     @XCSRA,R0       ;READ CSR AGAIN TO FIND OUT
2125  012334  042700  177773                  BIC     #177773,R0      ;THE STATE OF BIT 2.
2126  012340  060037  001126                  ADD     R0,$BDDAT       ;BIT 2 IS PART OF VECTOR ADDR.
2127  012344  013737  001174  001120          MOV     XCSRA,$GDADR;SET ADDR FOR TYPEOUT
2128  012352  020137  001264                  CMP     R1,XVTP         ;PRIORITY THE SAME?
2129  012356  001415                          BEQ     3$              ;YES -3$
2130  012360  010137  001264          2$:     MOV     R1,XVTP         ;NO - STORE NEW PRIORITY
2131  012364  013737  001126  001200          MOV     $BDDAT,XVT      ;STORE VECTOR ADDR.
2132  012372  013737  001200  001202          MOV     XVT,XVT2
2133  012400  062737  000002  001202          ADD     #2,XVT2
2134  012406  000137  012636                  JMP     XRCVPT          ;GO TYPEOUT CHANGE.
2135  012412  023737  001126  001200  3$:     CMP     $BDDAT,XVT      ;IS THE VECTOR THE SAME?
2136  012420  001357                          BNE     2$              ;NO - GO TYPE IT OUT.
2137  012422  023727  001200  000100          CMP     XVT,#100
2138  012430  002753                          BLT     2$
```

```
2139  012432  000002                          RTI
2140
2141  012434  017737  166544  001124  RCVP:   MOV     @RCSRA,$GDDAT   ;READ RECEIVER CSR FOR PRIORITY
2142  012442  042737  177717  001124          BIC     #177717,$GDDAT  ;PRIORITY IN BITS 4+5
2143  012450  013701  001124                  MOV     $GDDAT,R1
2144  012454  006237  001124                  ASR     $GDDAT          ;STATES STATE4  PRIORITY
2145  012460  006237  001124                  ASR     $GDDAT          ;  0      0        4
2146  012464  006237  001124                  ASR     $GDDAT          ;  0      1        6
2147  012470  006237  001124                  ASR     $GDDAT          ;  1      0        6
2148  012474  006301                          ASL     R1              ;  1      1        7
2149  012476  052701  000200                  BIS     #BIT7,R1        ;R0 TO KEEP COUNT.
2150  012502  017737  166476  001126          MOV     @RCSRA,$BDDAT   ;READ RECEIVER CSR FOR VECTOR.
2151  012510  042737  100777  001126          BIC     #100777,$BDDAT  ;VECTOR BITS 9-14 IN CSR.
2152  012516  000337  001126                  SWAB    $BDDAT          ;CSR BITS 9-14 REPRESENT ADD BITS 03-08
2153  012522  006337  001126                  ASL     $BDDAT
2154  012526  006337  001126                  ASL     $BDDAT
2155  012532  017700  166446                  MOV     @RCSRA,R0       ;READ CSR AGAIN TO FIND OUT
2156  012536  042700  177773                  BIC     #177773,R0      ;THE STATE OF BIT 2.
2157  012542  060037  001126                  ADD     R0,$BDDAT       ;BIT 2 IS PART OF VECTOR ADDR.
2158  012546  013737  001204  001120          MOV     RCSRA,$GDADR    ;SET ADDR. FOR TYPEOUT.
2159
2160  012554  020137  001266                  CMP     R1,RVTP         ;PRIORITY THE SAME?
2161  012560  001415                          BEQ     3$              ;YES -3$
2162  012562  010137  001266          2$:     MOV     R1,RVTP         ;NO-STORE NEW PRIORITY.
2163  012566  013737  001126  001210          MOV     $BDDAT,RVT      ;STORE VECTOR ADDR.
2164  012574  013737  001210  001212          MOV     RVT,RVT2
2165  012602  062737  000002  001212          ADD     #2,RVT2
2166  012610  000137  012636                  JMP     XRCVPT          ;GO TYPEOUT CHANGE.
2167  012614  023737  001126  001210  3$:     CMP     $BDDAT,RVT      ;DID VECTOR ADDR. CHANGE?
2168  012622  001357                          BNE     2$              ;YES-TYPE IT OUT.
2169  012624  023727  001210  000100          CMP     RVT,#100
2170  012632  002753                          BLT     2$
2171  012634  000002                          RTI
2172
2173                                          ;ROUTINE TO TYPEOUT A VECTOR OR PRIORITY CHANGE
2174
2175  012636                          XRCVPT:
2176  012636  104401  012644                  TYPE    ,65$            ;;TYPE ASCIZ STRING
2177  012642  000407                          BR      64$             ;;GET OVER THE ASCIZ
2178                                  ;;65$:  .ASCIZ  <15><12>#CSR ADDR: #
2179  012662                          64$:
2180
2181  012662  013746  001120                  MOV     $GDADR,-(SP)    ;;SAVE $GDADR FOR TYPEOUT
2182                                                                  ;;TYPES CSR ADDRESS OF DEVICE
2183  012666  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2184  012670  104401  012676                  TYPE    ,67$            ;;TYPE ASCIZ STRING
2185  012674  000410                          BR      66$             ;;GET OVER THE ASCIZ
2186                                  ;;67$:  .ASCIZ  # VECTOR ADDR.: #
2187  012716                          66$:
2188  012716  013746  001126                  MOV     $BDDAT,-(SP)    ;;SAVE $BDDAT FOR TYPEOUT
2189                                                                  ;;TYPES VECTOR ADDRESS OF DEVICE
2190  012722  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2191  012724  104401  012732                  TYPE    ,69$            ;;TYPE ASCIZ STRING
2192  012730  000406                          BR      68$             ;;GET OVER THE ASCIZ
2193                                  ;;69$:  .ASCIZ  # PRIORITY: #
2194  012746                          68$:
```

```
2195
2196                                                    ;.ASCIZ
2197
2198  012746  052737  000004  001124                    BIS     #BIT2,$GDDAT
2199  012754  013746  001124                            MOV     $GDDAT,-(SP)    ;;SAVE $GDDAT FOR TYPEOUT
2200                                                                            ;;TYPES PRIORITY OF DEVICE
2201  012760  104402                                    TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2202  012762  104401  001171                            TYPE,   $CRLF
2203  012766  023727  001126  000100                    CMP     $BDDAT,#100     ;IS VECTOR LESS THAN 100?
2204  012774  002070                                    BGE     2$              ;NO-THEN CONTINUE.
2205  012776  104401  013004                            TYPE    ,71$            ;;TYPE ASCIZ STRING
2206  013002  000430                                    BR      70$             ;;GET OVER THE ASCIZ
2207                                          ;;71$:    .ASCIZ  <15><12>#ILLEGAL VECTOR ADDR. CAN NOT BE LESS THAN 100#
2208  013064                                  70$:
2209  013064                                  1$:
2210  013064  104401  013072                            TYPE    ,73$            ;;TYPE ASCIZ STRING
2211  013070  000426                                    BR      72$             ;;GET OVER THE ASCIZ
2212                                          ;;73$:    .ASCIZ  <15><12>#RESET VECTOR SWITCHES AND PRESS CONTINUE.#
2213  013146                                  72$:
2214  013146  000000                                    HALT
2215  013150  162716  000002                            SUB     #2,(SP)
2216  013154  000442                                    BR      3$
2217  013156  023737  001210  001200  2$:               CMP     RVT,XVT         ;ARE VECTOR ADDRS. THE SAME?
2218  013164  001036                                    BNE     3$              ;NO CONTINUE.
2219
2220  013166  104401  013174                            TYPE    ,75$            ;;TYPE ASCIZ STRING
2221  013172  000432                                    BR      74$             ;;GET OVER THE ASCIZ
2222                                          ;;75$:    .ASCIZ  <15><12>#ILLEGAL-INPUT AND OUTPUT VECTOR ADDRS. THE SAME.#
2223  013260                                  74$:
2224  013260  000701                                    BR      1$
2225
2226  013262  000002                          3$:       RTI
2227
2228

                ;*CKADR ROUTINE TO CHECK VALIDITY OF
                ;*AN ADDRESS.
                ;*CALL = JSR PC,CKADR
                ;*REQUIRED:     ADDRESS IN R0
                ;*CARRY BIT CLEAR ON RETURN IF LEGAL
                ;*CARRY BIT SET ON RETURN IF ILLEGAL

2237
2238  013264  032700  000001          CKADR:  BIT     #BIT0,R0        ;ODD ADDRESS?
2239  013270  001422                          BEQ     3$              ;NO CONTINUE
2240  013272  104401  013300                  TYPE    ,65$            ;;TYPE ASCIZ STRING
2241  013276  000416                          BR      64$             ;;GET OVER THE ASCIZ
2242                                  ;;65$:  .ASCIZ  <15><12>#ODD ADDRESS NOT ALLOWED!#
2243  013334                          64$:
2244  013334  000435                          BR      CKADRE
2245
2246  013336  005700                  3$:     TST     R0              ;ANY ADDRESS ENTERED?
2247  013340  001001                          BNE     4$              ;YES-CONTINUE.
2248
2249  013342  000207                          RTS PC                  ;NO-EXIT.
2250
```

```
2251  013344                                  4$:
2252
2253  013344  005700                                  TST     R0              ;NUMBER LARGE ENOUGH?
2254  013346  100013                                  BPL     2$              ;NO-REPORT ERROR
2255  013350  032700  040000                          BIT     #BIT14,R0
2256  013354  001410                                  BEQ     2$
2257  013356  032700  020000                          BIT     #BIT13,R0
2258  013362  001405                                  BEQ     2$
2259  013364  020027  177670                          CMP     R0,#177670      ;ADDRESS TOO LARGE?
2260  013370  003002                                  BGT     2$
2261  013372  000241                                  CLC
2262  013374  000207                                  RTS     PC
2263
2264
2265  013376                                  2$:
2266  013376  104401  013404                          TYPE    ,67$            ;;TYPE ASCIZ STRING
2267  013402  000412                                  BR      66$             ;;GET OVER THE ASCIZ
2268                                          ;;67$:  .ASCIZ  <15><12>#ILLEGAL ADDRESS !#
2269  013430                                  66$:
2270
2271  013430  000261                          CKADRE: SEC
2272  013432  000207                                  RTS     PC
2273

                ;*ROUTINE TO HANDLE KEYBOARD INTERRUPTS

2277
2278  013434  017737  165506  001270  KEYSRV: MOV     @$TKB,CHAR      ;READ CHAR.
2279  013442  042737  000200  001270          BIC     #BIT7,CHAR      ;STRIP PARITY IF ANY
2280  013450  105777  165474          1$:     TSTB    @$TPS           ;SEE IF PRINTER BUSY
2281  013454  100375                          BPL     1$
2282  013456  113777  001270  165466          MOVB    CHAR,@$TPB      ;ECHO
2283
2284  013464  123727  001270  000003          CMPB    CHAR,#3         ;↑C TYPED?
2285  013472  001002                          BNE     2$
2286  013474  000137  003060                  JMP     RSTART          ;YES-RESTART
2287  013500  123727  001270  000001  2$:     CMPB    CHAR,#1         ;↑A TYPED?
2288  013506  001002                          BNE     3$
2289  013510  000137  001000                  JMP     @#1000          ;YES-START
2290
2291  013514  123727  001270  000015  3$:     CMPB    CHAR,#15        ;<CR> TYPED?
2292  013522  001010                          BNE     5$
2293
2294  013524  104401  001171                  TYPE,   $CRLF
2295  013530  013737  001236  001230          MOV     TTYPAT,PATTRN   ;CHANGE PATTERN
2296  013536  005037  001236                  CLR     TTYPAT
2297  013542  000002                  4$:     RTI
2298
2299  013544  123727  001270  000023  5$:     CMPB    CHAR,#23                ;"↑S" TYPED?
2300  013552  001463                          BEQ     SWRSET                  ;IF YES TYPE SWR.
2301
2302  013554  123727  001270  000005          CMPB    CHAR,#5                 ;"↑E" TYPED?
2303  013562  001425                          BEQ     LEXAM                   ;YES-TYPE OUT A LOCATION.
2304
2305
2306  013564  123727  001270  000060  6$:     CMPB    CHAR,#'0        ;OCTAL DIGIT?
```

