IDENTIFICATION

Product Code:	MAINDEC-11-DZDLC-B-D
Product Name:	DL11/C,/D, or /E Off Line Test
Date Created:	MAY 1977
Maintainer:	Diagnostic RELEASE ENGINEERING
Author:	E. Crowley/B. Burgess

TABLE OF CONTENTS

1.0   PROGRAM PURPOSE (ABSTRACT)

This program has the ability to test the DL11 (Asynchronous Modem Interface), off line. Models able to be tested are C, D, and E only. The use of a modem is not required for testing; however, a special cable connector BC05C and a special modem test connector H315A is required. This program is capable of the following:

a. Verification of maintenance bit
b. Verification that transmitter can cause an interrupt
c. Verification that receiver 'DONE' can cause an interrupt
d. Checks that 'REQ TO SEND' asserts 'RING'
e. Checks that 'SEC XMIT' asserts 'SEC REC' and 'DATA SET INT'
f. Checks that 'DTR' can assert 'CLR TO SEND' and 'CAR DET'
g. Verifies that 'DATA SET I.E.' can cause a RECVR INTR
h. Checks the 'BREAK' feature
i. Performs null-del-null pattern
j. Performs binary up count pattern
k. Performs binary down count pattern
l. Runs a worse case pattern

Included in the program are special user routines - PRG #2, PRG #3, PRG #4, and PRG #5 (which will be described further into this document).

Note well two(2) points:

1. This program is capable of testing sixteen(16) DL11's and assumes contiguous addressing from 1st device to last.

    a. If multiple devices are not being tested, thus not requiring a pass thru the program once per device, then the program will default to testing the 1st possible DL11-E device i.e., RCSR address = 775610, and test this device only.

    b. If multiple device testing is not being conducted, and the device existing is not the default DL11-E, then the user on starting the program will have to set SW<0>=1 to enter the question & answer mode.

2. This program has provision for character length i.e., it assumes data is 8 bits, but also has the ability to handle 5, 6, or 7 bits of data as well.

2.0   SYSTEM REQUIREMENTS

a. Hardware Requirements

    PDP-11 family processor with 8K of memory
    M7800 DL11 asynchronous line interface module

BC05C special cable connector
H315A special modem test connector

b.   Software Requirements

This program was specifically designed for the 11/40
Front End of the 1080 Console Processor System. In this
environment it would be loaded by the TCDP (Dectape)
diagnostic monitor. However, any 11/40 user with 8K of
memory can run this program to test one(1) or multiple
DL11's.

The program has the proper interface code to allow
running under the automated manufacturing test line
system - ACT11.

3.0   RELATED DOCUMENTS AND STANDARDS

   a.   Programming Practices - Document No. 175-003-009-00
   b.   PDP11/40 Processor Handbook
   c.   DL11 Asynchronous Line Interface Manual
            Document No. DEC-11-HDLAA
   d.   PDP-11 Maindec SYSMAC' Utility Package
            MAINDEC-11-DZQAC-C3
   e.   Applicable Circuit Schematic
            M7800

4.0   DIAGNOSTIC HIERARCHY PREREQUISITES

Before running this program, the following two(2) diagnostic
programs should be run for verification of functionality of
the 11-instruction set and memory:

   1.   MAINDEC-11-DBQEA and,
   2.   MAINDEC-11-DZQMC

5.0   LOADING AND STARTING PROCEDURE

Load program in memory using ABS loader
Load address 200.

NOTE

In the case of a 1080 system environment
load the program using the TCDP
(dectape) Diagnostic Monitor.

Press start.

   a.   There are also three(3) optional start addresses for the
        program:

```
204 - selects program #2
210 - selects program #3
214 - selects program #4
220 - selects program #5
```

6.0   SPECIAL ENVIRONMENTS

If this program is run in Quick Verify Mode under ACT11 the
program is done after the first pass.

7.0   PROGRAM OPTIONS

SWITCH	USE
15=1 or up	Halt on error
14=1 or up	Loop on test
13=1 or up	Inhibit error typeouts
12=1 or up	/C or /D model being tested
11=1 or up	Inhibit iterations
10=1 or up	Bell on error
9=1 or up	Loop on error
8=1 or up	Loop on test in SWR<7:0>
<7:0>	Holds test no. of test to be looped on. Used in conjunction with SW<8>.
0=1 or up	Used in device table creation (1 to 16 devices) i.e., default device not desired. Also used for character length setting.

            !! NOTE WELL !!

If sw<08> is set the user can only 'loop
on a test' of the default device i.e. -
DL11/E RCSR = 775610. If the user
desires to 'loop on a test' of other
than the default device he must first
patch the five (5) locations labeled

   DLRCSR;   DLRDBR;   DLXCSR;   DLXDBR;
DLVECT;

that appear under 'DL11 Definitions'
heading at the front of the listing.
i.e. - with sw<08> set sw<00> is not
functional.

8.0   EXECUTION TIMES

Execution time is dependent on type of memory and

number of DL11's being tested. A representative
time for 1 error free pass is:

11/40 - core memory - 1 DL11/E - 20 seconds

9.0   ERROR INFORMATION

9.1   Error Reporting

There are a total of seven(7) types of error
reports generated by the program. The key column
headings will be described below for clarity -

DEVADR - This is the address of the receiver
           control status register for the failing
           DL11

REGADR - This is the address of the DL11
           register on which testing is being
           conducted

WAS   - This is what the contents of the
           register of the DL11 undergoing test
           was (address is under column '(R2)')

S/B   - This is what the contents of the
           register of the DL11 undergoing test
           should be (address is under column
           '(R2)')

WASADR - This is what the memory address was
           (input data buffer address)

SHBADR - This is what the memory address should
           be (output data buffer address)

(REG)   - This is the contents of the DL11
           receiver data buffer in error (address
           is under column '(R2)')

9.2   Error Halts

With the 'Halt on Error' switch (SW15) not set there are
four(4) programmed 'HALTS' in the program:

a. In the case of error reporting and there is no terminal
    to allow the information transfer.

b. In the power fail routine if the power up sequence was

started before the power down sequence had a chance to complete itslef.

c. In the end of pass routine if multiple device testing is being conducted but no devices are shown as active.

d. In the case of sw<08> being set.

10.0   PERFORMANCE AND PROGRESS REPORTS

Not applicable.

11.0   DEVICE INFORMATION TABLES

a. The following is a picture view of a DL11-E Receiver Control Status Register, which will show bit assignments and definitions, to provide a handy reference:

```
===================================================
 I  I  I  I  I  I  I  I  I  I  I  I  I  I  I  I
|DS|RI|CT|C |R |S |  |R |RI|DI|  |S |RT|DT|  |
|I |NG| S| D| A| R|  | D|EN|EN|  | X|S |R |  |
 I  I  I  I  I  I  I  I  I  I  I  I  I  I  I  I
===================================================
 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
```
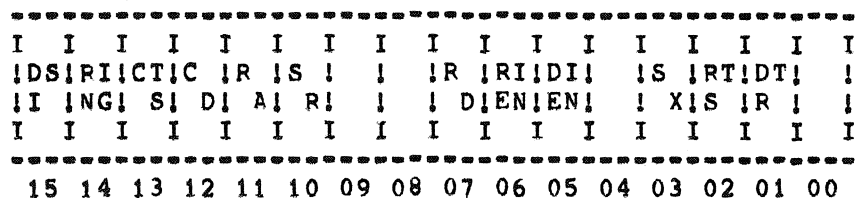
Bit assignments are defined as follows:

BIT15	Data Set Interrupt	1. Interrupt sequence initiated when BIT05 set. 2. Sets whenever bits 10, 11, 12 or 14 change state 3. Cleared by INIT or reading RCSR
BIT14	Ring	1. When set, indicates a control signal being received from dataset.
BIT13	Clear to send	1. When set indicates ON condition; when clear indicates OFF condition. 2. Dependent on state of 'CTS' signal from dataset
BIT12	Carrier Detect	1. Sets when data carrier received 2. When clear indicates end of current transmission or an error condition.

BIT11 Receiver Active

1. When set indicates receiver interface is active.
2. Cleared by INIT or RCVR DONE (BIT07).

BIT10 Secondary Receive or Supervisory Received Data

1. Provides receive capability, when set, for reverse channel of remote station. Sets when BIT03 is set.
2. Cleared by INIT

BIT07 Receiver Done

1. Sets when character has been received. Will initiate an interrupt providing BIT06 is also set
2. Cleared when RDBR is addressed or BIT00 is set.
3. Also, cleared by INIT

BIT06 Receiver Interrupt Enable

1. When set, allows interrupt providing BIT07 is set.
2. Cleared by INIT
3. ***READ/WRITE BIT***

BIT05 Dataset Interrupt Enable

1. When set, allows interrupt providing BIT15 is set.
2. Cleared by INIT
3. ***READ/WRITE BIT***

BIT03 Secondary Transmit or Supervisory Transmitted DATA

1. Provides transmit capability, when set, for reverse channel of remote station. Sets when BIT10 is set.
2. Cleared by INIT
3. ***READ/WRITE BIT***

BIT02 Request to Send

1. Jumper ties this bit to REQ TO SEND in dataset.
2. Required for transmission
3. Cleared by INIT
4. ***READ/WRITE BIT***

BIT01 Data Terminal Ready

1. When set, permits connection to channel.
2. When clear, disconnects interface from channel.
3. MUST be cleared by program
4. ***READ/WRITE BIT***

Special Notes on RCSR Register

1. Addresses should fall in the range of 175610 to

176170

2. BIT01 (Data terminal Ready) state is not defined after power-up.

3. On DL11-C or -D options bits 15, 14, 13, 12, 10, 5, 3, 2, and 1 are not used.

4. On DL11-C and -D options bit<00> is 'RDR ENB' . On a DL11-E option this bit is unused.

b. The following is a picture view of a DL11-E transmitter control status register, which will show bit assignments and definitions, to provide a handy reference:

```
-----------------------------------------------------
I  I  I  I  I  I  I  I  I  I  I  I  I  I  I  I
!  !  !  !  !  !  !  ! |XR|XI|  !  ! |M |  |BR|
!  !  !  !  !  !  !  ! |DY|EN|  !  ! | A|  | K|
I  I  I  I  I  I  I  I  I  I  I  I  I  I  I  I
-----------------------------------------------------
 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
```

Bit assignments are defined as follows:

BIT07	Transmitter Ready	1. Set when XDBR can accept another character. Will initiate an interrupt if BIT06 also set. 2. Also set by INIT 3. Cleared by loading XDBR
BIT06	Transmitter Interrupt Enable	1. When set, allows interrupt providing BIT07 is set. 2. Cleared by INIT 3. ***READ/WRITE BIT***
BIT02	Maintenance	1. When set, disables serial line input to receiver & connects XMIT output to receiver input which disconnects external device input. This forces receiver to run at xmitter speed. 2. Cleared by INIT 3. ***READ/WRITE BIT***
BIT00	Break	1. When set, transmits a continuous space to external device 2. Cleared by INIT 3. ***READ/WRITE BIT***

!! NOTE !!

DL11-C and -D options are the same.

c. The following is a picture view of the DL11-E receiver and transmitter data buffer registers, to provide a handy reference.

```
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
I  I  I  I  I  I  I  J  I  I  I  I  I  I  I  I
|ER|OR|FR|PE|  |  |  |  |  |D |A |T |A |  |  |
I  I  I  I  I  J  I  I  I  I  I  I  I  I  I  I
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
```

Bit assignments are defined as follows:

BITS 07-00   Data   1. Character   to   be
                                         transferred   to   external
                                         device.
                                      2. If character less than 8
                                         bits it must be loaded
                                         right justified.
                                      3. ***WRITE ONLY BITS***

BIT 15   Error   1. ***READ ONLY BIT***
                                      2. Cleared by error removal

BIT 14   Overrun   1. Same as BIT 15
                                      2. RCVR DONE not cleared

BIT 13   Framing   1. Same as BIT 15
                                      2. No valid STOP bit

BIT 12   parity   1. Same as BIT 15
                                      2. Parity   other   than
                                         expected

NOTE: Bits<15:12> only appear in the rcvr data buffer
         DL11-C and -D options are the same.