```
2307  013572  002763                          BLT     4$                      ;NO-IGNORE
2308  013574  123727  001270  000067          CMPB    CHAR,#'7
2309  013602  003357                          BGT     4$
2310  013604  006337  001236                  ASL     TTYPAT
2311  013610  006337  001236                  ASL     TTYPAT
2312  013614  006337  001236                  ASL     TTYPAT
2313  013620  042737  177770  001270          BIC     #177770,CHAR
2314  013626  053737  001270  001236          BIS     CHAR,TTYPAT
2315  013634  000742                          BR      4$
2316
2317                                          ;
2318                                          ;ROUTINE TO TYPE OUT THE CONTENTS OF A LOCATION
2319                                          ;
2320  013636                          LEXAM:
2321  013636  104401  013644                  TYPE    ,65$                    ;;TYPE ASCIZ STRING
2322  013642  000402                          BR      64$                     ;;GET OVER THE ASCIZ
2323                                  ;;65$:  .ASCIZ  #1E#
2324  013650                          64$:
2325  013650  104410                          RDOCT
2326  013652  012637  013720                  MOV     (SP)+,1$
2327  013656  104401  013664                  TYPE    ,67$                    ;;TYPE ASCIZ STRING
2328  013662  000402                          BR      66$                     ;;GET OVER THE ASCIZ
2329                                  ;;67$:  .ASCIZ  <15><12>
2330  013670                          66$:
2331  013670  013746  013720                  MOV     1$,-(SP)                ;;SAVE 1$ FOR TYPEOUT
2332  013674  104402                          TYPOC                           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2333  013676  104401  013704                  TYPE    ,69$                    ;;TYPE ASCIZ STRING
2334  013702  000402                          BR      68$                     ;;GET OVER THE ASCIZ
2335                                  ;;69$:  .ASCIZ  # / #
2336  013710                          68$:
2337  013710  017746  000004                  MOV     @1$,-(SP)               ;;SAVE @1$ FOR TYPEOUT
2338  013714  104402                          TYPOC                           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2339  013716  000002                          RTI
2340
2341  013720  000000                  1$:     .WORD   0                       ;TEMP STORAGE FOR LEXAM.
2342
2343                                          ;
2344                                          ;ROUTINE TO TYPE OUT THEN CHANGE THE SOFTWARE SWITCH REGISTER
2345                                          ;
2346  013722                          SWRSET:
2347  013722  104401  013730                  TYPE    ,65$                    ;;TYPE ASCIZ STRING
2348  013726  000404                          BR      64$                     ;;GET OVER THE ASCIZ
2349                                  ;;65$:  .ASCIZ  <15><12>#SWR= #
2350  013740                          64$:
2351  013740  104414                          SWCAL
2352  013742  013746  001302                  MOV     SWREQ,-(SP)             ;;SAVE SWREQ FOR TYPEOUT
2353  013746  104402                          TYPOC                           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2354  013750  104401  013756                  TYPE    ,67$                    ;;TYPE ASCIZ STRING
2355  013754  000402                          BR      66$                     ;;GET OVER THE ASCIZ
2356                                  ;;67$:  .ASCIZ  #/ #
2357  013762                          66$:
2358  013762  104410                          RDOCT
2359  013764  012637  001302                  MOV     (SP)+,SWREQ
2360  013770  000002                          RTI
2361
2362
```

```
                ;*ROUTINE TO HANDLE TRAPS TO LOC 4, 10 AND.
                ;*INTERRUPTS TO WRONG VECTORS.
                ;*.+2, IOTT(TRAPS) WERE PUT IN LOCATIONS 4-1000

 2369
 2370  013772                                IOTRD:
 2371  013772  011637  014142                       MOV     (R6),2$         ;GET WHERE WE TRAPPED TO.
 2372  013776  162737  000004  014142               SUB     #4,2$           ;=WHERE R6 RETURN 10-4
 2373  014004  104401  014012                       TYPE    ,65$            ;;TYPE ASCIZ STRING
 2374  014010  000412                               BR      64$             ;;GET OVER THE ASCIZ
 2375                                  ;;65$:  .ASCIZ  <15><12>*ILLEGAL TRAP TO: #
 2376  014036                                 64$:
 2377  014036  013746  014142                       MOV     2$,-(SP)        ;;SAVE 2$ FOR TYPEOUT
 2378  014042  104402                               TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 2379  014044  104401  014052                       TYPE    ,67$            ;;TYPE ASCIZ STRING
 2380  014050  000407                               BR      66$             ;;GET OVER THE ASCIZ
 2381                                  ;;67$:  .ASCIZ  # FROM LOC.: #
 2382  014070                                 66$:
 2383  014070  062706  000004                       ADD     #4,R6           ;POINT TO WHERE WE TRAPPED FROM.
 2384  014074  011637  014144                       MOV     (R6),3$         ;PICK UP LOC.
 2385  014100  162737  000002  014144               SUB     #2,3$           ;FROM REAL ADDR.
 2386  014106  013746  014144                       MOV     3$,-(SP)        ;;SAVE 3$ FOR TYPEOUT
 2387  014112  104402                               TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 2388
 2389  014114  023727  014142  000004               CMP     2$,#4           ;DID WE TRAP TO LOC 4?
 2390  014122  001405                               BEQ     1$              ;IF SO - DON'T RETURN!
 2391  014124  023727  014142  000010               CMP     2$,#10          ;DID WE TRAP TO LOC. 10?
 2392  014132  001401                               BEQ     1$              ;IF SO - DON'T RETURN!
 2393  014134  000002                               RTI                     ;TRY RETURNING.
 2394
 2395  014136  000000                         1$:   HALT                    ;WE STOPPED HERE BECAUSE WE TRAPPED
 2396  014140  000776                               BR      1$              ;TO LOC 4 OR LOC 10.  THIS IS A
 2397                                                                       ;FATAL CONDITION THAT WE CAN NOT
 2398                                                                       ;RECOVER FROM.
 2399
 2400  014142  000000                         2$:   .WORD   0               ;USED BY IOTRP TO STORE WHERE WE TRAPPED TO.
 2401  014144  000000                         3$:   .WORD   0               ;USED BY IOTRP TO STORE WHERE WE TRAPPED FROM.
 2402

                ;*
                ;*SWRCAL    ROUTINE TO READ THE HARDWARE SWITCH REGISTER IF IT
                ;*EXISTS AND IF IT ITS NO-ZERO.
                ;*CALL = SWCAL
                ;*EXITS WITH SWR VALUE IN SWREQ IF HSWR IS USED.


 2411
 2412  014146  005777  165126               SWRCAL: TST     @BSWR
 2413  014152  001403                               BEQ     1$
 2414  014154  017777  165120  164756               MOV     @BSWR,@SWR
 2415  014162  000002                         1$:   RTI
 2416
 2417
```

```
                        ;*
                        ;*HALTER   ROUTINE USED TO HALT ON ERROR IF NO SWR(HARDWARE)
                        ;*IS BEING USED THEN "HOLDS ON ERROR".
                        ;*CALL = HALTCL
                        ;*
 2425  014164  005777  165110          HALTER: TST     @#SWR
 2426  014170  001402                          BEQ     1$
 2427
 2428  014172  000000                          HALT            ;ERROR HALT.
 2429
 2430  014174  000425                          BR      3$
 2431
 2432  014176                          1$:
 2433  014176  104401  014204                  TYPE    ,65$            ;;TYPE ASCIZ STRING
 2434  014202  000412                          BR      64$             ;;GET OVER THE ASCIZ
 2435                                  ;;65$:  .ASCIZ  <15><12>#HOLDING ON ERROR#
 2436  014230                          64$:
 2437  014230  005037  177776                  CLR     PS
 2438  014234  052777  000100  164702          BIS     #100,@#TKS
 2439  014242  005777  164672          2$:     TST     @#SWR
 2440  014246  100775                          BMI     2$
 2441
 2442  014250  000002                  3$:     RTI
 2443
 2444
 2445
 2446
 2447


                        ;*
                        ;*RESETR- ROUTINE TO ISSUE SYSTEM INITIALIZE. AND WAIT FOR TYY
                        ;*SETTLE DOWN FIRST.

                        ;*CALL = RESETC
                        ;*
 2456
 2457  014252                          RESETR:
 2458  014252  104401  014260                  TYPE    ,65$            ;;TYPE ASCIZ STRING
 2459  014256  000402                          BR      64$             ;;GET OVER THE ASCIZ
 2460                                  ;;65$:  .ASCIZ  <1><1><1>##
 2461  014264                          64$:
 2462  014264  000005                          RESET                   ;SYSTEM INITIALIZE STATEMENT.
 2463  014266  000002                          RTI                     ;RETURN.
 2464
 2465


                        ;*TEXT USED BY MASTER TO TRANSFER TO SLAVE.


 2471
 2472  014270                          TABLE:
 2473  014270  000000                  TEXT:   000000
 2474  014272  000001                          000001                  ;FLOATING ONES.
```

```
2475  014274  000002                  000002
2476  014276  000004                  000004
2477  014300  000010                  000010
2478  014302  000020                  000020
2479  014304  000040                  000040
2480  014306  000100                  000100
2481  014310  000200                  000200
2482  014312  000400                  000400
2483  014314  001000                  001000
2484  014316  002000                  002000
2485  014320  004000                  004000
2486  014322  010000                  010000
2487  014324  020000                  020000
2488
2489  014326  040000                  040000
2490  014330  100000                  100000
2491  014332  000000                  000000
2492  014334  177777                  177777          ;FLOATING ZEROS.
2493  014336  177776                  177776
2494  014340  177775                  177775
```

```
2495   014342   177773                          177773
2496   014344   177767                          177767
2497   014346   177757                          177757
2498   014350   177737                          177737
2499   014352   177677                          177677
2500   014354   177577                          177577
2501   014356   177377                          177377
2502   014360   176777                          176777
2503   014362   175777                          175777
2504   014364   173777                          173777
2505   014366   167777                          167777
2506   014370   157777                          157777
2507   014372   137777                          137777
2508   014374   077777                          077777
2509
2510   014376   000000                          000000          ;ON/OFF
2511   014400   177777                          177777
2512   014402   000000                          000000
2513   014404   177777                          177777
2514   014406   000000                          000000
2515   014410   177777                          177777
2516   014412   000000                          000000
2517   014414   177777                          177777
2518   014416   000000                          000000
2519   014420   000000                          000000
2520   014422   177777                          177777
2521   014424   125252                          125252          ;ALTERNATES
2522   014426   052525                          052525
2523   014430   125252                          125252
2524   014432   052525                          052525
2525   014434   125252                          125252
2526   014436   052525                          052525
2527   014440   125252                          125252
2528   014442   052525                          052525
2529   014444   125252                          125252
2530   014446   052525                          052525
2531   014450   125252                          125252
2532   014452   052525                          052525
2533
2534   014454   000377                          000377          ;BYTE ALTERNATES
2535   014456   177400                          177400
2536   014460   000377                          000377
2537   014462   177400                          177400
2538   014464   000377                          000377
2539   014466   177400                          177400
2540   014470   000377                          000377
2541   014472   177400                          177400
2542   014474   000377                          000377
2543   014476   177400                          177400
2544   014500   000377                          000377
2545   014502   177400                          177400
2546   014504   000377                          000377
2547   014506   177400                          177400
2548   014510   000377                          000377
2549   014512   177777                          177777
2550   014514   000377                          000377
```

```
2551   014516  000000                                  000000
2552   014520  000377                                  000377
2553   014522  177777                                  177777
2554   014524  177400                                  177400
2555   014526  000000                                  000000
2556   014530  177400                                  177400
2557
2558   014532  000000                                  000000
2559   014534  177777                                  177777
2560   014536  177776                                  177776
2561   014540  000001                                  000001
2562   014542  177775                                  177775
2563   014544  000002                                  000002
2564   014546  177773                                  177773
2565   014550  000004                                  000004
2566   014552  177767                                  177767
2567   014554  000010                                  000010
2568   014556  177757                                  177757
2569   014560  000020                                  000020
2570   014562  177737                                  177737
2571   014564  000040                                  000040
2572   014566  177677                                  177677
2573   014570  000100                                  000100
2574   014572  177577                                  177577
2575   014574  000200                                  000200
2576   014576  177377                                  177377
2577   014600  000400                                  000400
2578   014602  176777                                  176777
2579   014604  001000                                  001000
2580   014606  175777                                  175777
2581
2582   014610  002000                                  002000
2583   014612  173777                                  173777
2584   014614  004000                                  004000
2585   014616  167777                                  167777
2586   014620  010000                                  010000
2587   014622  157777                                  157777
2588   014624  020000                                  020000
2589   014626  137777                                  137777
2590   014630  040000                                  040000
2591   014632  077777                                  077777
2592   014634  100000                                  100000
2593   014636  077777                                  077777
2594   014640  000000                                  000000
2595   014642                          TABLEE:
2596   014642  000000                  TEXTE:  000000
2597
2598   014644                          STORE:
2599   014644  000170                  STABL:  .BLKW   120.
2600   015224  000000                  STABLE: .WORD   0
2601
2602                                            ;
2603                                            ;ASCII MESSAGES
2604                                            ;
2605
2606   015226  005015  052123  051101  MSM:    .ASCIZ  <15><12>/START MODE/
```

```
 2607  015234  020124  047515  042504
 2608  015242     000
 2609  015243     015  044412  046515  MIM:    .ASCIZ  <15><12>/IMMED.  SEND MODE/
 2610  015250  042105  020056  051440
 2611  015256  047105  020104  047515
 2612  015264  042504     000
 2613  015267     015  041012  051125  MBM:    .ASCIZ  <15><12>/BURST  SEND  MODE/
 2614  015274  052123  020040  042523
 2615  015302  042116  020040  047515
 2616  015310  042504     000
 2617
 2618          015314                  .EVEN
 2619
 2620


               ;*SYSMAC ROUTINES BEGIN HERE


 2626
 2627
 2628
 2629                                  .SBTTL  END OF PASS ROUTINE
 2630
 2631                                  ;;*****************************************************************
 2632                                  ;*INCREMENT THE PASS NUMBER ($PASS)
 2633                                  ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
 2634                                  ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
 2635                                  ;*IF THERES A MONITOR GO TO IT
 2636                                  ;*IF THERE ISN'T JUMP TO RETI
 2637
 2638  015314                          $EOP:
 2639  015314  104414                         SWCAL           ;READ THE SWITCH REGISTER.
 2640  015316  032777  002000  163614         BIT     #BIT10,@$SWR
 2641  015324  001427                         BEQ     RETCON
 2642  015326  005237  001100                 INC     $PASS
 2643  015332  000423                         BR      RET
 2644  015334                          RET1:
 2645  015334  104401  015342                 TYPE    ,65$            ;;TYPE ASCIZ STRING
 2646  015340  000410                         BR      64$             ;;GET OVER THE ASCIZ
 2647                                  ;;65$:  .ASCIZ  #  ERROR COUNT  #
 2648  015362                          64$:
 2649  015362  013746  001112                 MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
 2650  015366  104405                         TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
 2651  015370  104401  015376                 TYPE    ,67$            ;;TYPE ASCIZ STRING
 2652  015374  000402                         BR      66$             ;;GET OVER THE ASCIZ
 2653                                  ;;67$:  .ASCIZ  #   #
 2654  015402                          66$:
 2655
 2656  015402  000002                  RET:    RTI
 2657
 2658  015404                          RETCON:
 2659
 2660  015404  005037  001102                 CLR     $TSTNM          ;;ZERO THE TEST NUMBER
 2661  015410  005037  001166                 CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
 2662  015414  005237  001100                 INC     $PASS           ;;INCREMENT THE PASS NUMBER
```

```
2663  015420  042737  100000  001100          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
2664  015426  005327                          DEC     (PC)+           ;;LOOP?
2665  015430  000001                $EOPCT:   .WORD   1
2666  015432  003022                          BGT     $DOAGN          ;;YES
2667  015434  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
2668  015436  000001                $ENDCT:   .WORD   1
2669  015440  015430                          $EOPCT
2670  015442  104401  015507                  TYPE    ,$ENDMG         ;;TYPE "END PASS #"
2671  015446  013746  001100                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
2672  015452  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
2673  015454  104401  015504                  TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
2674  015460  013700  000042        $GET42:   MOV     @#42,R0         ;;GET MONITOR ADDRESS
2675  015464  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
2676  015466  000005                          RESET                   ;;CLEAR THE WORLD
2677  015470  004710                $ENDAD:   JSR     PC,(R0)         ;;GO TO MONITOR
2678  015472  000240                          NOP                     ;;SAVE ROOM
2679  015474  000240                          NOP                     ;;FOR
2680  015476  000240                          NOP                     ;;ACT11
2681  015500                        $DOAGN:
2682  015500  000137                          JMP     @(PC)+          ;;RETURN
2683  015502  015334                $RTNAD:   .WORD   RET1
2684  015504     377     377     000  $ENULL:  .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
2685  015507     015  042412  042116  $ENDMG:  .ASCIZ  <15><12>/END PASS #/
2686  015514  050040  051501  020123
2687  015522  000043
2688
2689                                  .SBTTL  SCOPE HANDLER ROUTINE
2690
2691                                  ;;**********************************************************************
2692                                  ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
2693                                  ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
2694                                  ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
2695                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2696                                  ;*SW14=1         LOOP ON TEST
2697                                  ;*SW11=1         INHIBIT ITERATIONS
2698                                  ;*CALL
2699                                  ;*      SCOPE           ;;SCOPE=IOT
2700
2701  015524                        $SCOPE:
2702  015524  104414                          SWCAL
2703
2704  015526  032777  040000  163404  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
2705  015534  001055                          BNE     $OVER           ;;YES IF SW14=1
2706                                  ;#####START OF CODE FOR THE XOR TESTER#####
2707  015536  000416                $XTSTR:   BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
2708                                                                  ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
2709  015540  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
2710  015544  012737  015564  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
2711  015552  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
2712  015556  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
2713  015562  000436                          BR      $SVLAD          ;;GO TO THE NEXT TEST
2714  015564  022626                5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
2715  015566  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
2716  015572  000436                          BR      $OVER           ;;LOOP ON THE PRESENT TEST
2717  015574                        6$:;#####END OF CODE FOR THE XOR TESTER#####
2718  015574  105737  001103        2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
```

```
2719  015600  001404                          BEQ     3$                      ;;BR IF NO
2720  015602  105037  001103          4$:     CLRB    $ERFLG                  ;;ZERO THE ERROR FLAG
2721  015606  005037  001166                  CLR     $TIMES                  ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
2722  015612  032777  004000  163320  3$:     BIT     #BIT11,@SWR             ;;INHIBIT ITERATIONS?
2723  015620  001011                          BNE     1$                      ;;BR IF YES
2724  015622  005737  001100                  TST     $PASS                   ;;IF FIRST PASS OF PROGRAM
2725  015626  001406                          BEQ     1$                      ;;      INHIBIT ITERATIONS
2726  015630  005237  001104                  INC     $ICNT                   ;;INCREMENT ITERATION COUNT
2727  015634  023737  001166  001104          CMP     $TIMES,$ICNT            ;;CHECK THE NUMBER OF ITERATIONS MADE
2728  015642  002012                          BGE     $OVER                   ;;BR IF MORE ITERATION REQUIRED
2729  015644  012737  000001  001104  1$:     MOV     #1,$ICNT                ;;REINITIALIZE THE ITERATION COUNTER
2730  015652  013737  015704  001166          MOV     $MXCNT,$TIMES           ;;SET NUMBER OF ITERATIONS TO DO
2731  015660  105237  001102          $SVLAD: INCB    $TSTNM                  ;;COUNT TEST NUMBERS
2732  015664  011637  001106                  MOV     (SP),$LPADR             ;;SAVE SCOPE LOOP ADDRESS
2733  015670  013777  001102  163244  $OVER:  MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER
2734  015676  013716  001106                  MOV     $LPADR,(SP)             ;;FUDGE RETURN ADDRESS
2735  015702  000002                          RTI                             ;;FIXES PS
2736  015704  003720                  $MXCNT: 2000.                           ;;MAX. NUMBER OF ITERATIONS
2737
2738
2739                                          .SBTTL  ERROR HANDLER ROUTINE
2740                                  ;;**************************************************************
2741                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND ERROR COUNT.
2742                                  ;* IT WILL TYPE THE PC OF THE ERROR CALL IF SWR13=0
2743                                  ;* CALL = ERROR
2744
2745  015706  104414          $ERROR: SWCAL                           ;READ THE SWITCH REGISTER.
2746  015710  105237  001103          INCB    $ERFLG                          ;SGEGT THE ERROR FLAG.
2747  015714  001774                  BEQ     $ERROR
2748  015716  005237  001112          INC     $ERTTL                          ;INCREMENT ERROR COUNT.
2749  015722  011637  001116          MOV     (SP),$ERRPC                     ;GET PC+2 OF ERROR CALL.
2750  015726  162737  000002  001116  SUB     #2,$ERRPC                       ;MAKE PC OF ERROR CALL.
2751
2752  015734  032777  020000  163176  BIT     #BIT13,@SWR                     ;INHIBIT TYPEOUT OF THE ERROR?
2753  015742  001053                  BNE     BSR15                           ;YES-THEN SKIP TYPEOUT.
2754
2755  015744  005237  001304          INC     ERTYCN                          ;INCREMENT THE #OF ERRORS TYPED OUT.
2756  015750  023727  001304  000013  CMP     ERTYCN, #11.                    ;HAVE WE TYPED OUT 10 ERRORS?
2757  015756  001030                  BNE     1$
2758
2759  015760  104401  015766          TYPE    ,65$                    ;;TYPE ASCIZ STRING
2760  015764  000421                  BR      64$                     ;;GET OVER THE ASCIZ
2761                          ;;65$:  .ASCIZ  <15><12>#HALTED ON TEN ERRORS TYPED OUT#
2762  016030                  64$:
2763
2764  016030  000000                  HALT
2765  016032  005037  001304          CLR     ERTYCN
2766  016036  000422                  BR      BSR14
2767
2768
2769  016040                  1$:
2770  016040  104401  016046          TYPE    ,67$                    ;;TYPE ASCIZ STRING
2771  016044  000407                  BR      66$                     ;;GET OVER THE ASCIZ
2772                          ;;67$:  .ASCIZ  <15><12>#ERROR PC#<15><12>
2773  016064                  66$:
2774  016064  013746  001116          MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
```

```
2775  016070  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2776
2777  016072  104414                  BSR15:  SWCAL                           ;READ THE SWITCH REGISTER.
2778  016074  005777  163040                  TST     @SWR                    ;HALT ON ERROR?
2779  016100  100001                          BPL     BSR14                   ;NO-CONTINUE.
2780
2781  016102  104415                          HALTCL
2782
2783  016104                          BSR14:
2784
2785
2786  016104  000002                          RTI
2787
2788
2789
2790                                  ;;**********************************************************************
2791                                          .SBTTL  TTY INPUT ROUTINE
2792
2793                                  ;       NOTE: THIS ROUTINE IS NOT A SYSMAC ROUTINE. IT IS
2794                                  ;       A CLOSE COPY OF THE ORIGNAL ".$READ" EXCEPT FOR CONTROL
2795                                  ;       CHARACTER DETECTION WHICH SENDS PROGRAM TO "KEYSRV"
2796                                  ;       WHEN A CONTROL CHARACTER IS DETECTED.
2797
2798                                  ;*INPUT A SINGLE CHARACTER FOR THE TTY
2799                                  ;*CALL:
2800                                  ;*      RDCHAR                  ;INPUT A CHAR. FROM THE TTY.
2801                                  ;*RETURN HERE                   ;CHAR. ON STACK.
2802
2803  016106  011646                  $RDCHR: MOV     (SP),-(SP)              ;KPUSH DOWN THE PS.
2804  016110  016666  000004  000002          MOV     4(SP),2(SP)             ;SAVE THE PC
2805  016116  105777  163022          1$:     TSTB    @$TKS                   ;WAIT FOR A CHAR TO BE TYPED.
2806  016122  100375                          BPL     1$
2807  016124  117766  163016  000004          MOVB    @$TKB,4(SP)             ;READ THE TTY.
2808  016132  042766  177600  000004          BIC     #177600,4(SP)   ;GET RID OF PARITY,ETC.
2809  016140  000002                          RTI                             ;GO BACK TO CALLER.
2810
2811                                  ;;**********************************************************************
2812                                  ;;*INPUT A STRING FROM THE TTY
2813                                  ;*CALL:
2814                                  ;*      RDLIN           ;READ A STRING FROM THE TTY.
2815                                  ;*      RETURN HERE             ;ADDRESS OF FIRST CHAR ON STACK.
2816                                  ;*
2817
2818  016142  010346                  $RDLIN: MOV     R3,-(SP)                ;SAVE R3.
2819  016144  012703  016272          1$:     MOV     #$TTYIN,R3              ;PUT ADDR. OF BUFFER IN R3.
2820  016150  022703  016302          2$:     CMP     #$TTYIN+8.,R3   ;BUFFER FULL?
2821  016154  101405                          BLOS    4$                      ;IF YES ASK OPER WHATS GOING ON?
2822
2823  016156  104406                          RDCHR                           ;GOTO READ A CHAR. ROUTINE.
2824  016160  112613                          MOVB    (SP)+,(R3)              ;STORE THE CHAR.
2825  016162  122713  000177                  CMPB    #177,(R3)               ;WAS CHAR A RUBOUT??
2826  016166  001003                          BNE     3$                      ;NO CONTINUE.
2827  016170  104401  001170          4$:     TYPE    ,$QUES          ;YES TYPE A "?".
2828  016174  000763                          BR      1$                      ;RESTART INPUT.
2829  016176  111337  016270          3$:     MOVB    (R3),8$                 ;ECHO THE CHAR.
2830
```