12.0   SUBROUTINE SUMMARIES

12.1   DLADDR

This routine sets up the following:

        RCSR - Receiver Status Register
        RBUF - Receiver Buffer Register
        XCSR - Transmitter Status Register
        XBUF - Transmitter Buffer Register

The setup is done, initially, in response to user reply to 1st device he wants tested, and thereafter, at the end of a program pass to allow cycling thru all devices for multiple device testing (if required).

12.2 $EOP

This routine is supplied by MAINDEC-11-DZQACC3, the PDP-11 Maindec 'SYSMAC' utility package. This routine is responsible for the following:

a.   Incrementing the pass number ($PASS)
b.   Typing 'END PASS # XXX' (where 'XXX' is a decimal value)

NOTE

If multiple device testing is being conducted, then $PASS is only incremented after testing of all devices has transpired (multiple testing). Therefore, e.g., If 10 devices have been tested then 'END PASS #1' would be typed out; 'END PASS #2' would be typed out after the 10 devices have once again been tested by the program, etc.

c.   Goes to a monitor, if there is one
d.   If there is no monitor transfers control back to beginning of the program.

12.3 $SCOPE

This routine is supplied by MAINDEC-11-DZQAC-C3, the PDP-11 Maindec 'SYSMAC' utility package.

This routine is entered before and after every subtest to ascertain the following conditions:

a.   Loop on test just executed?
      This condition is enabled when SW<14> is set to a '1'.

b.   Loop on test if an error has occurred during the test?
      This condition is enabled when SW<09> is set to a '1'.

c.   Loop on test specified by test no. appearing in SWR<7:0>?
      This condition is enabled when SW<08> is set to a '1'.

d.   Inhibit subtest iterations?
      This condition is enabled when SW<11> is set to a '1'.

12.4 $ERROR

This routine is supplied by MAINDEC-11-DZQAC-C3, the PDP-11 Maindec 'SYSMAC' utility package.

This routine handles the following reactions to an error when an error is encountered:

a. 'HALT' on Error?
   This condition is enabled when SW<15> is set to a '1'.

b. Ring 'Bell' on Error?
   This condition is enabled when SW<10> is set to a '1'.

c. Loop on Error
   This condition is enabled when SW<09> is set to a '1'.

d. Inhibit Error Typeouts
   This condition is enabled when SW<13> is set to a '1'.

NOTE

On encountering an error while executing the program this routine will transfer control to '$ERRTYP' routine shown below (presumes 'HALT' on error SW<15> not set).

12.5 $ERRTYP

This routine is supplied by MAINDEC-11-DZQAC-C3, the PDP-11 Maindec 'SYSMAC' utility package.

This routine handles the information for error message typeouts as follows:

This routine uses the 'Item Control Byte' ($ITEMB) to determine which error is to be reported. It then obtains, from the 'error Table' ($ERRTB) the addresses of where the information, for printout, is stored; and causes the appropriate information concerning the error to be printed out.

Note: 1. The variable '$ITEMB' is supplied by .SCMTAG, a 'SYSMAC' utility package routine.

       2. The 1st address '$ERRTB' for location of 'error table' information is also supplied by .SCMTAG.

       3. If the '$ITEMB' value is zero(0), then only a program counter (PC) is printed out. It has no label, it is a pure number.

12.6 $TYPOC

This routine is supplied by MAINDEC-11-DZQAC-C3, the PDP-11 Maindec 'SYSMAC' utility package. This routine is used for all octal typeouts (16 bit values) throughout the program.

12.7 $TYPDS

This routine is supplied by MAINDEC-11-DZQAC-C3, the PDP-11 Maindec 'SYSMAC' utility package. This routine is used to type a decimal value at the end of a pass of the program of the form 'END PASS # XXX' where 'XXX' is the decimal value.

12.8 $RDCHR, $RDLIN, $RDOCT

These routines are supplied by MAINDEC-11-DZQAC-C3, the PDP-11 Maindec 'SYSMAC' utility package. Their uses are as follows:

a. $RDCHR - Handles a single character coming in from the TTY. The character is placed on top of the stack for future use.

b. $RDLIN - Handles a string of characters coming in from the TTY. The address of the 1st character is placed on top of the stack for future use.

c. $RDOCT - Handles an octal number coming in from the $RDDEC 104420 TTY decimal # input TTY. Low order bits are stored on top of the stack; high order bits are stored in location $HIOCT. $HIOCT is supplied by .$CMTAG, a 'SYSMAC' package utility routine.

12.9 $TYPE

This routine is supplied by MAINDEC-11-DZQAC-C3, the PDP-11 Maindec 'SYSMAC' utility package. This routine is used to type ASCII messages (which must terminate with a 0 byte) as well as all other forms of typed information. The routine is also responsible for inserting a number of fill characters after a line feed.

Note: 1. $NULL contains the character to be used as fill.

2. $FILLS contains the number of filler characters req'd.

3. $FILLC contains the character to fill after.

4. The above three(3) variables are supplied by .SCMTAG, a 'SYSMAC' package utility routine.

12.10 $TRAP, $TRPAD

These routines are supplied by MAINDEC-11-DZQAC-C3, the PDP-11 Maindec 'SYSMAC' utility package. The '$TRAP' routine will strip off the lower byte of a trap instruction and use it to index thru the trap table ($TRPAD) for the starting address of the desired routine. Then using the address obtained it will then transfer program control to that routine.

The following table defines all routines in the program called by a 'TRAP' instruction by showing their 'TRAP' equivalences -

$TYPE	104400	TTY typeout routine
$TYPOC	104402	Type octal # (with leading zeros)
$TYPOS	104404	Type octal # (no leading zeros)
$TYPON	104406	Type octal # (per last character method)
$TYPDS	104410	Type decimal # (with sign)
$RDCHR	104412	TTY character input
$RDLIN	104414	TTY string input
$RDOCT	104416	TTY octal # input

12.11 $PWRDN, $PWRUP

These routines are supplied by MAINDEC-11-DZQAC-C3, the PDP-11 Maindec 'SYSMAC' utility package. These routines handle the 'Power Down and Up' sequence. The program may be power failed when running; however, use caution in turning power off/on while the power fail message is being typed - it may cause stack overflow.

NOTE

When power returns the program will automatically start itself over at the beginning.

12.12   XINT, RINT

XINT   -   This is the transmitter interrupt service routine for 256(10) byte block transfers.

RINT   -   This is the receiver interrupt service routine for 256(10) byte block transfers.

12.13   DELAY, STALL, DATCHK, TIMERX, TIMETX

These routines are all used by programs 2, 3, 4 and 5. Programs 2 through 5 are the 'Special' user interaction routines which will be defined later in this document. The above routine uses are as follows:

a. DELAY - This routine is used by all the utility programs to wait a no. of milliseconds between character transfers as specified by the user.

b. STALL - This routine is used by program #4 and will allow a random no. of milliseconds to transpire before a transmission of a character. This routine is activated based on user response.

c. DATCHK - This routine is used by program #4 and will check for correct expected and received data after character transmission as well as any error bit conditions.

d. TIMERX + TIMETX -
These two(2) routines are used by program #4 to verify the 'DONE' bit after both transmitter and receiver operations.

12.14   SUERR1, SUER2

These two(2) routines are used throughout the program to set up the error information for 'Error Reporting' before the 'Error Report' call is made. 'Error Report' calls appear throughout the program in the form "ERROR + XX" where 'XX' indicates the particular error table (ERRTB:) entry used by the Error Service Routine.

12.15   PRIME

This routine is used to set up the data buffers an the device under test for each 256(10) byte block transfer.

12.16   CLDLBF

This routine is used in conjunction with routine 'PRIME' to clear input and output buffers before data transfers.

12.17   LDOUT1, LDOUT2, LDOUT3, LDOUT4

The routines are all used for set up and loading purposes as follows:

a. LDOUT1 - is called to set up the 'null-del-null' pattern
b. LDOUT2 - is called to load an ascending binary count pattern
c. LDOUT3 - is called to load a descending binary count pattern
d. LDOUT4 - is called to load a complementing worse case pattern

12.18   CHKDAT

This routine is used to check for data compare errors in 256(10) byte block transfers.

12.19   BUSERR, RSVERR

These two(2) routines are used to service 'Unexpected' bus error and reserved instruction traps, respectively.

12.20   TRPCOM

This routine is used to set up and report the information concerning 'Unexpected' bus error and reserved instruction traps. This routine is used in conjunction with routines 'BUSERR' and 'RSVERR' described above.

13.0   MISCELLANEOUS

a. The stack pointer is initially set to 1100.

b. The parity bit is not covered.

14.0   USER SELECTION PROGRAMS

14.1   Program #2 Description

This utility program will allow the following:

a. Selection of transmitter data buffer

b. Selection of a character for continuous transfer

c. Selection of an expiration time in milliseconds between each transmitter data buffer character transfer

d. A tight scope loop lock on a specific character

The program relies on user response (via TTY) to specific questions as described below:

a. What is the transmitter data buffer address?

The user should respond by typing an address in the range 175616 to 176176 and follow it with a 'carriage return' at which time the program will validate what was typed to see if -

1. The value typed is within the correct range

2. The value typed is an 'even' address, so as not to cause a 'Bus timeout' when referenced, and

3. Then checks to see if the device associated with the value typed is indeed present

NOTE

If either of the three(3) above conditions are not met the program will type a question mark (?), reiterate the initial question, and wait for a 'new' user response.

b. What is the character to be transmitted (octal ASCII e.g., A=101)?

The user should respond by typing an octal ASCII value, for the character desired and follow it with a 'carriage return'.

c. What is the desired msec. delay (octal e.g., 10=8(10))?

The user should respond by typing an octal value for the desired no. of msec. delay and follow it with a 'carriage return'.

E.G. - If user desired 16 msec. delay between each character transfer he should type '20'.

d. At this point the program will loop continuously sending the character specified, with the desired msec. delay between each character transmission.

14.2   Program #3 Description

This utility program will allow the following:

a. Selection of TRANSMITTER data buffer

b. Selection of a character for continuous transfer
   IN MAINTENANCE MODE.

c. Selection of an expiration time in milliseconds between
   each TRANSMITTER data buffer character transfer

d. A tight scope loop lock on a specific character

The program relies on user response (via TTY) to specific
questions as described below:

a. What is the TRANSMITTER data buffer address?

   The user should respond by typing an address in the range
   175616 to 176176 and follow it with a 'carriage return'
   at which time the program will validate what was typed to
   see if -

   1. The value typed is within the correct range

   2. The value typed is an 'even' address, so as not to
      cause a 'Bus timeout' when referenced, and

   3. Then checks to see if the device associated with the
      value typed is indeed present

                        NOTE

           If either of the three(3) above
           conditions are not met the program will
           type a question mark (?), reiterate the
           initial question, and wait for a 'new'
           user response.

b. What is the character to be transmitted (octal ASCII
   e.g., A=101)?

   The user should respond by typing an octal ASCII value,
   for the character desired and follow it with a 'carriage
   return'.

c. What is the desired msec. delay (octal e.g., 10=8(10))?

   The user should respond by typing an octal value for the
   desired no. of msec. delay and follow it with a
   'carriage return'.

      E.G. - If user desired 16 msec. delay between each
             character transfer he should type '20'.

d. At this point the program will loop continuously sending
   the character specified, with the desired msec. delay

between each character transmission.

14.3   Program #4 Description

This utility program will allow the following:

a. Selection of a transmitter data buffer

b. Selection of a single character to be sent, received and
checked with maintenance bit set.