```
2831  016202  104401  016270          5$:     TYPE    ,B$
2832  016206  122713  000015                  CMPB    #15,(R3)                ;CHECH FOR CARRAGE RETURN TYPED.
2833  016212  001412                          BEQ     7$                      ;IF NOT GETR NEX T CHAR.
2834  016214  122327  000040                  CMPB    (R3)+,#40
2835  016220  002353                          BGE     2$
2836  016222  012603                          MOV     (SP)+,R3
2837  016224  011646                          MOV     (SP),-(SP)
2838  016226  016666  000004  000002          MOV     4(SP),2(SP)
2839  016234  000137  013434                  JMP     KEYSRV
2840
2841  016240  105013                  7$:     CLRB    (R3)                    ;INSERT ZERO TERMINATOR.
2842  016242  104401  001172                  TYPE    ,$LF                    ;TYPE A LINE FEED.
2843  016246  012603                          MOV     (SP)+,R3                ;RESTORE R3.
2844  016250  011646                          MOV     (SP),-(SP)              ;ADJUST THE STACK AND
2845  016252  016666  000004  000002          MOV     4(SP),2(SP)             ;PUT THE FIRST ADDRESS OF THE
2846  016260  012766  016272  000004          MOV     #$TTYIN,4(SP)           ;CHAR. ON STACK.
2847  016266  000002                          RTI
2848  016270     000     000          B$:     .BYTE   0,0     ;STORE FOR CHAR. ECHO.
2849  016272  000010               $TTYIN:    .BLKB   8.              ;RESERVE 8. BYTES FOR LINE ASCII STORAGE.
2850
2851                                  .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
2852
2853                                  ;;**************************************************************
2854                                  ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2855                                  ;*CHANGE IT TO BINARY.
2856                                  ;*CALL:
2857                                  ;*      RDOCT                           ;;READ AN OCTAL NUMBER
2858                                  ;*      RETURN HERE                     ;;LOW ORDER BITS ARE ON TOP OF THE STACK
2859                                  ;*                                      ;;HIGH ORDER BITS ARE IN $HIOCT
2860
2861  016302  011646               $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
2862  016304  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
2863  016312  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2864  016314  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2865  016316  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2866  016320  104407                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
2867  016322  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
2868  016324  005001                          CLR     R1              ;;CLEAR DATA WORD
2869  016326  005002                          CLR     R2
2870  016330  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
2871  016332  001412                          BEQ     3$              ;;IF ZERO GET OUT
2872  016334  006301                          ASL     R1              ;;*2
2873  016336  006102                          ROL     R2
2874  016340  006301                          ASL     R1              ;;*4
2875  016342  006102                          ROL     R2
2876  016344  006301                          ASL     R1              ;;*8
2877  016346  006102                          ROL     R2
2878  016350  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
2879  016354  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
2880  016356  000764                          BR      2$              ;;LOOP
2881  016360  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
2882  016362  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
2883  016366  010237  016402                  MOV     R2,$HIOCT
2884  016372  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
2885  016374  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
2886  016376  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
```

```
2887  016400  000002                          RTI                     ;;RETURN
2888  016402  000000          $HIOCT: .WORD   0                       ;;HIGH ORDER BITS GO HERE
2889                          .SBTTL  TYPE ROUTINE
2890
2891                          ;;**********************************************************************
2892                          ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
2893                          ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
2894                          ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
2895                          ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
2896                          ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
2897                          ;*
2898                          ;*CALL:
2899                          ;*1) USING A TRAP INSTRUCTION
2900                          ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2901                          ;*OR
2902                          ;*      TYPE
2903                          ;*      MESADR
2904                          ;*
2905
2906  016404  105737  001157  $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
2907  016410  100002                  BPL     1$              ;;BR IF YES
2908  016412  000000                  HALT                    ;;HALT HERE IF NO TERMINAL
2909  016414  000407                  BR      3$              ;;LEAVE
2910  016416  010046          1$:     MOV     R0,-(SP)        ;;SAVE R0
2911  016420  017600  000002          MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
2912  016424  112046          2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
2913  016426  001005                  BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
2914  016430  005726                  TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
2915  016432  012600          60$:    MOV     (SP)+,R0        ;;RESTORE R0
2916  016434  062716  000002  3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
2917  016440  000002                  RTI                     ;;RETURN
2918  016442  122716  000011  4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
2919  016446  001430                  BEQ     8$
2920  016450  122716  000200          CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
2921  016454  001006                  BNE     5$
2922  016456  005726                  TST     (SP)+           ;;POP  <CR><LF> EQUIV
2923  016460  104401                  TYPE                    ;;TYPE A CR AND LF
2924  016462  001171                  $CRLF
2925  016464  105037  016620          CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
2926  016470  000755                  BR      2$              ;;GET NEXT CHARACTER
2927  016472  004737  016554  5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
2928  016476  123726  001156  6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
2929  016502  001350                  BNE     2$              ;;IF NO GO GET NEXT CHAR.
2930  016504  013746  001154          MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
2931                                                          ;;AND THE NULL CHAR.
2932  016510  105366  000001  7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
2933  016514  002770                  BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
2934  016516  004737  016554          JSR     PC,$TYPEC       ;;GO TYPE A NULL
2935  016522  105337  016620          DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
2936  016526  000770                  BR      7$              ;;LOOP
2937
2938                          ;HORIZONTAL TAB PROCESSOR
2939
2940  016530  112716  000040  8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
2941  016534  004737  016554  9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
2942  016540  132737  000007  016620  BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
```

```
                                        BNE     9$                      ;;TAB STOP
                                        TST     (SP)+                   ;;POP SPACE OFF STACK
                                        BR      2$                      ;;GET NEXT CHARACTER
                        162370  $TYPEC: TSTB    @$TPS                   ;;WAIT UNTIL PRINTER IS READY
                                        BPL     $TYPEC
                        000002  162362  MOVB    2(SP),@$TPB             ;;LOAD CHAR TO BE TYPED INTO DATA REG.
                        000015  000002  CMPB    #CR,2(SP)               ;;IS CHARACTER A CARRIAGE RETURN?
                                        BNE     1$                      ;;BRANCH IF NO
                        016620          CLRB    $CHARCNT                ;;YES--CLEAR CHARACTER COUNT
                                        BR      $TYPEX                  ;;EXIT
                        000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
                                        BEQ     $TYPEX                  ;;BRANCH IF YES
                                        INCB    (PC)+                   ;;COUNT THE CHARACTER
                        000000  $CHARCNT:.WORD  0                       ;;CHARACTER COUNT STORAGE
                        000207  $TYPEX: RTS     PC

                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

                                ;;**************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)               ;;NUMBER TO BE TYPED
                                ;*      TYPOS                           ;;CALL FOR TYPEOUT
                                ;*      .BYTE   N                       ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                ;*      .BYTE   M                       ;;M=1 OR 0
                                ;*                                              ;;1=TYPE LEADING ZEROS
                                ;*                                              ;;0=SUPPRESS LEADING ZEROS
                                ;*
                                ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                ;*$TYPOS OR $TYPOC
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)               ;;NUMBER TO BE TYPED
                                ;*      TYPON                           ;;CALL FOR TYPEOUT
                                ;*
                                ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)               ;;NUMBER TO BE TYPED
                                ;*      TYPOC                           ;;CALL FOR TYPEOUT

                        017646  000000  $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
                        116637  000001  017047  MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
                        112637  017051          MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
                        062716  000002          ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
                        000406                  BR      $TYPON
                        112737  000001  017047  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
                        112737  000006  017051          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
                        112737  000005  017046  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
                        010346                  MOV     R3,-(SP)        ;;SAVE R3
                        010446                  MOV     R4,-(SP)        ;;SAVE R4
                        010546                  MOV     R5,-(SP)        ;;SAVE R5
                        113704  017051          MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
                        005404                  NEG     R4
                        062704  000006          ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
                        110437  017050          MOVB    R4,$OMODE       ;;SAVE IT FOR USE
```

```
 2999  016716  113704  017047                    MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
 3000  016722  016605  000012                    MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
 3001  016726  005003                            CLR     R3              ;;CLEAR THE OUTPUT WORD
 3002  016730  006105                  1$:       ROL     R5              ;;ROTATE MSB INTO "C"
 3003  016732  000404                            BR      3$              ;;GO DO MSB
 3004  016734  006105                  2$:       ROL     R5              ;;FORM THIS DIGIT
 3005  016736  006105                            ROL     R5
 3006  016740  006105                            ROL     R5
 3007  016742  010503                            MOV     R5,R3
 3008  016744  006103                  3$:       ROL     R3              ;;GET LSB OF THIS DIGIT
 3009  016746  105337  017050                    DECB    $OMODE          ;;TYPE THIS DIGIT?
 3010  016752  100016                            BPL     7$              ;;BR IF NO
 3011  016754  042703  177770                    BIC     #177770,R3      ;;GET RID OF JUNK
 3012  016760  001002                            BNE     4$              ;;TEST FOR 0
 3013  016762  005704                            TST     R4              ;;SUPPRESS THIS 0?
 3014  016764  001403                            BEQ     5$              ;;BR IF YES
 3015  016766  005204                  4$:       INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
 3016  016770  052703  000060                    BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
 3017  016774  052703  000040          5$:       BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
 3018  017000  110337  017044                    MOVB    R3,8$           ;;SAVE FOR TYPING
 3019  017004  104401  017044                    TYPE    ,8$             ;;GO TYPE THIS DIGIT
 3020  017010  105337  017046          7$:       DECB    $OCNT           ;;COUNT BY 1
 3021  017014  003347                            BGT     2$              ;;BR IF MORE TO DO
 3022  017016  002402                            BLT     6$              ;;BR IF DONE
 3023  017020  005204                            INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
 3024  017022  000744                            BR      2$              ;;GO DO THE LAST DIGIT
 3025  017024  012605                  6$:       MOV     (SP)+,R5        ;;RESTORE R5
 3026  017026  012604                            MOV     (SP)+,R4        ;;RESTORE R4
 3027  017030  012603                            MOV     (SP)+,R3        ;;RESTORE R3
 3028  017032  016666  000002  000004            MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
 3029  017040  012616                            MOV     (SP)+,(SP)
 3030  017042  000002                            RTI                     ;;RETURN
 3031  017044     000                  8$:       .BYTE   0               ;;STORAGE FOR ASCII DIGIT
 3032  017045     000                            .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
 3033  017046     000                  $OCNT:    .BYTE   0               ;;OCTAL DIGIT COUNTER
 3034  017047     000                  $OFILL:   .BYTE   0               ;;ZERO FILL SWITCH
 3035  017050  000000                  $OMODE:   .WORD   0               ;;NUMBER OF DIGITS TO TYPE
 3036                                  .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
 3037
 3038                                  ;;**************************************************************
 3039                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
 3040                                  ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
 3041                                  ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
 3042                                  ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
 3043                                  ;*REPLACED WITH SPACES.
 3044                                  ;*CALL:
 3045                                  ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
 3046                                  ;*      TYPDS                   ;;GO TO THE ROUTINE
 3047
 3048  017052                          $TYPDS:
 3049  017052  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 3050  017054  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 3051  017056  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 3052  017060  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 3053  017062  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 3054  017064  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
```

```
3055  017070  016605  000020                    MOV     20(SP),R5       ;;GET THE INPUT NUMBER
3056  017074  100004                            BPL     1$              ;;BR IF INPUT IS POS.
3057  017076  005405                            NEG     R5              ;;MAKE THE BINARY NUMBER POS.
3058  017100  112766  000055  000001            MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
3059  017106  005000                    1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
3060  017110  012703  017266                    MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
3061  017114  112723  000040                    MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
3062  017120  005002                    2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
3063  017122  016001  017256                    MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
3064  017126  160105                    3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
3065  017130  002402                            BLT     4$              ;;BR IF DONE
3066  017132  005202                            INC     R2              ;;INCREASE THE BCD DIGIT BY 1
3067  017134  000774                            BR      3$
3068  017136  060105                    4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
3069  017140  005702                            TST     R2              ;;CHECK IF BCD DIGIT=0
3070  017142  001002                            BNE     5$              ;;FALL THROUGH IF 0
3071  017144  105716                            TSTB    (SP)            ;;STILL DOING LEADING 0'S?
3072  017146  100407                            BMI     7$              ;;BR IF YES
3073  017150  106316                    5$:     ASLB    (SP)            ;;MSD?
3074  017152  103003                            BCC     6$              ;;BR IF NO
3075  017154  116663  000001  177777            MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
3076  017162  052702  000060            6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
3077  017166  052702  000040            7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
3078  017172  110223                            MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
3079  017174  005720                            TST     (R0)+           ;;JUST INCREMENTING
3080  017176  020027  000010                    CMP     R0,#10          ;;CHECK THE TABLE INDEX
3081  017202  002746                            BLT     2$              ;;GO DO THE NEXT DIGIT
3082  017204  003002                            BGT     8$              ;;GO TO EXIT
3083  017206  010502                            MOV     R5,R2           ;;GET THE LSD
3084  017210  000764                            BR      6$              ;;GO CHANGE TO ASCII
3085  017212  105726                    8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
3086  017214  100003                            BPL     9$              ;;BR IF NO
3087  017216  116663  177777  177776            MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
3088  017224  105013                    9$:     CLRB    (R3)            ;;SET THE TERMINATOR
3089  017226  012605                            MOV     (SP)+,R5        ;;POP STACK INTO R5
3090  017230  012603                            MOV     (SP)+,R3        ;;POP STACK INTO R3
3091  017232  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
3092  017234  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
3093  017236  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
3094  017240  104401  017266                    TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
3095  017244  016666  000002  000004            MOV     2(SP),4(SP)     ;;ADJUST THE STACK
3096  017252  012616                            MOV     (SP)+,(SP)
3097  017254  000002                            RTI                     ;;RETURN TO USER
3098  017256  023420                    $DTBL:  10000.
3099  017260  001750                            1000.
3100  017262  000144                            100.
3101  017264  000012                            10.
3102  017266  000004                    $DBLK:  .BLKW   4
3103
3104                                      .SBTTL  TRAP DECODER
3105
3106                                      ;;**************************************************************
3107                                      ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3108                                      ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3109                                      ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3110                                      ;*GO TO THAT ROUTINE.
```

```
3111
3112  017276  010046                  $TRAP:  MOV     R0,-(SP)          ;;SAVE R0
3113  017300  016600  000002                  MOV     2(SP),R0          ;;GET TRAP ADDRESS
3114  017304  005740                          TST     -(R0)             ;;BACKUP BY 2
3115  017306  111000                          MOVB    (R0),R0           ;;GET RIGHT BYTE OF TRAP
3116  017310  006300                          ASL     R0                ;;POSITION FOR INDEXING
3117  017312  016000  017332                  MOV     $TRPAD(R0),R0     ;;INDEX TO TABLE
3118  017316  000200                          RTS     R0                ;;GO TO ROUTINE
3119
3120
3121                                  ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
3122
3123  017320  011646                  $TRAP2: MOV     (SP),-(SP)        ;;MOVE THE PC DOWN
3124  017322  016666  000004  000002          MOV     4(SP),2(SP)       ;;MOVE THE PSW DOWN
3125  017330  000002                          RTI                       ;;RESTORE THE PSW
3126
3127                                  .SBTTL  TRAP TABLE
3128
3129                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3130                                  ;*BY THE "TRAP" INSTRUCTION.
3131
3132                                  ;       ROUTINE
3133                                  ;       -------
3134  017332  017320                  $TRPAD: .WORD   $TRAP2
3135  017334  016404                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
3136  017336  016650                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3137  017340  016624                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
3138  017342  016664                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
3139  017344  017052                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
3140
3141
3142  017346  016106                          $RDCHR  ;;CALL=RDCHR    TRAP+6(104406)  TTY TYPEIN CHARACTER ROUTINE
3143  017350  016142                          $RDLIN  ;;CALL=RDLIN    TRAP+7(104407)  TTY TYPEIN STRING ROUTINE
3144  017352  016302                          $RDOCT  ;;CALL=RDOCT    TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY
3145
3146  017354  012434                          RCVP    ;;CALL=GETRC    TRAP+11(104411)
3147  017356  012232                          XCVP    ;;CALL=GETXC    TRAP+12(104412)
3148  017360  015314                          $EOP    ;;CALL=EOPCAL   TRAP+13(104413)
3149  017362  014146                          SWRCAL  ;;CALL=SWCAL    TRAP+14(104414)
3150  017364  014164                          HALTER  ;;CALL=HALTCL   TRAP+15(104415)
3151  017366  013772                          IOTRD   ;;CALL=IOTT     TRAP+16(104416)
3152  017370  014252                          RESETR  ;;CALL=RESETC   TRAP+17(104417)
3153                                  .SBTTL  POWER DOWN AND UP ROUTINES
3154
3155                                  ;;*****************************************************************
3156                                  ;POWER DOWN ROUTINE
3157  017372  012737  017532  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC  ;;SET FOR FAST UP
3158  017400  012737  000340  000026          MOV     #340,@#PWRVEC+2   ;;PRIO:7
3159  017406  010046                          MOV     R0,-(SP)          ;;PUSH R0 ON STACK
3160  017410  010146                          MOV     R1,-(SP)          ;;PUSH R1 ON STACK
3161  017412  010246                          MOV     R2,-(SP)          ;;PUSH R2 ON STACK
3162  017414  010346                          MOV     R3,-(SP)          ;;PUSH R3 ON STACK
3163  017416  010446                          MOV     R4,-(SP)          ;;PUSH R4 ON STACK
3164  017420  010546                          MOV     R5,-(SP)          ;;PUSH R5 ON STACK
3165  017422  017746  161512                  MOV     @SWR,-(SP)        ;;PUSH @SWR ON STACK
3166  017426  010637  017536                  MOV     SP,$SAVR6         ;;SAVE SP
```

```
3167  017432  012737  017444  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
3168  017440  000000                          HALT
3169  017442  000776                          BR      .-2              ;;HANG UP
3170
3171                                  ;;*****************************************************************
3172                                  ;POWER UP ROUTINE
3173  017444  012737  017532  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
3174  017452  013706  017536                  MOV     $SAVR6,SP        ;;GET SP
3175  017456  005037  017536                  CLR     $SAVR6           ;;WAIT LOOP FOR THE TTY
3176  017462  005237  017536          1$:     INC     $SAVR6           ;;WAIT FOR THE INC
3177  017466  001375                          BNE     1$               ;;OF  WORD
3178  017470  012677  161444                  MOV     (SP)+,@SWR       ;;POP STACK INTO @SWR
3179  017474  012605                          MOV     (SP)+,R5         ;;POP STACK INTO R5
3180  017476  012604                          MOV     (SP)+,R4         ;;POP STACK INTO R4
3181  017500  012603                          MOV     (SP)+,R3         ;;POP STACK INTO R3
3182  017502  012602                          MOV     (SP)+,R2         ;;POP STACK INTO R2
3183  017504  012601                          MOV     (SP)+,R1         ;;POP STACK INTO R1
3184  017506  012600                          MOV     (SP)+,R0         ;;POP STACK INTO R0
3185  017510  012737  017372  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
3186  017516  012737  000340  000026          MOV     #340,@#PWRVEC+2  ;;PRIO:7
3187  017524  104401                          TYPE                     ;REPORT THE POWER FAILURE
3188  017526  017540          $PWRMG: .WORD   $POWER                   ;;POWER FAIL MESSAGE POINTER
3189  017530  000002                  RTI
3190  017532  000000          $ILLUP: HALT                             ;;THE POWER UP SEQUENCE WAS STARTED
3191  017534  000776                  BR      .-2                      ;; BEFORE THE POWER DOWN WAS COMPLETE
3192  017536  000000          $SAVR6: 0                                ;;PUT THE SP HERE
3193  017540  005015  047520  042527  $POWER: .ASCIZ  <15><12>"POWER"
3194  017546  000122
3195                                  .EVEN
3196            000001                .END
```

```
AMODE   004404          723    737*
BISR    007260         1235   1290*
BIT0  = 000001          134*    391     774     900     903    1323    1433    1438    1553    1555    1612    1633    1789
                       1793   1854    1856    1872    1873    1878    1924    1948    2238
BIT00 = 000001          124*    134     720     778    1619    1635    1932    1950    1978    1995
BIT01 = 000002          123*    133
BIT02 = 000004          122*    132
BIT03 = 000010          121*    131
BIT04 = 000020          120*    130
BIT05 = 000040          119*    129
BIT06 = 000100          118*    128
BIT07 = 000200          117*    127
BIT08 = 000400          116*    126    1613    1641    1926    1956    1972    2003
BIT09 = 001000          115*    125
BIT1  = 000002          133*   1814    1854    1856
BIT10 = 002000          114*   2640
BIT11 = 004000          113*   1182    1284    2722
BIT12 = 010000          112*
BIT13 = 020000          111*    942     995    1160    1262    1361    2257    2752
BIT14 = 040000          110*    972    1200    1217    1337    1401    2255    2704
BIT15 = 100000          109*    963
BIT2  = 000004          132*   2198
BIT3  = 000010          131*
BIT4  = 000020          130*
BIT5  = 000040          129*
BIT6  = 000100          128*    343     519     521     753     766     888     894     987    1237    1244    1438    1459
                       1534   1552    1553    1555    1562    1572    1573    1578    1672    1684    1708    1713    1720
                       1810   1814    1853    1854    1856    1862    1872    1873    1878    2033    2045    2070    2075
                       2082
BIT7  = 000200          127*    743    1349    1534    2118    2149    2279
BIT8  = 000400          126*   1124    1131    1187    1193    1226    1231    1302    1310    1371    1379    1418    1426
                       1534   1552    1553    1555    1572    1573    1578    1595    1597    1[illegible]    1633    1663    1672
                       1684   1708    1713    1720    1810    1814    1853    1854    1856    1872    1873    1878    1902
                       1924   1948    2024    2033    2045    2070    2075    2082
BIT9  = 001000          125*
BMODE   004416          727    742*    744
BPTVEC= 000014          141*
BSM     007574         1341   1393*   1402
BSR14   016104         2766   2779    2733*
BSR15   016072         2753   2777*
BSWR    001300          248*    495*    658    2412    2414    2425
CHAR    001270          244*   2278*   2279*   2282    2284    2287    2291    2299    2302    2306    2308    2313*   2314
CIO     005700          929   1017*
CKADR   013264          380    428     456     469    2238*
CKADRE  013430         2244   2271*
CMODE   004444          731    750*
COMP    004314          681    692     706*
CR    = 000015           49*   2949    2959
CRLF  = 000200           50*   2920    2959
DDISP = 177570           56*    184     294
DECP    004306          678    703*
DISLGH  005604          921    985*
DISPLA  001142          184*    294*    302*    496*   2733*
DISPRE  000176           28*    302
DISTTY  005624          925    992*
DMODE   004554          733    774*
```

```
DSWR   = 177570            55#   183   248   293
EMTVEC= 000030            144#   281*  282*
EOPCAL= 104413            936   1030  1220  1404  1731  2094  3148#
ERRVEC= 000004            137#   291   292*  303*  2709  2710*  2712*  2715*
ERTYCN  001304            250#   486*  2755* 2756  2765*
GETRC  = 104411           885   1234  1662  1702  3146#
GETXC  = 104412           750   1428  2023  2064  3147#
GMODE   005140            856   868#
GNS    = ****** U         346   352   360   369   375   386   395   402   410   424   433   446   452
                          465   505   514   563   580   588   607   804   811   947   951   957   999
                         1049  1066  1165  1169  1175  1267  1271  1277  1366  2178  2186  2193  2207
                         2212  2222  2242  2268  2323  2329  2335  2349  2356  2375  2381  2435  2460
                         2647  2653  2761  2772  3135  3136  3137  3138  3139  3142  3143  3144  3146
                         3147  3148  3149  3150  3151  3152
HALTCL= 104415            965   2781  3150#
HALTER  014164           2425#  3150
HMODE   005152            859   875#  876
HT     = 000011            47#  2918  2959
IMLT    010052            534  1472#  1733
IMODE   005172            862   885#
INCP    004300            675   700#
INPASK  002646            362   449#  457   806   1052
INPM    004662            549   799#
INPMWK  005100            624   827   838#  988   1005  1033
INPRD   005102            840#  869   878   896   904
INSRT1  006260            316*  1097#
INSRT2  006310            319*  1108#
IOTRD   013772           2370#  3151
IOTT   = 104416             23   335   337   415   499   768   3151#
IOTVEC= 000020            142#   279*  280*
ISM     007340           1323#  1338
ISR     010016           1430   1456#
JMODE   005250            864   900#
KEYSRV  013434            487   518   2278#  2839
LEXAM   013636           2303   2320#
LF     = 000012            48#  2953  2959
MBM     015267           1209   1393  2613#
MIM     015243           1140   1326  2609#
MODE    001220            219#   523*  721   725   729   822   823*  1405
MODEI   001222            220#   822*  855   858   861
MODPAT  004062            619   638#
MSM     015226           1117   1300  2606#
MST     007304           1060   1300#  1413
MTTRP   006232           1081   1089#  1103
OMLT    011052            538  1745#  2095
OUTASK  002752            462#   470   609   1069
OUTM    003500            554   559#  569   571   828
OUTMWK  004040            602   619#  623   974
OUTWR   004042            620#   647   653   660   698
OUTWR2  004046            622#   738   746   770   780
PATFRO  001224            221#   573*  574   638   641   645   787
PATOLD  001244            230#   992   1004*
PATOUT  001240            228#   517*  622   816*  826   919   923   927
PATTIN  001242            229#   868*  877*  895*  901*  937   959   985   992   1001  1004  1017  1150*
                         1151  1171
PATTM   001226            223#   592*  674   677   680
```

```
PATTRN  001230          224#   583*   649*   659*   683    685*   688    691    695*   696*   700*   703*   706*
                        737    745    765    779    789*   937    953    2295*
PC     =%000007          68#   380*   428*   456*   469*   742*   756*   775*   786*   792*   1020*  1156*  1185*
                       1194*  1357*  1382*  2249*  2262*  2272*  2664*  2667*  2677*  2682   2927*  2934*  2941*
                       2955*  2957*
PIRQ   = 177772          54#
PIRQVE= 000240          148#
POPSP2= 022626          150#   494    763    893    1486   1503   1683   1719   1759   1776   2044   2081
PPM     004202          643    674#
PPMEXT  004274          686    698#   701    704    707
PR0    = 000000          71#
PR1    = 000040          72#
PR2    = 000100          73#
PR3    = 000140          74#
PR4    = 000200          75#
PR5    = 000240          76#
PR6    = 000300          77#
PR7    = 000340          78#
PS     = 177776          51#    52     719*   755*   758*   769*   889*   1046*  1062*  1239*  1439*  1551*  1564*
                       1571*  1587*  1666*  1673*  1685*  1704*  1709*  1714*  1721*  1809*  1823*  1852*  1864*
                       1871*  1887*  2027*  2034*  2046*  2066*  2071*  2076*  2083*  2437*
PSW    = 177776          52#
PSWR    004162          639    658#
PTRM    001246          232#   1076*  1080   1093*  1107
PTRS    001250          233#   1077*  1096   1104*
PWRVEC= 000024          143#   285*   286*   3157*  3158*  3167*  3173*  3185*  3186*
P1      007310         1302#
P13     007354         1326#
P14     007360         1328#  1335
P15     007366         1330#  1355   1358
P16     007432         1331   1349#
P17     007424         1344#  1350
P18     007442         1352#
P19     007460         1347   1357#
P2      007316         1305#  1306
P20     007550         1362   1371#
P21     007556         1375#  1377
P22     007376         1333#
P23     007404         1335#
P24     007600         1395#
P25     007606         1397#  1446
P26     007666         1398   1418#
P27     007674         1421#  1423
P28     007702         1426#
P29     007742         1436#
P3      007324         1310#  1313
P30     007772         1440#  1443
P31     007774         1442#
P32     007630         1404#
QE      003434          541    544#
RCSRA   001204          210#   333*   458*   799    875    888*   894    900*   903*   1042   1120   1124*  1126
                       1131*  1147   1187*  1189   1193*  1223   1226*  1228   1231*  1237*  1244*  1472   1482
                       1517   1534   1552*  1553*  1554   1563*  1572*  1573*  1575   1586*  1595*  1597*  1599
                       1611*  1612*  1613   1619   1630*  1631   1635   1641   1663*  1672*  1684*  1708*  1713*
                       1720*  2141   2150   2155   2158
RCVP    012434         2141#  3146
```