The program relies on user response (via TTY) to specific
questions as described below:

a. What is the transmitter data buffer address?

The user should respond by typing an address in the range
175616 to 176176 and follow it with a 'carriage return'
at which time the program will validate what was typed to
see if -

1. The value typed is within the correct range

2. The value typed is an 'even' address, so as not to
cause a 'Bus timeout' when referenced, and

3. Then checks to see if the device associated with the
value typed is indeed present.

NOTE

If either   of   the   three(3)   above
conditions are not met the program will
type a question mark (?), reiterate the
initial question, and wait for a 'new'
user response.

b. Is a random wait time (msec.) desired 1/0=yes/no?

The user should respond as asked and follow it with a
'carriage return'.

c. What is the character to be transmitted (octal ASCII e.g.
A=101)?

The user should respond by typing an octal ASCII value,
for the character desired and follow it with a 'carriage
return'.

d. At this point the program will loop continuously sending
the character specified, with a random msec. delay
between each character transmission.   Between   each
transmission, 'RCVR' & 'XMITTER' done bits will be

verified, as well as checks for correct data and any error bit conditions.

NOTE

If user response to Item b. (directly above) was a '0' or a plain 'carriage return' then there is no delay between character transmissions.

14.4   Program #5 Description

This utility program will allow user parameters for running a binary count in maintenance mode.

The program relies on user response (via TTY) to specific questions as described below:

a. What is the transmitter data buffer address?

The user should respond by typing an address in the range 175616 to 176176 and follow it with a 'carriage return' at which time the program will validate what was typed to see if -

1. The value typed is within the correct range

2. The value typed is an 'even' address, so as not to cause a 'Bus timeout' when referenced, and

3. Then checks to see if the device associated with the value typed is indeed present.

NOTE

If either of the three(3) above conditions are not met the program will type a question mark (?), reiterate the initial question, and wait for a 'new' user response.

b. Is a random wait time (msec.) desired 1/0=yes/no?

The user should respond as asked and follow it with a 'carriage return'.

c. At this point the program will loop continuously sending binary characters, with a random msec. delay between each character transmission. Between each transmission, 'RCVR' & 'XMITTER' done bits will be verified, as well as checks for correct data and any error bit conditions.

NOTE

If user response to Item B. (directly above) was a '0' or a plain 'carriage return' then there is no delay between character transmissions.