```
RDBRA   001206          211#   460*   821    868    877    895    901   1130   1150   1192   1290   1498   1596
                       1664   1686  1701
RDCHR = 104406          354   2823  3142#
RDLIN = 104407          520    566   590    814   2866  3143#
RDOCT = 104410          378    389   426    454    467    582   2325   2358   3144#
REDMOD  005106          838    855#
REPPAT  005300          840    919#
RESETC= 104417          986   1516  1533   1550   1791   1812   1833   1851   2092   3152#
RESETR  014252         2457#  3152
RESVEC= 000010          138#
RET     015402         2643   2656#
RETCON  015404         2641   2658#
RET1    015334         2644#  2683
RSTART  003060           33    483#  2286
RST2    003264          501    511#   558
RVT     001210          213#   336*   340*   886*  1235*  1665   1703   2163*  2164   2167   2169   2217
RVTP    001266          242#  1709   2160  2162*
RVT2    001212          214#   336    337*   341*   887*  1236*  2164*  2165*
RXMIT   007470          786   1357  1359#
R0      =%000000         59#   355*   356    364    379*   382    390*   391    398    406    416    417*   418
                        427*   440    441*   442    455*   458    459*   460    468*   471    472*   473    522*
                        523    524    528    532    536    540    544    546    551    553    567*   568    570
                        573    591*   592    593    595    815*   816    817    819    985*  1665*  1669   1670*
                       1703*  1726   1728*  1729*  2026*  2030   2031*  2065*  2088   2090*  2091*  2124*  2125*
                       2126   2155*  2156*  2157   2238   2246   2253   2255   2257   2259   2674*  2677   2863
                       2867*  2870   2886*  2910   2911*  2912   2915*  3049   3059*  3063   3079   3080   3093*
                       3112   3113*  3114   3115*  3116*  3117*  3118*  3159   3184*
R1      =%000001         60#   313*   382*   435    436*   437   1726*  1728   2088*  2089   2090   2112*  2117*
                       2118*  2128   2130   2143*  2148*  2149*  2160   2162   2864   2868*  2872*  2874*  2876*
                       2879*  2882   2885*  3050   3063*  3064   3068   3092*  3160   3183*
R2      =%000002         61#   2865  2869*  2873*  2875*  2877*  2883   2884*  3051   3062*  3066*  3069   3076*
                       3077*  3078   3083*  3091*  3161   3182*
R3      =%000003         62#   2818  2819*  2820   2824*  2825   2829   2832   2834   2836*  2841*  2843*  2992
                       3001*  3007*  3008*  3011*  3016*  3017*  3018   3027*  3052   3060*  3061*  3075*  3078*
                       3087*  3088*  3090*  3162   3181*
R4      =%000004         63#   2993  2995*  2996*  2997*  2998   2999*  3013   3015*  3023*  3026*  3163   3180*
R5      =%000005         64#   2994  3000*  3002*  3004*  3005*  3006*  3007   3025*  3053   3055*  3057*  3064*
                       3068*  3083   3089*  3164   3179*
R6      =%000006         65#    273*   274*   275    323*   2371   2383*  2384
R7      =%000007         66#
SAVPC   001276          247#
SBSM    006664         1209#  1218
SISM    006352         1138#  1201
SLERR   006450         1020   1156  1159#
SMSST   001252          234#  1094*  1106
SMST    006312         1044   1117#  1221
SNDPAT  004322          620    719#
SP      =%000006         67#    277*   291*   299*   303    355    379    390    427    455    468    483*   522
                        567    583    591    815    953*   959*  1001*  1051*  1068*  1079*  1080*  1093   1094
                       1095*  1096*  1104   1105   1106*  1107*  1171*  1177*  1273*  1279*  1368*  1480*  1489
                       1496*  1506   1753*  1762   1769*  1779   2181*  2188*  2199*  2215*  2326   2331*  2337*
                       2352*  2359   2377*  2386*  2649*  2671*  2709*  2712   2714   2715   2732   2734*  2749
                       2774*  2803*  2804*  2807*  2808*  2818*  2824   2836   2837*  2838*  2843   2844*  2845*
                       2846*  2861*  2862*  2863*  2864*  2865*  2867   2870*  2878*  2879   2881   2882*  2884
                       2885   2886   2910*  2911   2912*  2914   2915   2916*  2918   2920   2922   2928   2930*
                       2932*  2940*  2944   2948   2949   2953   2984*  2985   2986   2987*  2992*  2993*  2994*
```

```
SWREQ   001302          249*   330    495    2352   2359*
SWRSET  013722         2300   2346*
SW0   = 000001          106*
SW00  = 000001           96*   106
SW01  = 000002           95*   105
SW02  = 000004           94*   104
SW03  = 000010           93*   103
SW04  = 000020           92*   102
SW05  = 000040           91*   101
SW06  = 000100           90*   100
SW07  = 000200           89*    99
SW08  = 000400           88*    98
SW09  = 001000           87*    97
SW1   = 000002          105*
SW10  = 002000           86*
SW11  = 004000           85*
SW12  = 010000           84*
SW13  = 020000           83*
SW14  = 040000           82*
SW15  = 100000           81*
SW2   = 000004          104*
SW3   = 000010          103*
SW4   = 000020          102*
SW5   = 000040          101*
SW6   = 000100          100*
SW7   = 000200           99*
SW8   = 000400           98*
SW9   = 001000           97*
SXN0    001260          238*   1138*  1196*  1211*  1213   1258*
TABLE   014270          548    598    976    978    1025   1142   1233   1247   2472*
TABLEE  014642         1023    1144   1242   1255   2595*
TABLEP  001234          226*   548*   976    978*   980*   1017   1022*  1023   1025*  1142*  1144   1151   1153*
                       1177    1184*  1233*  1242   1247*  1249   1253*  1255   1273   1292*
TBITVE= 000014          139*
TEXT    014270          655    1328   1434   2473*
TEXTE   014642          651    1330   1442   1457   2596*
TEXTP   001232          225*   598*   649    650*   651    655*   785*   789    790*   1328*  1330   1352   1354*
                       1359*   1368   1434*  1436   1437*  1442   1456   1457   1461*
TKV     001214          216*   414*   416*   487*   518*
TKVEC = 000060          146*
TKV2    001216          217*   342*   414    415*   418*   419*
TPVEC = 000064          147*
TRAPVE= 000034          145*   283*   284*
TRTVEC= 000014          140*
TSTFLG  004614          742    756    775    782*
TST1    010060         1477*
TST10   010466         1609*
TST11   010526         1628*
TST12   010600         1659*
TST13   010716         1698*
TST14   011060         1750*
TST15   011120         1767*
TST16   011152         1784*
TST17   011212         1804*
TST2    010120         1494*
TST20   011276         1828*
```

```
TST21   011330          1846#
TST22   011430          1869#
TST23   011512          1899#
TST24   011536          1921#
TST25   011576          1943#
TST26   011650          1967#
TST27   011710          1988#
TST3    010152          1511#
TST30   011762          2020#
TST31   012074          2060#
TST4    010220          1529#
TST5    010252          1545#
TST6    010352          1569#
TST7    010434          1593#
TTYPAT  001236           227#   599*  2295  2296*  2310*  2311*  2312*  2314*
TYPDS = 104405          2650   2672  3139#
TYPE  = 104401           344    350   358   367    373    384    393    400   408   422   431   444   450
                         463    503   512   561    578    586    605    802   809   945   949   955   997
                        1047   1064  1117  1140   1163   1167   1173   1209  1265  1269  1275  1300  1326
                        1364   1393  2176  2184   2191   2202   2205   2210  2220  2240  2266  2294  2321
                        2327   2333  2347  2354   2373   2379   2433   2458  2645  2651  2670  2673  2759
                        2770   2827  2831  2842   2923   3019   3094   3135# 3187
TYPOC = 104402           954    960  1002  1172   1178   1274   1280   1369  2183  2190  2201  2332  2338
                        2353   2378  2387  2775   3136#
TYPON = 104404          3138#
TYPOS = 104403          3137#
XCSRA   001174           203#   332*  471*  559    600*   601    720*   743   753*  766*  774*  776   778*
                         783   1058  1302* 1305   1310*  1312   1323*  1344  1349  1371* 1375  1379* 1418*
                        1421   1426* 1433* 1438*  1459*  1745   1755   1789* 1793  1810* 1814  1816  1822*
                        1853*  1854* 1855  1863*  1872*  1873*  1875   1886* 1901* 1902* 1903  1923* 1924*
                        1926   1932  1945* 1946   1950   1956   2024*  2033* 2045* 2070* 2075* 2082* 2110
                        2119   2124  2127
XCVP    012232          2110#  3147
XDBRA   001176           205#   473*  737*  745*   765*   779*  1352*  1436* 1456* 1771  1835  1969* 1970*
                        1972   1978  1990* 1991   1995   2003   2025   2063
XNO     001256           237#  1324* 1333* 1395*  1397   1445*
XRCVPT  012636          2134   2166  2175#
XVT     001200           207#   334*  338*  751*  1430*  2026   2065   2131* 2132  2135  2137  2217
XVTP    001264           241#  2071  2128  2130*
XVT2    001202           208#   334   335*  339*   752*   767*   768*   1431* 2132* 2133*
$AUTOB  001134           180#
$BDADR  001122           175#
$BDDAT  001126           177#  1517* 1518* 1519   1554*  1555   1575*  1576* 1578  1816* 1835* 1855* 1856
                        1875* 1876* 1878  2119*  2120*  2121*  2122*  2123* 2126* 2131  2135  2150* 2151*
                        2152* 2153* 2154* 2157*  2163   2167   2188   2203
$CHARC  016620          2925*  2935* 2942  2951*  2956#
$CKSWR= ****** U        3142
$CMTAG  001100           163#   272   273   281    287    288
$CM1  = 000001           195#   196#
$CM2  = 000002           195#   196#
$CM3  = 000001           193#   195
$CM4  = 000001           196#   197#
$CRLF   001171           199#  2202  2294  2924   2959
$DBLK   017266          3060   3094  3102#
$DOAGN  015500          2666   2675  2681#
$DTBL   017256          3063   3098#
```

```
$ENDAD  015470          2677#
$ENDCT  015436          2668#
$ENDMG  015507          2670    2685#
$ENULL  015504          2673    2684#
$EOP    015314          2638#   3148
$EOPCT  015430          2665#   2669
$ERFLG  001103           166#   2694    2718    2720*   2737    2746*
$ERMAX  001115           172#   2737
$ERROR  015706           281    2745#   2747
$ERRPC  001116           173#   2749*   2750*   2774
$ERRTB  001306           266#
$ERTTL  001112           170#    485*    940*   1180*   1282*   2649    2748*
$FILLC  001156           191#   2928    2959
$FILLS  001155           190#   2959
$GDADR  001120           174#   1472*   1745*   2127*   2158*   2181
$GDDAT  001124           176#   2110*   2111*   2112    2113*   2114*   2115*   2116*   2141*   2142*   2143    2144*   2145*
                        2146*   2147*   2198*   2199
$GET42  015460          2674#
$GTSWR= ******  U       3141
$HD   = 000001           13      14
$HIOCT  016402          2883*   2888#
$ICNT   001104           167#    516*    931*    933     935*    1026*   1027    1031*   2726*   2727    2729*   2736
$ILLUP  017532          3157    3173    3190#
$INTAG  001135           181#
$ITEMB  001114           171#
$LF     001172           200#   2842    2959
$LPADR  001106           168#    288*   1478*   1751*   2732*   2734    2736
$LPERR  001110           169#
$MAIL = ******  U        305    2732    2912
$MXCNT  015704          2730    2736#
$NULL   001154           189#   2930    2959
$NWTST= 000001          1474#   1491#   1508#   1526#   1542#   1566#   1590#   1606#   1625#   1649#   1651    1688#   1690
                        1747#   1764#   1781#   1801#   1825#   1843#   1866#   1889#   1891    1918#   1940#   1964#   1985#
                        2010#   2012    2050#   2052
$OCNT   017046          2991*   3020*   3033#
$OMODE  017050          2986*   2990*   2995    2998*   3009*   3035#
$OVER   015670          2705    2716    2728    2733#
$PASS   001100           164#    484*   1407*   1514    1531    1548    1786    1806    1831    1849    2642*   2662*   2663*
                        2671    2684    2724    2737
$POWER  017540          3188    3193#
$PWRDN  017372           285    3157*   3185
$PWRMG  017526          3188#
$PWRUP  017444          3167    3173#
$QUES   001170           198#   2827    2959
$RDCHR  016106          2803#   3142
$RDDEC= ******  U       3145
$RDLIN  016142          2818#   3143
$RDOCT  016302          2861#   3144
$REGAD  001160           193#
$REGO   001162           195#    496
$RTNAD  015502          2683#
$R2A  = ******  U       3145
$SAVRE= ******  U       3145
$SAVR6  017536          3166*   3174    3175*   3176*   3192#
$SCOPE  015524           279    2701#
$SETUP= 000017           269#    278     279     281     283     285     287     288     2660    2702
```

```
$STUP = 177777          269#
$SVLAD  015660         2713    2731#
$SWR  = 064000            2#      13      197     198     287     288     289    1478    1495    1512    1530    1546    1570
                       1594    1610    1629    1660    1699    1751    1768    1785    1805    1829    1847    1870    1900
                       1922    1944    1968    1989    2021    2061    2635    2661    2676    2682    2684    2695    2696
                       2697    2698    2704    2716    2718    2719    2720    2721    2722    2733    2736    3189
$SWRMK= 000000         2698
$TEMPO  001274          246#    1631*   1632*   1633*   1669*   1946*   1947*   1948*   1991*   1992*   1993*
$TIMES  001166          197#    287*    1512*   1530*   1546*   1660*   1699*   1785*   1805*   1829*   1847*   2021*   2061*
                       2661*   2721*   2727    2730*   2736
$TKB    001146          186#    437*    2278    2807
$TKS    001144          185#    343*    377*    435*    449*    519*    521*    987*    1562*   1862*   2438*   2805
$TMPO   001164          196#    1729    2030*   2091
$TN   = 000032           13#    1474    1478#   1491    1495#   1508    1512#   1526    1530#   1542    1546#   1566    1570#
                       1590    1594#   1606    1610#   1625    1629#   1649    1660#   1688    1699#   1747    1751#   1764
                       1768#   1781    1785#   1801    1805#   1825    1829#   1843    1847#   1866    1870#   1889    1900#
                       1918    1922#   1940    1944#   1964    1968#   1985    1989#   2010    2021#   2050    2061#
$TPB    001152          188#    442*    2282*   2948*   2959
$TPFLG  001157          192#    2906    2959
$TPS    001150          187#    440*    2280    2946    2959
$TRAP   017276          283     3112#
$TRAP2  017320         3123#    3134
$TRP  = 000020         3127#    3136#   3137#   3138#   3139#   3140#   3142    3143#   3144#   3145#   3146    3147#   3148#
                       3149#    3150#   3151#   3152#   3153#
$TRPAD  017332         3117     3134#
$TSTNM  001102          165#    2660*   2694    2731*   2733    2737
$TTYIN  016272         2819     2820    2846    2849#
$TYPBN= ****** U       3140
$TYPDS  017052         3048#    3139
$TYPE   016404         2906#    3127    3135
$TYPEC  016554         2927     2934    2941    2946#   2947
$TYPEX  016622         2952     2954    2957#
$TYPOC  016650         2989#    3136
$TYPON  016664         2988     2991#   3138
$TYPOS  016624         2984#    3137
$XTSTR  015536         2707#
$$GET4= 000000         2676#
$OFILL  017047         2985*    2989*   2999    3034#
$40CAT= ****** U       2704
.     = 017550           17#     23#     26#     29#     32#     35#    162#    201     276     288     353#    361#    376#
                        387#    396#    466#    506#    581#    805#    812#    952#    958#   1000#   1067#   1170#   1176#
                       1272#   1278#   1367#   2179#   2223#   2243#   2324#   2330#   2357#   2382#   2436#   2599#   2618#
                       2684    2688    2736    2737    2762#   2773#   2849#   2959    3102#   3169    3191
```

```
COMMEN      1#   149#   477   613   626   662   709   795   833   842   906  1008  1038  1054  1071
         1113   1296  1317  1387  1448  1465  1737  2097  2223  2274  2363  2403  2418  2448  2466
         2621
ENDCOM      1#   149#   481   617   636   672   718   797   836   853   917  1014  1040  1056  1074
         1115   1298  1321  1391  1454  1470  1743  2108  2236  2276  2368  2410  2424  2455  2470
         2625
EOPM     2628#  2639
ERROR      43#  1487  1504  1522  1538  1558  1581  1602  1616  1622  1638  1644  1677  1723  1760
         1777   1796  1818  1838  1859  1881  1906  1929  1935  1953  1959  1975  1981  1999  2006
         2038   2085
ESCAPE      1#   149#
GETPRI      1#   149#
GETSWR      1#   149#
MTAGS     152#   202
MULT        1#   149#
NEWTST      1#   149#  1474  1491  1508  1526  1542  1566  1590  1606  1625  1649  1688  1747  1764
         1781   1801  1825  1843  1866  1889  1918  1940  1964  1985  2010  2050
POP         1#   149#  2884  3089  3178  3179
PUSH        1#   149#  2863  3048  3159  3165
REPORT      1#   149#
SCOPE      44#  1494  1511  1529  1545  1569  1593  1609  1628  1659  1698  1767  1784  1804  1828
         1846   1869  1899  1921  1943  1967  1988  2020  2060
SCOPM    2689#  2702
SETPRI      1#   149#
SETTRA   3127#  3136  3137  3138  3139  3142  3143  3144  3146  3147  3148  3149  3150  3151  3152
SETUP       1#   149#   270
SKIP        1#   149#
SLASH       1#   149#
SPACE     149#
STARS       1#   149#   158   201  1474  1476  1491  1493  1508  1510  1526  1528  1542  1544  1566
         1568   1590  1592  1606  1608  1625  1627  1649  1658  1688  1697  1747  1749  1764  1766
         1781   1783  1801  1803  1825  1827  1843  1845  1866  1868  1889  1898  1918  1920  1940
         1942   1964  1966  1985  1987  2010  2019  2050  2059  2631  2691  2740  2790  2811  2853
         2891   2961  3038  3106  3155  3171
SWRSU       1#   149#   289#
TRMTRP   3127#
TYPBIN      1#   149#
TYPDEC      1#   149#  2649  2671
TYPNAM      1#   149#
TYPNUM      1#   149#
TYPOCS      1#   149#
TYPOCT      1#   149#   953   959  1001  1171  1177  1273  1279  1368  2181  2188  2199  2331  2337
         2352   2377  2386  2774
TYPTXT      1#   149#   344   350   358   367   372   383   393   400   408   421   431   444   450
          463    503   511   561   577   585   604   802   808   945   949   955   997  1047  1064
         1163   1167  1173  1265  1269  1275  1364  2175  2184  2191  2205  2209  2220  2240  2265
         2320   2327  2333  2346  2354  2373  2379  2432  2457  2644  2651  2759  2769
WARN      154#  1651  1690  1891  1908  2012  2052
$BODT       1#
$$CMRE    156#   195
$$CMTM    156#   196
$$ESCA      1#   149#
$$NEWT      1#   149#  1474  1491  1508  1526  1542  1566  1590  1606  1625  1649  1688  1747  1764
         1781   1801  1825  1843  1866  1889  1918  1940  1964  1985  2010  2050
$$SET    3127#  3136  3137  3138  3139  3142  3143  3144  3146  3147  3148  3149  3150  3151  3152
$$SKIP      1#   149#
```

```
.EQUAT      1#     39
.HEADE      1#      3
.KT11       1#
.SETTR      1#
.SETUP      1#    269
.SWRHI      1#
.TRMTR      1#
.$ACTI      1#
.$APTB      1#
.$APTH      1#
.$APTY      1#
.$ASTA      1#
.$CATC      1#
.$CMTA      1#    156
.$DB2D      1#
.$DB20      1#
.$DIV       1#
.$EOP       1#   2629
.$ERRO      1#
.$ERRT      1#
.$MULT      1#
.$POWE      1#   3153
.$RAND      1#
.$RDDE      1#
.$RDOC      1#   2851
.$READ      1#
.$R2AZ      1#
.$SAVE      1#
.$SB2D      1#
.$SB20      1#
.$SCOP      1#   2689
.$SIZE      1#
.$SUPR      1#
.$TRAP      1#   3104
.$TYPB      1#
.$TYPD      1#   3036
.$TYPE      1#   2889
.$TYPO      1#   2959
.$40CA      1#
.1170       1#
```

```
                919   923   927  2284  2287  2291  2299  2302  2306  2308  2825  2832  2834  2918  2920
               2928  2949  2953
COM             696   706
DEC             703  1196  1258  1407  2664
DECB           2932  2935  3009  3020
EMT              43
HALT            507  2214  2395  2428  2764  2908  3168  3190
INC             685   700   931   940  1026  1180  1282  1409  1632  1947  1992  2642  2662  2726  2748
               2755  3015  3023  3066  3176
INCB           2731  2746  2955
IOT              44
JMP              30    33    36   362   508   526   530   534   538   542   549   609   620   624   643
              647   653   660   698   723   727   731   733   738   746   770   780   806   828   840
              869   878   896   904   921   925   929   939   974   988  1005  1033  1044  1052  1060
             1069  1154  1157  1221  1259  1341  1358  1413  1446  1733  2095  2134  2166  2286  2289
             2682  2839
JSR             380   428   456   469   742   756   775   786  1020  1156  1357  2677  2927  2934  2941
MOV             273   277   279   280   281   282   283   284   285   286   288   291   292   293   294
              299   301   302   303   308   316   319   326   330   334   335   336   337   342   355
              379   382   390   414   415   416   418   419   427   435   437   440   442   455   458
              460   468   471   473   483   487   489   495   496   498   499   518   522   548   567
              583   591   598   600   649   655   658   659   737   745   751   752   755   765   767
              768   779   789   815   816   822   868   877   886   887   895   901   953   959   978
              985  1001  1004  1025  1051  1068  1076  1077  1079  1080  1081  1082  1083  1092  1093
             1094  1095  1096  1103  1104  1105  1106  1107  1138  1142  1150  1171  1177  1211  1232
             1233  1235  1236  1246  1247  1273  1279  1290  1324  1328  1352  1368  1395  1430  1431
             1434  1436  1456  1472  1478  1480  1481  1489  1496  1497  1506  1512  1517  1530  1546
             1551  1554  1571  1575  1631  1660  1665  1666  1668  1669  1670  1699  1703  1704  1705
             1706  1709  1726  1728  1729  1745  1751  1753  1754  1762  1769  1770  1779  1785  1805
             1809  1816  1829  1835  1847  1852  1855  1871  1875  1946  1991  2021  2026  2027  2029
             2030  2031  2061  2065  2066  2067  2068  2071  2088  2090  2091  2110  2112  2119  2124
             2127  2130  2131  2132  2141  2143  2150  2155  2158  2162  2163  2164  2181  2188  2199
             2278  2295  2326  2331  2337  2352  2359  2371  2377  2384  2386  2414  2649  2667  2671
             2674  2709  2710  2712  2715  2729  2730  2732  2733  2734  2749  2774  2803  2804  2818
             2819  2836  2837  2838  2843  2844  2845  2846  2861  2862  2863  2864  2865  2867  2882
             2883  2884  2885  2886  2910  2911  2915  2930  2984  2992  2993  2994  3000  3007  3025
             3026  3027  3028  3029  3049  3050  3051  3052  3053  3054  3055  3060  3063  3083  3089
             3090  3091  3092  3093  3095  3096  3112  3113  3117  3123  3124  3157  3158  3159  3160
             3161  3162  3163  3164  3165  3166  3167  3173  3174  3178  3179  3180  3181  3182  3183
             3184  3185  3186
MOVB            523   573   592  1612  1633  1924  1948  1970  1993  2282  2807  2824  2829  2870  2912
             2940  2948  2985  2986  2989  2990  2991  2995  2998  2999  3018  3058  3061  3075  3078
             3087  3115
NEG             2996  3057
NOP              754   902  1240  1440  1477  1675  1711  1750  2036  2073  2678  2679  2680
RESET            488  2462  2676
ROL              695  2873  2875  2877  3002  3004  3005  3006  3008
RTI              300  1084  1097  1108  1293  1462  2139  2171  2226  2297  2339  2360  2393  2415  2442
             2463  2656  2735  2786  2809  2847  2887  2917  3030  3097  3125  3189
RTS              792  1185  1194  1382  2249  2262  2272  2957  3118
SEC             2271
SUB              785   823   980  1333  1359  1445  2215  2372  2385  2750  3064
SWAB            2121  2152
TRAP            3127  3136  3137  3138  3139  3142  3143  3144  3146  3147  3148  3149  3150  3151  3152
TST              491   500   559   601   683   783   799   821  1042  1058  1120  1126  1130  1189  1192
             1213  1223  1228  1305  1312  1344  1375  1421  1482  1498  1514  1531  1548  1596  1664
```