15.0   PROGRAM FUNCTIONAL FLOW CHARTS

16.0   PROGRAM LISTING

```
     1                                   .NLIST  CND,MD,MC
     2                                   .LIST   ME,SEQ,BIN
     3            167400                  $SWR=167400
     4                                   .ENABLE ABS
     5
     6                                   .TITLE   MAINDEC-11-DZDLC-B
     7                                   ;*COPYRIGHT (C) 1977
     8                                   ;*DIGITAL EQUIPMENT CORP.
     9                                   ;**MAYNARD, MASS. 01754
    10                                   ;*
    11                                   ;*PROGRAM BY E. CROWLEY/B. BURGESS
    12                                   ;*
    13                                   ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
    14                                   ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
    15                                   ;*
    16            000001                  $TN=1
    17
    18
    19
    20                                   .SBTTL  OPERATIONAL SWITCH SETTINGS
    21                                   ;*
    22                                   ;*      SWITCH                   USE
    23                                   ;*      ------          --------------------
    24                                   ;*        15           HALT ON ERROR
    25                                   ;*        14           LOOP ON TEST
    26                                   ;*        13           INHIBIT ERROR TYPEOUTS
    27                                   ;*        12           /C OR /D MODEL
    28                                   ;*        11           INHIBIT ITERATIONS
    29                                   ;*        10           BELL ON ERROR
    30                                   ;*         9           LOOP ON ERROR
    31                                   ;*         8           LOOP ON TEST IN SWR<7:0>
    32
    33                                   ;*         0           CREATION OF DEVICE/S TABLE
    34                                   ;*                     OR CHANGE CHARACTER LENGTH
    35
    36                                   .SBTTL  TRAP CATCHER
    37
    38            000000                  .=0
    39                                   ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
    40                                   ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
    41                                   ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
    42            000174                  .=174
    43 000174     000000          DISPREG: .WORD  0          ;;SOFTWARE DISPLAY REGISTER
    44 000176     000000          SWREG:   .WORD  0          ;;SOFTWARE SWITCH REGISTER
    45                                   .SBTTL  STARTING ADDRESS(ES)
    46 000200     000137  001446          JMP    @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
    47 000204     000137  006344          JMP    @#PRG2     ;JUMP TO USER PROGRAM NO. 2
    48 000210     000137  006604          JMP    @#PRG3     ;JUMP TO USER PROGRAM NO. 3
    49 000214     000137  007054          JMP    @#PRG4     ;JUMP TO USER PROGRAM NO. 4
    50 000220     000137  007446          JMP    @#PRG5     ;JUMP TO USER PROGRAM NO. 5
    51
    52            000052                  .=52               ;INFORMATION LOCATION FOR ACT11
    53 000052     000000                  .WORD  0           ;NO POWER FAIL REQUIRED <BIT15=0>
    54                                                       ;IS NOT MEMORY SIZE DEPENDENT <BIT14=0>
    55                                                       ;IS SUITABLE FOR AUTOMATIC OPERATION <BIT13=0>
    56
```

```
    57                                   .SBTTL  BASIC DEFINITIONS
    58
    59                                   ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
    60            001100                  STACK=  1100
    61                                   .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
    62                                   .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL
    63
    64                                   ;*MISCELLANEOUS DEFINITIONS
    65            000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
    66            000012                  LF=     12              ;;CODE FOR LINE FEED
    67            000015                  CR=     15              ;;CODE FOR CARRIAGE RETURN
    68            000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
    69            177776                  PS=     177776          ;;PROCESSOR STATUS WORD
    70                                   .EQUIV  PS,PSW
    71            177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
    72            177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
    73            177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
    74            177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
    75
    76                                   ;*GENERAL PURPOSE REGISTER DEFINITIONS
    77            000000                  R0=     %0              ;;GENERAL REGISTER
    78            000001                  R1=     %1              ;;GENERAL REGISTER
    79            000002                  R2=     %2              ;;GENERAL REGISTER
    80            000003                  R3=     %3              ;;GENERAL REGISTER
    81            000004                  R4=     %4              ;;GENERAL REGISTER
    82            000005                  R5=     %5              ;;GENERAL REGISTER
    83            000006                  R6=     %6              ;;GENERAL REGISTER
    84            000007                  R7=     %7              ;;GENERAL REGISTER
    85            000006                  SP=     %6              ;;STACK POINTER
    86            000007                  PC=     %7              ;;PROGRAM COUNTER
    87
    88                                   ;*PRIORITY LEVEL DEFINITIONS
    89            000000                  PR0=    0               ;;PRIORITY LEVEL 0
    90            000040                  PR1=    40              ;;PRIORITY LEVEL 1
    91            000100                  PR2=    100             ;;PRIORITY LEVEL 2
    92            000140                  PR3=    140             ;;PRIORITY LEVEL 3
    93            000200                  PR4=    200             ;;PRIORITY LEVEL 4
    94            000240                  PR5=    240             ;;PRIORITY LEVEL 5
    95            000300                  PR6=    300             ;;PRIORITY LEVEL 6
    96            000340                  PR7=    340             ;;PRIORITY LEVEL 7
    97
    98                                   ;*"SWITCH REGISTER" SWITCH DEFINITIONS
    99            100000                  SW15=   100000
   100            040000                  SW14=   40000
   101            020000                  SW13=   20000
   102            010000                  SW12=   10000
   103            004000                  SW11=   4000
   104            002000                  SW10=   2000
   105            001000                  SW09=   1000
   106            000400                  SW08=   400
   107            000200                  SW07=   200
   108            000100                  SW06=   100
   109            000040                  SW05=   40
   110            000020                  SW04=   20
   111            000010                  SW03=   10
   112            000004                  SW02=   4
```

```
113       000002              SW01=    2
114       000001              SW00=    1
115                           .EQUIV   SW09,SW9
116                           .EQUIV   SW08,SW8
117                           .EQUIV   SW07,SW7
118                           .EQUIV   SW06,SW6
119                           .EQUIV   SW05,SW5
120                           .EQUIV   SW04,SW4
121                           .EQUIV   SW03,SW3
122                           .EQUIV   SW02,SW2
123                           .EQUIV   SW01,SW1
124                           .EQUIV   SW00,SW0
125
126                           ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
127       100000              BIT15=   100000
128       040000              BIT14=   40000
129       020000              BIT13=   20000
130       010000              BIT12=   10000
131       004000              BIT11=   4000
132       002000              BIT10=   2000
133       001000              BIT09=   1000
134       000400              BIT08=   400
135       000200              BIT07=   200
136       000100              BIT06=   100
137       000040              BIT05=   40
138       000020              BIT04=   20
139       000010              BIT03=   10
140       000004              BIT02=   4
141       000002              BIT01=   2
142       000001              BIT00=   1
143                           .EQUIV   BIT09,BIT9
144                           .EQUIV   BIT08,BIT8
145                           .EQUIV   BIT07,BIT7
146                           .EQUIV   BIT06,BIT6
147                           .EQUIV   BIT05,BIT5
148                           .EQUIV   BIT04,BIT4
149                           .EQUIV   BIT03,BIT3
150                           .EQUIV   BIT02,BIT2
151                           .EQUIV   BIT01,BIT1
152                           .EQUIV   BIT00,BIT0
153
154                           ;*BASIC "CPU" TRAP VECTOR ADDRESSES
155       000004              ERRVEC=  4           ;;TIME OUT AND OTHER ERRORS
156       000010              RESVEC=  10          ;;RESERVED AND ILLEGAL INSTRUCTIONS
157       000014              TBITVEC=14           ;;"T" BIT
158       000014              TRTVEC=  14          ;;TRACE TRAP
159       000014              BPTVEC=  14          ;;BREAKPOINT TRAP (BPT)
160       000020              IOTVEC=  20          ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
161       000024              PWRVEC=  24          ;;POWER FAIL
162       000030              EMTVEC=  30          ;;EMULATOR TRAP (EMT) **ERROR**
163       000034              TRAPVEC=34           ;;"TRAP" TRAP
164       000060              TKVEC=   60          ;;TTY KEYBOARD VECTOR
165       000064              TPVEC=   64          ;;TTY PRINTER VECTOR
166       000240              PIRQVEC=240          ;;PROGRAM INTERRUPT REQUEST VECTOR
167
```

```
168                           .SBTTL  COMMON TAGS
169
170                           ;;*******************************************************
171                           ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
172                           ;*USED IN THE PROGRAM.
173
174       001100              .=1100
175 001100                    $CMTAG:                    ;;START OF COMMON TAGS
176 001100   000000           $PASS:   .WORD   0         ;;CONTAINS PASS COUNT
177 001102   000              $TSTNM:  .BYTE   0         ;;CONTAINS THE TEST NUMBER
178 001103   000              $ERFLG:  .BYTE   0         ;;CONTAINS ERROR FLAG
179 001104   000000           $ICNT:   .WORD   0         ;;CONTAINS SUBTEST ITERATION COUNT
180 001106   000000           $LPADR:  .WORD   0         ;;CONTAINS SCOPE LOOP ADDRESS
181 001110   000000           $LPERR:  .WORD   0         ;;CONTAINS SCOPE RETURN FOR ERRORS
182 001112   000000           $ERTTL:  .WORD   0         ;;CONTAINS TOTAL ERRORS DETECTED
183 001114   000              $ITEMB:  .BYTE   0         ;;CONTAINS ITEM CONTROL BYTE
184 001115   001              $ERMAX:  .BYTE   1         ;;CONTAINS MAX. ERRORS PER TEST
185 001116   000000           $ERRPC:  .WORD   0         ;;CONTAINS PC OF LAST ERROR INSTRUCTION
186 001120   000000           $GDADR:  .WORD   0         ;;CONTAINS ADDRESS OF "GOOD" DATA
187 001122   000000           $BDADR:  .WORD   0         ;;CONTAINS ADDRESS OF "BAD" DATA
188 001124   000000           $GDDAT:  .WORD   0         ;;CONTAINS "GOOD" DATA
189 001126   000000           $BDDAT:  .WORD   0         ;;CONTAINS "BAD" DATA
190 001130   000000                    .WORD   0         ;;RESERVED--NOT TO BE USED
191 001132   000000                    .WORD   0
192 001134   000              $AUTOB:  .BYTE   0         ;;AUTOMATIC MODE INDICATOR
193 001135   000              $INTAG:  .BYTE   0         ;;INTERRUPT MODE INDICATOR
194 001136   000000                    .WORD   0
195 001140   177570           SWR:     .WORD   DSWR      ;;ADDRESS OF SWITCH REGISTER
196 001142   177570           DISPLAY: .WORD   DDISP     ;;ADDRESS OF DISPLAY REGISTER
197 001144   177560           $TKS:    177560            ;;TTY KBD STATUS
198 001146   177562           $TKB:    177562            ;;TTY KBD BUFFER
199 001150   177564           $TPS:    177564            ;;TTY PRINTER STATUS REG. ADDRESS
200 001152   177566           $TPB:    177566            ;;TTY PRINTER BUFFER REG. ADDRESS
201 001154   000              $NULL:   .BYTE   0         ;;CONTAINS NULL CHARACTER FOR FILLS
202 001155   002              $FILLS:  .BYTE   2         ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
203 001156   012              $FILLC:  .BYTE   12        ;;INSERT FILL CHARS. AFTER A "LINE FEED"
204 001157   000              $TPFLG:  .BYTE   0         ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
205 001160   000000           $REGAD:  .WORD   0         ;;CONTAINS THE ADDRESS FROM
206                                                       ;;WHICH ($REG0) WAS OBTAINED
207 001162   000000           $REG0:   .WORD   0         ;;CONTAINS (($REGAD)+0)
208 001164   000000           $REG1:   .WORD   0         ;;CONTAINS (($REGAD)+2)
209 001166   000000           $REG2:   .WORD   0         ;;CONTAINS (($REGAD)+4)
210 001170   000000           $REG3:   .WORD   0         ;;CONTAINS (($REGAD)+6)
211 001172   000000           $REG4:   .WORD   0         ;;CONTAINS (($REGAD)+10)
212 001174   000000           $REG5:   .WORD   0         ;;CONTAINS (($REGAD)+12)
213 001176   000000           $REG6:   .WORD   0         ;;CONTAINS (($REGAD)+14)
214 001200   000000           $REG7:   .WORD   0         ;;CONTAINS (($REGAD)+16)
215 001202   000000           $TMP0:   .WORD   0         ;;USER DEFINED
216 001204   000000           $TMP1:   .WORD   0         ;;USER DEFINED
217 001206   000000           $TMP2:   .WORD   0         ;;USER DEFINED
218 001210   000000           $TMP3:   .WORD   0         ;;USER DEFINED
219 001212   000000           $TMP4:   .WORD   0         ;;USER DEFINED
220 001214   000000           $TMP5:   .WORD   0         ;;USER DEFINED
221 001216   000000           $TMP6:   .WORD   0         ;;USER DEFINED
222 001220   000000           $TMP7:   .WORD   0         ;;USER DEFINED
223 001222   000000           $TMP10:  .WORD   0         ;;USER DEFINED
```

```
224  001224  000000            $TMP11: .WORD   0              ;;USER DEFINED
225  001226  000000            $TMP12: .WORD   0              ;;USER DEFINED
226  001230  000000            $TMP13: .WORD   0              ;;USER DEFINED
227  001232  000000            $TMP14: .WORD   0              ;;USER DEFINED
228  001234  000000            $TMP15: .WORD   0              ;;USER DEFINED
229  001236  000000            $TMP16: .WORD   0              ;;USER DEFINED
230  001240  000000            $TMP17: .WORD   0              ;;USER DEFINED
231  001242  000000            $TIMES: 0                      ;;MAX. NUMBER OF ITERATIONS
232  001244  000000            $ESCAPE:0                      ;;ESCAPE ON ERROR ADDRESS
233  001246  177607  000377    $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
234  001252     077            $QUES:  .ASCII  /?/            ;;QUESTION MARK
235  001253     015            $CRLF:  .ASCII  <15>           ;;CARRIAGE RETURN
236  001254  000012            $LF:    .ASCIZ  <12>           ;;LINE FEED
237                            ;;************************************************************
238
239                            ;THE FOLLOWING TAG(S) ARE USER SUPPLIED BY CALLING THE MACRO
240                            ;"MORETAGS" AS ONE OF THE ARGUMENTS TO THE SYSMAC ROUTINE .SCMTAG
241
242  001256  000000            TABFLG: .WORD   0              ;AN INDICATOR TO SHOW THAT THE
243                                                           ;INFORMATION FOR MULTIPLE DEVICE
244                                                           ;TESTING HAS ALREADY TRANSPIRED
245                                                           ;& "MAINDEC" NAME HAS BEEN PRINTED
246  001260  000000            DLBASE: .WORD   0              ;STORAGE & WORKING LOCATION FOR A DEVICE
247                                                           ;RECEIVER STATUS REGISTER ADDRESS
248  001262  000000            KEEPAD: .WORD   0              ;STORAGE LOCATION FOR THE 1ST
249                                                           ;DEVICE RCSR FROM WHICH
250                                                           ;"BASEADD" IS RESTORED AT THE
251                                                           ;END OF A COMPLETE PROGRAM PASS.
252  001264  000000            BASEADD:.WORD   0              ;STORAGE LOCATION WHICH HOLDS
253                                                           ;THE RCSR ADDRESS OF THE "NEXT"
254                                                           ;DEVICE DURING MULTIPLE TESTING
255  001266  000000            KEEPIV: .WORD   0              ;STORAGE LOCATION FOR THE 1ST
256                                                           ;DEVICE RECEIVER VECTOR FROM
257                                                           ;WHICH "BASEIV" IS RESTORED AT THE
258                                                           ;END OF A COMPLETE PROGRAM PASS
259  001270  000000            BASEIV: .WORD   0              ;STORAGE LOCATION WHICH HOLDS
260                                                           ;THE VECTOR ADDRESS OF THE "NEXT"
261                                                           ;DEVICE DURING MULTIPLE TESTING
262  001272  000000            MULTD:  .WORD   0              ;FLAG TO INDICATE TO "END OF PASS"
263                                                           ;ROUTINE THAT MULTIPLE DEVICE
264                                                           ;TESTING IS BEING CONDUCTED
265                                                           ;0=NO, 1=YES
266  001274  000000            ACTREG: .WORD   0              ;THIS IS THE DEVICE ACTIVE REGISTER
267                                                           ;A BIT IS SET (STARTING AT
268                                                           ;BIT0)FOR EACH CONTIGUOUS DEVICE
269                                                           ;(A MAX. OF 16) THAT IS TO UNDERGO
270                                                           ;TESTING.  THIS LOCATION IS
271                                                           ;AUTOMATICALLY FILLED BASED ON
272                                                           ;USER RESPONSE TO PROGRAM QUESTIONS
273  001276  000000            ROTADD: .WORD   0              ;A ROTATING POINTER TO SIGNAL
274                                                           ;THE LAST DEVICE TESTED (IF
275                                                           ;MULTIPLE DEVICE TESTING WAS BEING
276                                                           ;DONE) IF LESS THAN A FULL COMPLE-
277                                                           ;MENT OF DEVICES (16)WAS SELECTED
278  001300  000000            LASTADD:.WORD   0              ;STORAGE LOCATION FOR THE
279                                                           ;RCSR ADDRESS OF THE LAST DEVICE
```

```
280                                                           ;TESTED (IF MULTIPLE DEVICE
281                                                           ;TESTING WAS SELECTED BY USER)
282  001302  000000            DLPRI:  .WORD   0              ;STORAGE LOCATION FOR THE DEVICE
283                                                           ;INTERRUPT PRIORITY LEVEL
284  001304  000000            LESS1:  .WORD   0              ;THE PRIORITY LEVEL THE CPU
285                                                           ;MUST BE AT TO ALLOW DEVICE INTERRUPTS.
286                                                           ;THIS WILL BE 1 LEVEL LESS THAN
287                                                           ;THE DEVICE LEVEL (BASED ON &
288                                                           ;CALCULATED FROM USER RESPONSE TO
289                                                           ;DEVICE PRIORITY LEVEL QUESTION)
290  001306  177740            STLMSK: 177740                 ;THIS MASK IS USED BY THE "STALL"
291                                                           ;ROUTINE WHICH WAITS A RANDOM NO.
292                                                           ;OF MILLISECONDS.  ITS' USE PREVENTS
293                                                           ;A STALL > 37 MSEC.  THIS LOCATION
294                                                           ;HOWEVER, CAN BE PATCHED BY THE
295                                                           ;USER TO ALLOW LARGER "STALLS".
296
297                            ;END OF USER SUPPLIED TAG(S)
```

```
298                                      .SBTTL  ERROR POINTER TABLE
299
300                                      ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR,
301                                      ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
302                                      ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
303                                      ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC),
304                                      ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
305
306                                      ;*      EM              ;;POINTS TO THE ERROR MESSAGE
307                                      ;*      DH              ;;POINTS TO THE DATA HEADER
308                                      ;*      DT              ;;POINTS TO THE DATA
309                                      ;*      DF              ;;POINTS TO THE DATA FORMAT
310
311
312  001310                             $ERRTB:
313
314                                      ;ERROR TABLE ITEM FOR ERROR MESSAGE 1
315
316  001310  015146                              EM1             ;"DL11 REGISTER REFERENCE CAUSED TIMEOUT"
317  001312  015215                              DH1             ;" (PC) (PS) (SP) TEST DEVADR REGADR  "
318  001314  015274                              DT1             ; (R7) (PSW) (R6) (R0) (R1) (R2)
319  001316  000000                              0               ;PRINT ALL OCTAL
320
321                                      ;ERROR TABLE ITEM FOR ERROR MESSAGE 2
322
323  001320  015312                              EM2             ;" DL11 REGISTER ERROR "
324  001322  015336                              DH2             ;" (PC) (PS) (SP) TEST DEVADR REGADR WAS S/B  "
325  001324  015434                              DT2          .  ;" (R7) (PSW) (R6) (R0) (R1) (R2) (R3) (R4)  "
326  001326  000000                              0               ;PRINT ALL OCTAL
327
328                                      ;ERROR TABLE ITEM FOR ERROR MESSAGE 3
329
330  001330  015456                              EM3             ;" DL11 DATA COMPARE ERROR "
331  001332  015506                              DH3             ;" (PC) (PS) (SP) TEST WASADR SHBADR WAS S/B  "
332  001334  015604                              DT3             ;" (R7) (PSW) (R6) (R0) (R1) (R2) (R3) (R4)  "
333  001336  000000                              0               ;PRINT ALL OCTAL
334
335                                      ;ERROR TABLE ITEM FOR ERROR MESSAGE 4
336
337  001340  015626                              EM4             ;" UNEXPECTED TRAP TO VECTOR AT LOCATION XXX "
338  001342  015700                              DH4             ;" (PC) (PS) (SP) TEST "
339  001344  015736                              DT4             ;" (R7) (PSW) (R6) (R0) "
340  001346  000000                              0               ;PRINT ALL OCTAL
341
342                                      ;ERROR TABLE ITEM FOR ERROR MESSAGE 5
343
344  001350  015750                              EM5             ;" DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN)  "
345  001352  016025                              DH5             ;" (PC) (PS) (SP) TEST DEVADR REGADR (REG)  "
346  001354  016114                              DT5             ;" (R7) (PSW) (R6) (R0) (R1) (R2) (R3)  "
347  001356  000000                              0
348
349                                      ;ERROR TABLE ITEM FOR ERROR MESSAGE 6
350
351  001360  015146                              EM1             ;"DL11 REGISTER REFERENCE CAUSED TIMEOUT"
352  001362  016134                              DH6             ;" (PC)    (PS)    (SP)    REGADR"
353  001364  016174                              DT6             ;$ERRPC,STMP0,$REG6,$REG2
```

```
354  001366  000000                              0               ;PRINT ALL OCTAL
355
356                                      ;ERROR TABLE ITEM FOR ERROR MESSAGE 7
357
358  001370  015750                              EM5             ;" DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN) "
359  001372  016206                              DH7             ;" (PC) DEVADR REGADR (REG)"
360  001374  016246                              DT7             ;$ERRPC,$REG1,$REG2,$REG3
361  001376  000000                              0               ;PRINT ALL OCTAL
362
363                                      ;ERROR TABLE ITEM FOR ERROR MESSAGE 10
364
365  001400  015456                              EM3             ;" DL11 DATA COMPARE ERROR "
366  001402  016260                              DH10            ;" (PC) DEVADR REGADR (REG)  S/B"
367  001404  016326                              DT10            ;$ERRPC,$REG1,$REG2,$REG3,$REG4
368  001406  000000                              0               ;PRINT ALL OCTAL
369
370                                      ;;******************************************************************
371                                      ;DL11 DEFINITIONS
372                                      ;;******************************************************************
373
374  001410  175610                              DLRCSR: 175610          ;CONTAINS ADDRESS OF RCVR CSR
375  001412  175612                              DLRDBR: 175612          ;CONTAINS ADDRESS OF RCVR DBR
376  001414  175614                              DLXCSR: 175614      .   ;CONTAINS ADDRESS OF XMIT CSR
377  001416  175616                              DLXDBR: 175616          ;CONTAINS ADDRESS OF XMIT DBR
378  001420  000300                              DLVECT: 300             ;CONTAINS VECTOR ADDRESS OF CURRENT DL11
379  001422  000000                              XFLG0:  0               ;FLAG FOR HARD XMIT ERRORS
380  001424  000000                              RFLG0:  0               ;FLAG FOR HARD RCVR ERRORS
381  001426  000000                              RFLG1:  0               ;FLAG FOR SOFT RCVR ERRORS
382  001430  000000                              RTRY:   0               ;COUNTS NO. OF RETRIES ON SOFT ERRORS
383  001432  000000                              OPTR:   0               ;CONTAINS POINTER TO OUTPUT BUFFER
384  001434  000000                              IPTR:   0               ;CONTAINS POINTER TO INPUT BUFFER
385  001436  000000                              LDOUT:  0               ;CONTAINS POINTER TO LOAD BUFFER ROUTINE
386  001440  000000                              TIMR1:  0               ;TIMERS FOR 256. BYTE BLOCK TRANSFERS
387  001442  000000                              TIMR2:  0
388  001444  000000                              INTFLG: 0               ;SOFTWARE INTR. FLAG
389
390
391
392
393  001446  000240                      BEGIN:  NOP                     ;PROGRAM WILL START HERE
394                                      .SBTTL  INITIALIZE THE COMMON TAGS
395                                      ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
396  001450  012706  001100                      MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
397  001454  005026                              CLR     (R6)+           ;;CLEAR MEMORY LOCATION
398  001456  022706  001140                      CMP     #SWR,R6 ;;DONE?
399  001462  001374                              BNE     .-6             ;;LOOP BACK IF NO
400  001464  012706  001100                      MOV     #STACK,SP       ;;SETUP THE STACK POINTER
401                                      ;;INITIALIZE A FEW VECTORS
402  001470  012737  010476  000020              MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
403  001476  012737  000340  000022              MOV     #340,@#IOTVEC+2 ;;LEVEL 7
404  001504  012737  010746  000030              MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