```
                1686  1701  1727  1755  1771  1786  1806  1831  1849  2025  2063  2089  2246  2253  2412
                2425  2439  2711  2724  2778  2881  2914  2922  2944  3013  3069  3079  3114
TSTB             688   691   776   875  1147  1599  1903  2280  2718  2805  2906  2946  3071  3085
WAIT             890
XOR              313
.ASCII           198   199
.ASCIZ           200   347   353   361   370   376   387   396   403   411   425   434   447   453   466
                 506   515   564   581   589   608   805   812   948   952   958  1000  1050  1067  1166
                1170  1176  1268  1272  1278  1367  2179  2187  2194  2208  2213  2223  2243  2269  2324
                2330  2336  2350  2357  2376  2382  2436  2461  2606  2609  2613  2648  2654  2685  2762
                2773  3193
.BLKB           2849
.BLKW           2599  3102
.BYTE            165   166   171   172   180   181   189   190   191   192  2684  2848  3031  3032  3033
                3034
.ENABL             1
.END            3196
.ENDC              8    43   135   149   159   163   165   193   196   197   198   202   252   269   277
                 278   281   283   285   287   288   289   305   347   353   361   370   376   387   396
                 403   411   425   434   447   453   466   478   481   506   515   564   581   589   608
                 614   617   627   636   663   672   710   718   796   797   805   812   834   836   843
                 853   907   917   948   952   958  1000  1009  1014  1039  1040  1050  1055  1056  1067
                1072  1074  1114  1115  1166  1170  1176  1268  1272  1278  1297  1298  1318  1321  1367
                1388  1391  1449  1454  1466  1470  1475  1476  1477  1478  1479  1492  1493  1494  1495
                1509  1510  1511  1512  1513  1527  1528  1529  1530  1531  1543  1544  1545  1546  1547
                1567  1568  1569  1570  1591  1592  1593  1594  1607  1608  1609  1610  1626  1627  1628
                1629  1650  1651  1658  1659  1660  1661  1689  1690  1697  1698  1699  1700  1738  1743
                1748  1749  1750  1751  1752  1765  1766  1767  1768  1782  1783  1784  1785  1786  1802
                1803  1804  1805  1806  1826  1827  1828  1829  1830  1844  1845  1846  1847  1848  1867
                1868  1869  1870  1890  1891  1898  1899  1900  1919  1920  1921  1922  1941  1942  1943
                1944  1965  1966  1967  1968  1986  1987  1988  1989  2011  2012  2019  2020  2021  2022
                2051  2052  2059  2060  2061  2062  2098  2108  2179  2187  2194  2208  2213  2223  2230
                2236  2243  2269  2275  2276  2324  2330  2336  2350  2357  2364  2368  2376  2382  2404
                2410  2419  2424  2436  2449  2455  2461  2467  2470  2622  2625  2632  2634  2635  2637
                2648  2654  2660  2666  2669  2670  2674  2676  2682  2684  2685  2688  2692  2695  2698
                2704  2706  2717  2718  2720  2722  2726  2731  2732  2733  2736  2737  2741  2762  2773
                2791  2812  2854  2856  2889  2892  2912  2962  3039  3107  3113  3116  3135  3136  3137
                3138  3139  3140  3141  3142  3143  3144  3145  3146  3147  3148  3149  3150  3151  3152
                3156  3165  3166  3172  3178  3179  3189  3196
.EQUIV            43    44    52    97    98    99   100   101   102   103   104   105   106   125   126
                 127   128   129   130   131   132   133   134
.EVEN            347   353   361   370   376   387   396   403   411   425   434   447   453   466   506
                 515   564   581   589   608   805   812   948   952   958  1000  1050  1067  1166  1170
                1176  1268  1272  1278  1367  2179  2187  2194  2208  2213  2223  2243  2269  2324  2330
                2336  2350  2357  2376  2382  2436  2461  2618  2648  2654  2762  2773  3195
.IF                4    41   107   135   158   162   164   193   196   197   198   201   202   269   272
                 277   279   281   283   285   287   288   305   346   352   360   369   375   386   395
                 402   410   424   433   446   452   465   478   481   505   514   563   580   588   607
                 614   617   627   636   663   672   710   718   796   797   804   811   834   836   843
                 853   907   917   947   951   957   999  1009  1014  1039  1040  1049  1055  1056  1066
                1072  1074  1114  1115  1165  1169  1175  1267  1271  1277  1297  1298  1318  1321  1366
                1388  1391  1449  1454  1466  1470  1474  1476  1478  1479  1491  1493  1495  1508  1510
                1512  1513  1526  1528  1530  1531  1542  1544  1546  1547  1566  1568  1570  1590  1592
                1594  1606  1608  1610  1625  1627  1629  1649  1651  1658  1660  1661  1688  1690  1697
                1699  1700  1738  1743  1747  1749  1751  1752  1764  1766  1768  1781  1783  1785  1786
                1801  1803  1805  1806  1825  1827  1829  1830  1843  1845  1847  1848  1866  1868  1870
```

```
                1889  1891  1898  1900  1918  1920  1922  1940  1942  1944  1964  1966  1968  1985  1987
                1989  2010  2012  2019  2021  2022  2050  2052  2059  2061  2062  2098  2108  2178  2186
                2193  2207  2212  2222  2230  2236  2242  2268  2275  2276  2323  2329  2335  2349  2356
                2364  2368  2375  2381  2404  2410  2419  2424  2435  2449  2455  2460  2467  2470  2622
                2625  2631  2632  2633  2634  2635  2636  2637  2639  2647  2653  2665  2668  2670  2674
                2676  2682  2684  2685  2691  2694  2698  2702  2704  2716  2718  2719  2720  2722  2724
                2732  2733  2735  2736  2737  2740  2761  2772  2790  2811  2853  2856  2868  2891  2912
                2961  3038  3106  3112  3116  3127  3136  3137  3138  3139  3140  3141  3142  3143  3144
                3145  3146  3147  3148  3149  3150  3151  3152  3155  3165  3166  3171  3178  3179  3187
                3189  3193
.IFF              41   159   162   164   193   202   277  1474  1475  1476  1477  1478  1492  1493  1494
                1495  1509  1510  1511  1512  1527  1528  1529  1530  1543  1544  1545  1546  1567  1568
                1569  1570  1591  1592  1593  1594  1607  1608  1609  1610  1626  1627  1628  1629  1650
                1651  1659  1660  1661  1689  1690  1698  1699  1700  1747  1748  1749  1750  1751  1765
                1766  1767  1768  1782  1783  1784  1785  1802  1803  1804  1805  1826  1827  1828  1829
                1830  1844  1845  1846  1847  1867  1868  1869  1870  1890  1891  1899  1900  1919  1920
                1921  1922  1941  1942  1943  1944  1965  1966  1967  1968  1986  1987  1988  1989  2011
                2012  2020  2021  2022  2051  2052  2060  2061  2062  2632  2636  2639  2665  2668  2684
                2698  2716  2718  2720  2736  2737  2741  2791  2812  2854  2892  2962  3039  3107  3113
                3155  3172  3189
.IFT              347   353   361   370   376   387   396   403   411   425   434   447   453   466   506
                 515   564   581   589   608   805   812   948   952   958  1000  1050  1067  1166  1170
                1176  1268  1272  1278  1367  2179  2187  2194  2208  2213  2223  2243  2269  2324  2330
                2336  2350  2357  2376  2382  2436  2461  2648  2654  2722  2762  2773  2872  2888  2889
.IFTF             347   353   361   370   376   387   396   403   411   425   434   447   453   466   506
                 515   564   581   589   608   805   812   948   952   958  1000  1050  1067  1166  1170
                1176  1268  1272  1278  1367  2179  2187  2194  2208  2213  2223  2243  2269  2324  2330
                2336  2350  2357  2376  2382  2436  2461  2648  2654  2720  2762  2773  2868  2872  2888
.IIF                3     8    13    14   201   278   281   287   288   289   954   960  1002  1172  1178
                1274  1280  1369  2182  2189  2200  2332  2338  2353  2378  2387  2634  2650  2660  2661
                2672  2684  2688  2695  2696  2697  2698  2702  2721  2733  2736  2737  2775  2959  3135
                3136  3137  3138  3139  3142  3143  3144  3146  3147  3148  3149  3150  3151  3152
.IRP              202   269  1474  1491  1508  1526  1542  1566  1590  1606  1625  1649  1688  1747  1764
                1781  1801  1825  1843  1866  1889  1918  1940  1964  1985  2010  2050  2639  2702  2863
                2884  3049  3089  3159  3165  3178  3179
.LIST               1    23   149   193   195   196   197   269   289   347   353   361   370   376   387
                 396   403   411   425   434   447   453   466   477   482   506   515   564   581   589
                 608   613   618   626   637   662   673   709   719   795   798   805   812   833   837
                 842   854   906   918   948   952   958  1000  1008  1015  1039  1041  1050  1054  1057
                1067  1071  1075  1113  1116  1166  1170  1176  1268  1272  1278  1296  1299  1317  1322
                1367  1387  1392  1448  1455  1465  1471  1474  1478  1491  1495  1508  1512  1526  1530
                1542  1546  1566  1570  1590  1594  1606  1610  1625  1629  1649  1652  1660  1688  1691
                1699  1737  1744  1747  1751  1764  1768  1781  1785  1801  1805  1825  1829  1843  1847
                1866  1870  1889  1892  1900  1909  1918  1922  1940  1944  1964  1968  1985  1989  2010
                2013  2021  2050  2053  2061  2097  2109  2179  2187  2194  2208  2213  2223  2229  2237
                2243  2269  2274  2277  2324  2330  2336  2350  2357  2363  2369  2376  2382  2403  2411
                2418  2425  2436  2448  2456  2461  2466  2471  2621  2626  2648  2654  2660  2676  2698
                2762  2773  3127  3135  3136  3137  3138  3139  3140  3142  3143  3144  3145  3146  3147
                3148  3149  3150  3151  3152  3153
.MACRO              1   152   154   156  2628  2689  3127
.MCALL              1   149   289
.NLIST              1    23   149   193   195   196   197   269   289   347   353   361   370   376   387
                 396   403   411   425   434   447   453   466   477   482   506   515   564   581   589
                 608   613   618   626   637   662   673   709   719   795   798   805   812   833   837
                 842   854   906   918   948   952   958  1000  1008  1015  1038  1041  1050  1054  1057
                1067  1071  1075  1113  1116  1166  1170  1176  1268  1272  1278  1296  1299  1317  1322
```

```
            1367    1387    1392    1448    1455    1465    1471    1474    1478    1491    1495    1508    1512    1526    1530
            1542    1546    1566    1570    1590    1594    1606    1610    1625    1629    1649    1652    1660    1688    1691
            1699    1737    1744    1747    1751    1764    1768    1781    1785    1801    1805    1825    1829    1843    1847
            1866    1870    1889    1892    1900    1909    1918    1922    1940    1944    1964    1968    1985    1989    2010
            2013    2021    2050    2053    2061    2097    2109    2179    2187    2194    2208    2213    2223    2229    2237
            2243    2269    2274    2277    2324    2330    2336    2350    2357    2363    2369    2376    2382    2403    2411
            2418    2425    2436    2448    2456    2461    2466    2471    2621    2626    2648    2654    2660    2676    2698
            2762    2773    3127    3135    3136    3137    3138    3139    3140    3142    3143    3144    3145    3146    3147
            3148    3149    3150    3151    3152    3153
.PAGE        156     252
.REPT         23     195     196
.SBTTL        15      39     156     252     268     271     475     611     830    1035    1110    1134    1204    1294    1315
            1384    1464    1474    1491    1508    1526    1542    1566    1590    1606    1625    1649    1688    1735    1747
            1764    1781    1801    1825    1843    1866    1889    1918    1940    1964    1985    2010    2050    2629    2689
            2739    2791    2851    2889    2959    3036    3104    3127    3153
.TITLE        3
.WORD        23      24      25     164     167     168     169     170     173     174     175     176     177     178     179
            182     183     184     193     195     196     203     205     207     208     210     211     213     214     216
            217     219     220     221     223     224     225     226     227     228     229     230     232     233     234
            235     237     238     240     241     242     244     245     246     247     248     249     250    1415    2341
           2400    2401    2600    2665    2668    2683    2888    2956    3035    3134    3188
```

```
ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

*,DZDRH.SEQ/SOL/CRF/PAGNUM←LIB:SYSMAC.SML,DSK:DZDRH
RUN-TIME:  66 49 6 SECONDS
RUN-TIME RATIO:  205/122=1.6
CORE USED:  34K  (67 PAGES)
```

Spooler runtime 12 Seconds, 52 KCS, 354 disk reads, 4 disk writes, 90 pages