405  001512  012737  000340  000032              MOV     #340,@#EMTVEC+2 ;;LEVEL 7
406  001520  012737  013066  000034              MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
407  001526  012737  000340  000036              MOV     #340,@#TRAPVEC+2;;LEVEL 7
408  001534  012737  013146  000024              MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
409  001542  012737  000340  000026              MOV     #340,@#PWRVEC+2 ;;LEVEL 7
```

```
410  001550  005067  177466            CLR    $TIMES              ;;INITIALIZE NUMBER OF ITERATIONS
411  001554  005067  177464            CLR    $ESCAPE             ;;CLEAR THE ESCAPE ON ERROR ADDRESS
412  001560  112767  000001  177327    MOVB   #1,$ERMAX           ;;ALLOW ONE ERROR PER TEST
413  001566  012767  001566  177312    MOV    #.,$LPADR           ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
414  001574  012767  001574  177306    MOV    #.,$LPERR           ;;SETUP THE ERROR LOOP ADDRESS
415                                 ;;SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
416                                 ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
417  001602  013746  000004            MOV    @#ERRVEC,-(SP)      ;;SAVE ERROR VECTOR
418  001606  012737  001642  000004    MOV    #64$,@#ERRVEC       ;;SET UP ERROR VECTOR
419  001614  012767  177570  177316    MOV    #DSWR,SWR           ;;SETUP FOR A HARDWARE SWICH REGISTER
420  001622  012767  177570  177312    MOV    #DDISP,DISPLAY      ;;AND A HARDWARE DISPLAY REGISTER
421  001630  022777  177777  177302    CMP    #-1,@SWR            ;;TRY TO REFERENCE HARDWARE SWR
422  001636  001012                     BNE    66$                 ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
423                                                                 ;;AND  THE HARDWARE SWR IS NOT = -1
424  001640  000403                     BR     65$                 ;;BRANCH IF NO TIMEOUT
425  001642  012716  001650    64$:    MOV    #65$,(SP)           ;;SET UP FOR TRAP RETURN
426  001646  000002                     RTI
427  001650  012767  000176  177262 65$:  MOV  #SWREG,SWR          ;;POINT TO SOFTWARE SWR
428  001656  012767  000174  177256    MOV    #DISPREG,DISPLAY
429  001664  012637  000004    66$:    MOV    (SP)+,@#ERRVEC      ;;RESTORE ERROR VECTOR
430
431  001670  005067  177376            CLR    MULTD               ;CLEAR MULTIPLE DEVICE
432                                                                 ;TESTING FLAG
433  001674  005067  177356            CLR    TABFLG              ;CLEAR TABLE CREATION FLAG
434  001700  012767  000010  177326    MOV    #8.,$TMP15          ;SET CHARACTER LENGTH DESIGNATOR
435                                                                 ;FOR 8 BITS --- THIS IS THE DEFAULT
436                                                                 ;LENGTH ASSUMED BY THE PROGRAM
437                                                                 ;UNLESS THE USER CHANGES IT THRU
438                                                                 ;THE QUESTION AND ANSWER CYCLE
439                                                                 ;INITIATED BY SETTING SW<0> TO A 1
440  001706  012767  000200  177366    MOV    #200,DLPRI          ;SET STANDARD PRIORITY LEVEL
441                                                                 ;FOR DEVICE
442  001714  032777  000400  177216    BIT #SW8,@SWR             ;IS THE 'LOOP ON TEST' SWITCH SET?
443  001722  001411                     BEQ    1$                  ;BRANCH IF NOT
444                                 ;
445                                 ;IF THE 'LOOP ON TEST' SWITCH WAS SET WE WILL TAKE THE NEXT BRANCH
446                                 ;INSTRUCTION THUS BYPASSING TABLE CREATION
447                                 ;
448                                 ;IF THE USER DESIRED TO LOOP ON A TEST OF OTHER THAN THE DEFAULT DEVICE
449                                 ;THEN HE SHOULD HAVE PREVIOUSLY FILLED THE FOLLOWING PROGRAM LOCATIONS
450                                 ;WITH THE DESIRED DEVICE REGISTER VALUES:
451                                 ;
452                                 ;        *******************************
453                                 ;        UNDER  ;DL11 DEFINITIONS          ABOVE
454                                 ;        *******************************
455                                 ;
456                                 ;    DLRCSR:     PATCH THE ADDRESS OF THE RCVR CSR
457                                 ;    DLRDBR:     PATCH THE ADDRESS OF THE RCVR DBR
458                                 ;    DLXCSR:     PATCH THE ADDRESS OF THE XMIT CSR
459                                 ;    DLXDBR:     PATCH THE ADDRESS OF THE XMIT DBR
460                                 ;    DLVECT:     PATCH THE VECTOR ADDRESS OF THIS DL11
461                                 ;
462  001724  104401  016651            TYPE,  STMES               ;PRINT OUT 'MAINDEC' NAME
463  001730  104401  020673            TYPE,  FAILSA              ;TYPE FAILSAFE MESSAGE
464  001734  104000                     ERROR  +0                  ;TYPE OUT THE PC VALUE
465  001736  104401  021251            TYPE,  PCMSG               ;FOLLOWED BY =PC
```

```
466  001742  000000                     HALT                       ;WAIT FOR USER TO RESPOND
467  001744  000443                     BR     ONCE                ;GO TO TEST DEVICE PATCHED IN BY USER
468  001746                       1$:
469                                 ;ENSURE THAT IF MULTIPLE DEVICE TESTING WAS BEING DONE
470                                 ;AND THE USER 'HALTED' THE PROGRAM BEFORE ALL DEVICES
471                                 ;WERE COMPLETED AND WENT BACK TO 'LOAD ADDRESS 200'
472                                 ;TO RESTART THE PROGRAM THAT AS A BARE MINIMUM
473                                 ;HE CAN RUN THE DEFAULT DEVICE (1S1 RECEIVER
474                                 ;STATUS REGISTER ADDRESS 175610)
475                                 ;NOTE:  IF THIS IS NOT SUITABLE THE USER WILL
476                                 ;         HAVE TO SET SW0=1 (OR UP) IN ORDER TO
477                                 ;         RECREATE THE TABLE HE DESTROYED FROM
478                                 ;         ABOVE
479  001746  012767  175610  177304    MOV    #175610,DLBASE      ;1ST POSSIBLE RECEIVER CSR
480  001754  004767  006110            JSR    PC,DLADDR           ;FORM DL ADDRESSES FOR
481                                                                 ;1ST POSSIBLE DEVICE
482  001760  012767  000300  177432    MOV    #300,DLVECT         ;1ST POSSIBLE INTERRUPT VECTOR
483  001766  005067  177264            CLR    TABFLG              ;CLEAR TABLE CREATION FLAG
484
485  001772  012706  001100    RESTRT: MOV    #STACK,SP            ;SET UP STACK POINTER
486  001776  012737  015034  000004    MOV    #BUSERR,@#ERRVEC    ;SET UP BUS ERROR VECTOR
487  002004  012737  000340  000006    MOV    #340,@#ERRVEC+2
488  002012  012737  015060  000010    MOV    #RSVERR,@#RESVEC    ;SET UP RSVD INSTR. VECTOR
489  002020  012737  000340  000012    MOV    #340,@#RESVEC+2
490
491                                 ;THIS  NEXT SECTION WILL CHECK TO SEE IF MULTIPLE DEVICE TESTING
492                                 ;WILL TAKE PLACE I.E.-
493                                 ;    A)  HAS FREE RUNNING DEVICE TABLE ALREADY BEEN CREATED, AND/OR
494                                 ;    B)  IF IT HAS, DOES USER WISH TO CHANGE IT, OR DO WE TEST DEFAULT DEVICE?
495  002026  105767  177224            TSTB   TABFLG              ;HAS TABLE CREATION BEEN PERFORMED?
496  002032  001010                     BNE    ONCE                ;BRANCH IF YES TO SKIP 'MAINDEC'
497                                                                 ;TITLE' MESSAGE
498  002034  104401  016651            TYPE   ,STMES              ;OTHERWISE, PRINT OUT 'MAINDEC'
499                                                                 ;NAME
500  002040  105167  177212            COMB   TABFLG              ;IF TABLE CREATION HAS NOT BEEN
501                                                                 ;PERFORMED, THEN SET FLAG, AND DO SO
502  002044  032777  000001  177066    BIT    #SW0,@SWR           ;THE PROGRAM HAS OBVIOUSLY BEEN
503                                                                 ;RESTARTED - DOES USER WISH TO
504                                                                 ;RESELECT VECTOR AND CONTROL REGISTER
505                                                                 ;ADDRESSES I.E. - CREATE A NEW TABLE?
506  002052  001012                     BNE    GO                  ;BRANCH IF YES
507  002054  005077  177334    ONCE:   CLR    @DLXCSR             ;CLEAR OUT BOTH CSR'S
508  002060  005077  177324            CLR    @DLRCSR
509  002064  005777  177322            TST    @DLRDBR             ;FLUSH RCVR "DONE" BIT
510  002070  005777  177316            TST    @DLRDBR
511  002074  000167  000670            JMP    TST1                ;OTHERWISE, GO WITH EXISTING
512                                                                 ;TABLE OR NOT USE ANY TABLE AT
513                                                                 ;ALL WHICHEVER THE CASE MAY BE
514                                                                 ;(DEFAULT CASE IS 1ST POSSIBLE
515                                                                 ;DEVICE)
516                                 ;IF WE COME THIS PATH THE USER HAS DECIDED 1 OF 2 ALTERNATIVES:
517                                 ;    A)  TO RUN MULTIPLE DEVICES
518                                 ;    B)  TO CREATE A NEW TABLE TO RUN FROM, OR
519                                 ;    C)  TO CHANGE THE CHARACTER LENGTH
520  002100  104401  017233    GO:     TYPE,  LENGTH              ;ASK USER FOR THE CHARACTER LENGTH
521                                                                 ;FOR WHICH HIS DEVICE IS SET
```

```
 522  002104  104411                        RDDEC                 ;ACCEPT THE ANSWER TYPED BY USER
 523                             ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
 524  002106  012600                        MOV     (SP)+,R0      ;GET THE ANSWER TYPED
 525  002110  020027  000010                CMP     R0,#8.        ;IS THE NUMBER TOO HIGH?
 526  002114  101114                         BHI     RETRY         ;IF YES - GO TO RETRY SITUATION
 527  002116  020027  000005                CMP     R0,#5.        ;IS THE NUMBER TOO LOW?
 528  002122  103511                         BLO     RETRY         ;IF YES - GO TO RETRY SITUATION
 529  002124  010067  177104                MOV     R0,$TMP15     ;THE VALUE TYPED IS OK
 530                                                               ;STORE FOR FUTURE USE
 531  002130  104401  017315                TYPE,   DEFAULT       ;ASK USER IF HE WISHES TO TEST OTHER
 532                                                               ;THAN THE DEFAULT DEVICE
 533  002134  104410                        RDOCT                 ;ACCEPT THE ANSWER TYPED BY USER
 534  002136  005726                        TST     (SP)+         ;LOOK AT THE ANSWER
 535  002140  001002                        BNE     1$            ;BRANCH IF REPLY WAS YES
 536  002142  000137  002724                JMP     @#FLUSH       ;OTHERWISE, SKIP REST OF INTERROGATION
 537  002146  012700  000300        1$:     MOV     #300,R0       ;START RESTORATION OF TRAPCATCHER
 538  002152  012701  000302                MOV     #302,R1       ;AREA FROM LOCATIONS 300 TO 776
 539  002156  012702  000004                MOV     #4,R2         ;SO THAT WE CREATE THE MULTIPLE
 540  002162  010110                 2$:     MOV     R1,(R0)       ;DEVICES TABLE WITH A CLEAN SLATE
 541  002164  005011                        CLR     (R1)          ;       .
 542  002166  060200                        ADD     R2,R0         ;       .
 543  002170  060201                        ADD     R2,R1         ;       .
 544  002172  022701  001000                CMP     #1000,R1      ;       .
 545  002176  002771                        BLT     2$            ;       .
 546                             ;THE TRAPCATCHER VECTOR AREA FROM 300 - 776 SHOULD NOW BE RESTORED.
 547                             ;PROCEED TO FIND OUT THE 1ST DEVICE RECEIVER CONTROL REGISTER
 548                             ;ADDRESS
 549  002200  104401  017422     FIRSTD: TYPE    ,MFIRSTD      ;ASK USER FOR THE RECEIVER CONTROL
 550                                                               ;REGISTER ADDRESS OF HIS FIRST
 551                                                               ;DEVICE
 552  002204  104410                        RDOCT                 ;ACCEPT THE ANSWER TYPED BY USER
 553                                                               ;AND STORE ON TOP OF STACK
 554                             ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
 555  002206  012600                        MOV     (SP)+,R0      ;GET THE ANSWER TYPED
 556  002210  020027  176170                CMP     R0,#176170    ;IS THE NUMBER TOO HIGH?
 557  002214  101060                         BHI     RETRY0        ;IF YES-GO TO RETRY SITUATION
 558  002216  020027  175610                CMP     R0,#175610    ;IS THE NUMBER TOO LOW?
 559  002222  103455                         BLO     RETRY0        ;IF YES - GO TO RETRY SITUATION
 560  002224  132700  000001                BITB    #BIT0,R0      ;NUMBER IS IN RANGE BUT IS IT
 561                                                               ;ON AN EVEN BOUNDARY?
 562  002230  001052                        BNE     RETRY0        ;IF NO - GO TO RETRY SITUATION
 563                             ;CHECK TO SEE IF USER RESPONSE WAS TRULY A RCVR STATUS REGISTER
 564  002232  032700  000007                BIT     #7,R0         ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
 565                                                               ;USER RESPONSE EQUAL TO A ZERO?
 566  002236  001047                        BNE     RETRY0        ;BRANCH IF NOT
 567  002240  010067  177014                MOV     R0,DLBASE     ;THE 1ST ADDRESS VALUE TYPED IS OK
 568                                                               ;STORE FOR FUTURE USE
 569                             ;NOW WE ARE READY TO FIND OUT THE DEVICE INTERRUPT VECTOR
 570  002244  016767  177010  177010        MOV     DLBASE,KEEPADD ;GET 1ST ADDRESS VALUE
 571  002252  004767  005612                JSR     PC,DLADDR     ;GO FORM DL ADDRESSES FOR
 572                                                               ;1ST DEVICE SELECTED
 573  002256  016767  177000  177000        MOV     KEEPADD,BASEADD ;RESTORE 1ST DEVICE ADDRESS
 574  002264  104401  017510     VECT:   TYPE    ,MVECT        ;ASK USER FOR A VECTOR ADDRESS
 575  002270  104410                        RDOCT                 ;ACCEPT THE ANSWER TYPED BY USER
 576                                                               ;AND STORE ON TOP OF STACK
 577                             ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
```

```
 578  002272  012600                        MOV     (SP)+,R0      ;GET THE ANSWER TYPED
 579  002274  020027  000776                CMP     R0,#776       ;IS THE NUMBER TOO HIGH?
 580  002300  101032                         BHI     RETRY1        ;IF YES - GO TO RETRY SITUATION
 581  002302  020027  000300                CMP     R0,#300       ;IS THE NUMBER TOO LOW?
 582  002306  103427                         BLO     RETRY1        ;IF YES - GO TO RETRY SITUATION
 583  002310  132700  000001                BITB    #BIT0,R0      ;NUMBER IS IN RANGE BUT IS IT
 584                                                               ;ON AN EVEN BOUNDARY?
 585  002314  001024                        BNE     RETRY1        ;IF NO - GO TO RETRY SITUATION
 586                             ;CHECK TO SEE IF THE USER RESPONSE WAS TRULY A RCVR VECTOR ADDRESS
 587  002316  032700  000007                BIT     #7,R0         ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
 588                                                               ;USER RESPONSE EQUAL TO A ZERO?
 589  002322  001021                        BNE     RETRY1        ;BRANCH IF NOT
 590  002324  010067  177070                MOV     R0,DLVECT     ;THE VECTOR VALUE TYPED IS OK
 591                                                               ;STORE FOR FUTURE USE
 592  002330  016767  177064  176730        MOV     DLVECT,KEEPIV ;GET THE FIRST VECTOR VALUE
 593  002340  016767  177056  176724        MOV     DLVECT,BASEIV ;SAVE FIRST VECTOR VALUE
 594  002344  000414                        BR      HOWMANY       ;GO TO SEE IF USER WANTS MORE
 595                                                               ;THAN 1 DEVICE
 596  002346  104401  001252     RETRY:  TYPE,   $QUES         ;TYPE '?' INDICATING USER TYPED
 597                                                               ;SOMETHING WRONG FOR CHARACTER LENGTH
 598  002352  000167  177522                JMP     GO            ;GO BACK TO REISSUE QUESTION
 599  002356  104401  001252     RETRY0: TYPE    ,$QUES        ;TYPE '?' INDICATING USER TYPE
 600                                                               ;SOMETHING WRONG FOR 1ST ADDRESS
 601  002362  000167  177612                JMP     FIRSTD        ;GO BACK TO REISSUE QUESTION
 602  002366  104401  001252     RETRY1: TYPE    ,$QUES        ;TYPE '?' INDICATING USER TYPED
 603                                                               ;SOMETHING WRONG FOR VECTOR
 604  002372  000167  177666                JMP     VECT          ;GO BACK TO REISSUE QUESTION
 605  002376  104401  017566     HOWMANY:TYPE    ,MULDEV       ;ASK USER IF HE WISHES TO RUN
 606                                                               ;MULTIPLE DEVICES
 607  002402  104410                        RDOCT                 ;ACCEPT THE ANSWER TYPED BY USER
 608                                                               ;AND STORE ON TOP OF STACK
 609  002404  012600                        MOV     (SP)+,R0      ;GET THE ANSWER TYPED
 610  002406  005700                        TST     R0            ;WAS THE ANSWER YES?
 611  002410  001003                        BNE     1$            ;BRANCH IF IT WAS
 612  002412  005067  176654                CLR     MULTD         ;OTHERWISE, INITIALIZE FLAG TO
 613                                                               ;INDICATE NON-MULTIPLE DEVICES
 614  002416  000402                        BR      2$            ;SKIP NEXT INSTRUCTION
 615  002420  105167  176646        1$:     COMB    MULTD         ;INITIALIZE FLAG TO INDICATE
 616                                                               ;RUNNING OF MULTIPLE DEVICES
 617  002424                       2$:
 618  002424  105767  176642                TSTB    MULTD         ;ARE THERE MULTIPLE DEVICES ON
 619                                                               ;THE SYSTEM?
 620  002430  100406                        BMI     LASTD     ;IF SO, GO TO ASK NEXT QUESTION
 621  002432  005067  176636                CLR     ACTREG        ;CLEAR DEVICE ACTIVE FLAG TO
 622                                                               ;INDICATE NO RUNNING OF MULTIPLE
 623                                                               ;DEVICES
 624  002436  005067  176634                CLR     ROTADD        ;CLEAR DEVICE ADDRESS POINTER IN
 625                                                               ;USE WHEN RUNNING MULTIPLE DEVICES
 626  002442  000167  000160                JMP     CONQUES       ;SKIP ASKING NEXT QUESTION
 627  002446                       LASTD:
 628                             ;WE WILL NOW BEGIN TO SET UP THE DEVICE ACTIVE REGISTER FOR RUNNING
 629                             ;MULTIPLE DL11 DEVICES
 630  002446  104401  017653                TYPE    ,MLASTD       ;ASK USER FOR THE RECEIVER
 631                                                               ;CONTROL REGISTER ADDRESS OF
 632                                                               ;HIS LAST DEVICE
 633  002452  104410                        RDOCT                 ;ACCEPT THE ANSWER TYPED BY
```

```
634                                                      ;USER AND STORE ON TOP OF STACK
635                              ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
636   002454  012600            1$:    MOV     (SP)+,R0         ;GET THE ANSWER TYPED
637   002456  020027  176170           CMP     R0,#176170       ;IS THE NUMBER TOO HIGH?
638   002462  101132                    BHI     RETRY2           ;IF YES - GO TO RETRY SITUATION
639   002464  020027  175610           CMP     R0,#175610       ;IS THE NUMBER TOO LOW?
640   002470  103527                    BLO     RETRY2           ;IF YES - GO TO RETRY SITUATION
641   002472  132700  000001           BITB    #BIT0,R0         ;NUMBER IS IN RANGE BUT IS IT
642                                                              ;ON AN EVEN BOUNDARY?
643   002476  001124                    BNE     RETRY2           ;IF NOT - GO TO RETRY SITUATION
644                              ;CHECK TO SEE IF USER RESPONSE WAS TRULY A RCVR STATUS REGISTER
645   002500  032700  000007           BIT     #7,R0            ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
646                                                              ;USER RESPONSE EQUAL TO A ZERO?
647   002504  001121                    BNE     RETRY2           ;BRANCH IF NOT
648   002506  010067  176566           MOV     R0,LASTADD       ;THE LAST ADDRESS VALUE TYPED IS OK
649                                                              ;STORE FOR FUTURE USE
650                              ;NOW WE BEGIN TO ACTUALLY INITIALIZE THE DEVICE ACTIVE REGISTER
651                              ;FROM WHICH THE PROGRAM WILL CYCLE UNTIL ALL DEVICES HAVE BEEN TESTED
652   002512  012767  000001  176556   MOV     #1,ROTADD        ;SET UP POINTER FOR 'ACTREG'
653   002520  005067  176550           CLR     ACTREG           ;CLEAR DEVICE ACTIVE REGISTER
654   002524  056767  176546  176542 2$: BIS    ROTADD,ACTREG    ;MAKE 1ST DEVICE ACTIVE
655   002532  000241                    CLC                      ;CLEAR CARRY BIT FOR POINTER
656                                                              ;ROTATION
657   002534  006167  176536           ROL     ROTADD           ;ARE WE PAST 16 LINE RANGE?
658   002540  103422                    BCS     3$               ;BRANCH IF YES
659   002542  062767  000010  176514   ADD     #10,BASEADD      ;STEP UP BASE ADDRESS
660   002550  026767  176524  176506   CMP     LASTADD,BASEADD  ;IS THIS THE LAST DEVICE?
661   002556  101362                    BHI     2$               ;BRANCH IF NOT
662                              ;NOTE:  IF THIS PATH IS TAKEN IT IS ASSUMED THAT AT LEAST 2 DEVICES
663                              ;       EXIST AND THAT ALL ADDRESSING IS CONTIGUOUS
664   002560  056767  176512  176506   BIS     ROTADD,ACTREG    ;INDICATE NEXT DEVICE ACTIVE
665   002566  012767  000001  176502   MOV     #1,ROTADD        ;RESET POINTER FOR 'ACTREG' FOR
666                                                              ;LATER USE IN END OF PASS ROUTINE
667   002574  016767  176462  176462   MOV     KEEPAD,BASEADD   ;RESET 1ST DEVICE RECEIVER
668                                                              ;CONTROLLER REGISTER ADDRESS FOR
669                                                              ;LATER USE IN END OF PASS ROUTINE
670   002602  000167  000020           JMP     CONQUES          ;GO TO CONTINUE QUESTIONING OF USER
671   002606                   3$:
672                              ;IF WE TAKE THIS PATH IT APPEARS THAT THERE ARE NOT AT LEAST
673                              ;TWO DEVICES PRESENT - IN RESPONSE TO USER TYPING 'YES' TO MULTIPLE
674                              ;DEVICES QUESTION
675   002606  016767  176450  176450   MOV     KEEPADD,BASEADD  ;RESET 1ST DEVICE RECEIVER
676                                                              ;CONTROLLER REGISTER ADDRESS
677   002614  104401  017751           TYPE    ,MRANGE          ;INFORM USER TO CHECK AND RETYPE
678                                                              ;THE LAST DEVICE RCSR ADDRESS
679   002620  104410                    RDOCT                    ;ACCEPT THE ANSWER TYPED BY USER
680                                                              ;AND STORE ON TOP OF STACK
681   002622  000167  177626           JMP     1$
682   002626                   CONQUES:
683                              ;IF WE HAVE REACHED THIS PORTION WE KNOW:
684                              ;     A)  THE 'RXCSR' ADDRESS OF THE 1ST DEVICE
685                              ;     B)  THE 'RXCSR' ADDRESS OF THE LAST DEVICE, SAND
686                              ;     C)  THE INTERRUPT VECTOR OF THE 1ST DEVICE
687                              ;NOW LET'S FIND THE PRIORITY LEVEL
688   002626  104401  020035           TYPE    ,PLEVEL          ;ASK USER FOR PRIORITY LEVEL
689   002632  104410                    RDOCT                    ;ACCEPT ANSWER TYPED BY USER AND
```

```
690                                                      ;STORE ON TOP OF STACK
691   002634  012600                    MOV     (SP)+,R0         ;GET THE ANSWER TYPED
692   002636  020027  000007           CMP     R0,#7            ;IS THE NUMBER TOO HIGH?
693   002642  101046                    BHI     RETRY3           ;IF YES - GO TO RETRY SITUATION
694   002644  020027  000004           CMP     R0,#4            ;IS THE NUMBER TOO LOW?
695   002650  103443                    BLO     RETRY3           ;IF YES GO TO RETRY SITUATION
696   002652  010067  176424           MOV     R0,DLPRI         ;THE PRIORITY TYPED IN IS OK
697                                                              ;STORE FOR FUTURE USE
698
699                              ;THIS SECTION WILL CALCULATE THE PRIORITY LEVEL FOR THE
700                              ;PROCESSOR BASED ON THE USER RESPONSE FOR PRIORITY LEVEL OF THE
701                              ;DEVICE
702   002656  006367  176420           ASL     DLPRI            ;FORM BITS <7-5> OF PSW
703   002662  006367  176414           ASL     DLPRI            ;      .
704   002666  006367  176410           ASL     DLPRI            ;      .
705   002672  006367  176404           ASL     DLPRI            ;      .
706   002676  006367  176400           ASL     DLPRI            ;
707   002702  016767  176374  176374   MOV     DLPRI,LESS1      ;START TO FORM LEVEL TO ALLOW
708                                                              ;INTERRUPTS
709   002710  162767  000001  176366   SUB     #1,LESS1         ;DROP DEVICE LEVEL PRIORITY
710                                                              ;BY 1 LEVEL FOR PSW
711   002716  042767  000037  176360   BIC     #37,LESS1        ;MAKE SURE THE T,N,Z,V & C
712                                                              ;BITS FOR THE PROCESSOR ARE CLEAR
713   002724  005077  176464   FLUSH:   CLR     @DLXCSR          ;CLEAR OUT BOTH CSR'S
714   002730  005077  176454           CLR     @DLRCSR
715   002734  005777  176452           TST     @DLRDBR          ;FLUSH RCVR "DONE" BIT
716   002740  005777  176446           TST     @DLRDBR
717   002744  000167  000020           JMP     TST1             ;BEGIN TESTING
718   002750  104401  001252   RETRY2:  TYPE    ,$QUES           ;TYPE '?' INDICATING USER TYPED
719                                                              ;SOMETHING WRONG FOR LAST ADDRESS
720   002754  000167  177466           JMP     LASTD    ;GO BACK TO REISSUE QUESTION
721   002760  104401  001252   RETRY3:  TYPE    ,$QUES           ;TYPE'?' INDICATING USER TYPED
722                                                              ;SOMETHING WRONG FOR PRIORITY
723   002764  000167  177636           JMP     CONQUES          ;GO BACK TO REISSUE QUESTION
724
725
726                              ;;************************************************************
727                              ;*TEST 1      TEST THAT REFERENCE TO RCSR DOES NOT CAUSE TIMEOUT
728                              ;;************************************************************
729   002770  000004            TST1:    SCOPE
730   002772  016746  175006           MOV     ERRVEC,-(SP)     ;SAVE THE TIMEOUT VECTOR
731   002776  012767  003014  175000   MOV     #1$,ERRVEC       ;GO TO 1$ IF TIMEOUT
732   003004  016702  176400           MOV     DLRCSR,R2        ;REGADR = RCSR ADR
733   003010  005712                    TST     (R2)             ;USE REGADR ON BUS
734   003012  000407                    BR      3$               ;;<GO TO NEXT TEST IF NO TIMEOUT>
735   003014  004767  011112   1$:      JSR     PC,SUERT1        ;GO SET UP ERROR INFO
736   003020  012767  003030  176216   MOV     #2$,$ESCAPE      ;RETURN TO 2$ AFTER ERROR PRINT
737   003026  104001            ERROR+1                          ;DL REFERENCE CAUSED BUS TIMEOUT
738   003030  022626            2$:      CMP     (SP)+,(SP)+      ;CLEAN STACK FROM TIMEOUT
739   003032  012667  174746   3$:      MOV     (SP)+,ERRVEC     ;RESTORE TIMEOUT VECTOR
740
741                              ;;************************************************************
742                              ;*TEST 2      TEST THAT REFERENCE TO XCSR DOES NOT CAUSE TIMEOUT
743                              ;;************************************************************
744   003036  000004            TST2:    SCOPE
745   003040  016746  174740           MOV     ERRVEC,-(SP)     ;SAVE THE TIMEOUT VECTOR
```

```
746  003044  012767  003062  174732        MOV   #1$,ERRVEC       ;GO TO 1$ IF TIMEOUT
747  003052  016702  176336               MOV   DLXCSR,R2        ;REGADR = XCSR ADR
748  003056  005712                        TST   (R2)             ;USE REGADR ON BUS
749  003060  000407                        BR    3$               ;;<GO TO NEXT TEST IF NO TIMEOUT>
750  003062  004767  011044        1$:     JSR   PC,SUERT1        ;GO SET UP ERROR INFO
751  003066  012767  003076  176150        MOV   #2$,$ESCAPE      ;RETURN TO 2$ AFTER ERROR PRINT
752  003074  104001                         ERROR+1              ;DL REFERENCE CAUSED BUS TIMEOUT
753  003076  022626                 2$:     CMP   (SP)+,(SP)+      ;CLEAN STACK FROM TIMEOUT
754  003100  012667  174700        3$:     MOV   (SP)+,ERRVEC     ;RESTORE TIMEOUT VECTOR
755
756                                  ;;*******************************************************
757                                  ;*TEST 3       TEST THAT REFERENCE TO RDBR DOES NOT CAUSE TIMEOUT
758                                  ;;*******************************************************
759  003104  000004        TST3:    SCOPE
760  003106  016746  174672        MOV   ERRVEC,-(SP)     ;SAVE THE TIMEOUT VECTOR
761  003112  012767  003130  174664        MOV   #1$,ERRVEC       ;GO TO 1$ IF TIMEOUT
762  003120  016702  176266               MOV   DLRDBR,R2        ;REGADR = RDBR ADR
763  003124  005712                        TST   (R2)             ;USE REGADR ON BUS
764  003126  000407                        BR    3$               ;;<GO TO NEXT TEST IF NO TIMEOUT>
765  003130  004767  010776        1$:     JSR   PC,SUERT1        ;GO SET UP ERROR INFO
766  003134  012767  003144  176102        MOV   #2$,$ESCAPE      ;RETURN TO 2$ AFTER ERROR PRINT
767  003142  104001                         ERROR+1              ;DL REFERENCE CAUSED BUS TIMEOUT
768  003144  022626                 2$:     CMP   (SP)+,(SP)+      ;CLEAN STACK FROM TIMEOUT
769  003146  012667  174632        3$:     MOV   (SP)+,ERRVEC     ;RESTORE TIMEOUT VECTOR
770
771                                  ;;*******************************************************
772                                  ;*TEST 4       TEST THAT REFERENCE TO XDBR DOES NOT CAUSE TIMEOUT
773                                  ;;*******************************************************
774  003152  000004        TST4:    SCOPE
775  003154  016746  174624        MOV   ERRVEC,-(SP)     ;SAVE THE TIMEOUT VECTOR
776  003160  012767  003176  174616        MOV   #1$,ERRVEC       ;GO TO 1$ IF TIMEOUT
777  003166  016702  176224               MOV   DLXDBR,R2        ;REGADR = XDBR ADR
778  003172  005712                        TST   (R2)             ;USE REGADR ON BUS
779  003174  000407                        BR    3$               ;;<GO TO NEXT TEST IF NO TIMEOUT>
780  003176  004767  010730        1$:     JSR   PC,SUERT1        ;GO SET UP ERROR INFO
781  003202  012767  003212  176034        MOV   #2$,$ESCAPE      ;RETURN TO 2$ AFTER ERROR PRINT
782  003210  104001                         ERROR+1              ;DL REFERENCE CAUSED BUS TIMEOUT
783  003212  022626                 2$:     CMP   (SP)+,(SP)+      ;CLEAN STACK FROM TIMEOUT
784  003214  012667  174564        3$:     MOV   (SP)+,ERRVEC     ;RESTORE TIMEOUT VECTOR
785
786                                  ;;*******************************************************
787                                  ;*TEST 5       TEST THAT RCSR IS ALL ZEROES ON ENTRY
788                                  ;;*******************************************************
789  003220  000004        TST5:    SCOPE
790  003222  005004                        CLR   R4               ;RESULT IN RCSR S/B = 0
791  003224  016702  176160               MOV   DLRCSR,R2        ;REGADR = RCSR ADR
792  003230  020412                        CMP   R4,(R2)          ;[RCSR]=000000 ??
793  003232  001403                        BEQ   TST6             ;;<BR IF YES>
794  003234  004767  010612               JSR   PC,SUER2         ;GO SET UP ERROR INFO
795  003240  104002                         ERROR+2              ;RCSR NOT CLEAR ON START UP
796                                  ;;*******************************************************
797                                  ;*TEST 6       TEST THAT "READY" BIT IS ONLY BIT SET IN XCSR
798                                  ;;*******************************************************
799  003242  000004        TST6:    SCOPE
800  003244  012704  000200        MOV   #200,R4          ;RESULT IN XCSR S/B = 000200
801  003250  016702  176140               MOV   DLXCSR,R2        ;REGADR = XCSR ADR
```

```
802  003254  020412                        CMP   R4,(R2)          ;[XCSR]=000200 ??
803  003256  001403                        BEQ   TST7             ;;<BR IF YES>
804  003260  004767  010566               JSR   PC,SUER2         ;GO SETUP ERROR INFO
805  003264  104002                         ERROR+2              ;[XCSR] INCORRECT ON START UP
806                                  ;;*******************************************************
807                                  ;*TEST 7       TEST THAT "MAINT" BIT CAN BE SET AND CLEARED
808                                  ;;*******************************************************
809  003266  000004        TST7:    SCOPE
810  003270  012704  000204        MOV   #204,R4          ;RESULT IN XCSR S/B = 000204
811  003274  016702  176114               MOV   DLXCSR,R2        ;REGADR = XCSR ADR
812  003300  052712  000004               BIS   #BIT2,(R2)       ;SET THE "MAINT" BIT
813  003304  020412                        CMP   R4,(R2)  ;RESULT IN XCSR OK ??
814  003306  001403                        BEQ   1$               ;;<BR IF YES>
815  003310  004767  010536               JSR   PC,SUER2         ;GO SET UP ERROR INFO
816  003314  104002                         ERROR+2              ;MAINT, BIT FAILED TO SET PROPERLY
817  003316  012704  000200        1$:     MOV   #200,R4          ;RESULT IN XCSR S/B = 000200
818  003322  042712  000004               BIC   #BIT2,(R2)       ;NOW CLEAR THE "MAINT" BIT
819  003326  020412                        CMP   R4,(R2)          ;RESULT IN XCSR OK ??
820  003330  001403                        BEQ   TST10            ;;<BR IF YES>
821  003332  004767  010514               JSR   PC,SUER2         ;GO SET UP ERROR INFO
822  003336  104002                         ERROR+2              ;MAINT BIT FAILED TO CLEAR PROPERLY
823                                  ;;*******************************************************
824                                  ;*TEST 10      TEST THAT XMIT I.E. CAN CAUSE AN INTR
825                                  ;;*******************************************************
826  003340  000004        TST10:   SCOPE
827  003342  005067  176076        CLR   INTFLG           ;INIT SOFTWARE INTR FLAG
828  003346  016705  176046               MOV   DLVECT,R5        ;GET VECTOR ADDRESS
829  003352  012765  003440  000004        MOV   #2$,4(R5)        ;GO TO 4$ ON INTR
830  003360  016765  175716  000006        MOV   DLPRI,6(R5)      ;PRIORITY LEVEL 4
831  003366  005005                        CLR   R5               ;INIT INTR, TIMER
832  003370  012704  000200               MOV   #200,R4          ;RESULT IN XCSR S/B = 000200
833  003374  016702  176014               MOV   DLXCSR,R2        ;REGADR = XCSR ADR
834  003400  052712  000100               BIS   #100,(R2)        ;SET INTR. ENABLE BIT 06
835  003404  005767  176034        1$:     TST   INTFLG           ;DID INTR OCCUR YET ??
836  003410  001020                        BNE   3$               ;BR IF IT DID
837  003412  005305                        DEC   R5               ;COUNT THE TIMER
838  003414  001373                        BNE   1$               ;BR IF NO TIMEOUT
839  003416  012704  000300               MOV   #300,R4          ;RESULT IN XCSR S/B = 000300
840  003422  004767  010424               JSR   PC,SUER2         ;GO SETUP ERROR INFO
841  003426  012767  003436  175610        MOV   #4$,$ESCAPE      ;RETURN TO 4$ AFTER ERROR PRINT
842  003434  104002                         ERROR+2              ;INTR. FAILED
843  003436                         4$:
844  003436  000412                        BR    TST11            ;;<GO TO NEXT TEST>
845  003440  005167  176000        2$:     COM   INTFLG           ;SET THE SOFTWARE FLAG
846  003444  042712  000100               BIC   #100,(R2)        ;TURN OFF I.E. BIT
847  003450  000002                        RTI                   ;RETURN CONTROL TO INTR, ROUTINE
848  003452  020412                3$:     CMP   R4,(R2)          ;RESULT IN XCSR OK ??
849  003454  001403                        BEQ   TST11            ;;<BR IF YES>
850  003456  004767  010370               JSR   PC,SUER2         ;GO SET UP ERROR INFO.
851  003462  104002                         ERROR+2              ;XMIT INTR. NOT SERVICED PROPERLY
852                                  ;;*******************************************************
853                                  ;*TEST 11      TEST THAT RCVR I.E. BIT CAN BE SET AND CLEARED
854                                  ;;*******************************************************
855  003464  000004        TST11:   SCOPE
856  003466  012704  000100        MOV   #100,R4          ;RESULT IN RCSR S/B = 000100
857  003472  016702  175712               MOV   DLRCSR,R2        ;REGADR = RCSR ADR
```

```
 858  003476  052712  000100           BIS     #BIT6,(R2)        ;SET I.E. BIT
 859  003502  020412                    CMP     R4,(R2)           ;DID IT SET PROPERLY ??
 860  003504  001403                    BEQ     1$                ;;<BR IF YES>
 861  003506  004767  010340            JSR     PC,SUER2          ;GO SET UP ERROR INFO.
 862  003512  104002                    ERROR+2                   ;RCVR I.E. BIT FAILED TO SET PROPERLY
 863  003514  005004            1$:     CLR     R4                ;RESULT IN RCSR S/B = 000000
 864  003516  042712  000100            BIC     #BIT6,(R2)        ;CLEAR THE I.E. BIT
 865  003522  020412                    CMP     R4,(R2)           ;DID IT CLEAR PROPERLY ??
 866  003524  001403                    BEQ     TST12             ;;<BR IF YES>
 867  003526  004767  010320            JSR     PC,SUER2          ;GO SET UP ERROR INFO
 868  003532  104002                    ERROR+2                   ;RCVR I.E. BIT FAILED TO CLEAR PROPERLY
 869                               ;;***************************************************************
 870                               ;*TEST 12      TEST THAT RCVR "DONE" CAN GENERATE AN INTR.
 871                               ;;***************************************************************
 872  003534  000004            TST12:  SCOPE
 873  003536  016705  175656            MOV     DLVECT,R5         ;GET THE VECTOR ADDRESS
 874  003542  012725  003720            MOV     #3$,(R5)+         ;GO TO 3$ ON RCVR INTR.
 875  003546  016715  175530            MOV     DLPRI,(R5)        ;AT LEVEL 4
 876  003552  005067  175666            CLR     INTFLG            ;INIT THE SOFTWARE FLAG
 877  003560  005005                    CLR     R5                ;INIT INTR. TIMER
 878  003560  105067  175420            CLRB    STMP1             ;INIT WHERE DATA WILL BE STORED
 879  003564  016702  175620            MOV     DLRCSR,R2         ;REGADR = RCSR ADR
 880  003570  005012                    CLR     (P2)              ;INIT THE RCSR TO 000000
 881  003572  052712  000100            BIS     #BIT6,(R2)        ;ENABLE RCVR INTERRUPTS
 882  003576  052762  000004  000004    BIS     #BIT2,4(R2)       ;NOW TURN ON MAINT MODE
 883  003604  112767  000252  175372    MOVB    #252,STMP1        ;GET DATA PATTERN AND
 884  003612  004767  011126            JSR     PC,UPMASK         ;GO MASK OFF BITS AS A FUNCTION OF
 885                                                              ;CHARACTER LENGTH ( 5, 6, 7, OR 8 BITS)
 886  003616  116700  175410            MOVB    STMP14,R0         ;SAVE DATA PATTERN FOR FURTHER USE
 887  003622  116762  175404  000006    MOVB    STMP14,6(R2)      ;LOAD XMIT BUFFER REG.
 888  003630  005767  175610    1$:     TST     INTFLG            ;DID RCVR INTR. YET ??
 889  003634  001044                    BNE     4$                ;BR IF IT DID
 890  003636  005305                    DEC     R5                ;COUNT THE TIMER
 891  003640  001373                    BNE     1$                ;BR IF NO TIMEOUT
 892  003642  013767  177776  175332    MOV     @#PSW,STMP0       ;SAVE ERROR PSW
 893  003650  042762  000004  000004    BIC     #BIT2,4(R2)       ;DISABLE MAINT MODE
 894  003656  042712  000100            BIC     #100,(R2)         ;DISABLE RCVR INTR.
 895  003662  010667  175310            MOV     SP,SREG6          ;SAVE THE ERROR SP
 896  003666  010201                    MOV     R2,R1             ;DEVADR = RCSR ADR
 897  003670  011203                    MOV     (R2),R3           ;GET THE WAS DATA
 898  003672  012704  000200            MOV     #200,R4           ;[RCSR] S/B = 000200
 899  003676  004767  010176            JSR     PC,SUERR1         ;GO SET UP ERROR INFO.
 900  003702  012767  003712  175334    MOV     #2$,SESCAPE       ;RETURN TO 2$ AFTER ERROR ALWAYS
 901  003710  104002                    ERROR+2                   ;RCVR INTERUPT FAILED
 902  003712  005762  000002    2$:     TST     2(R2)             ;REFERENCE RCVR DATA BUFFER
 903                                                              ;TO CLEAR RCSR IN CASE RCVR
 904                                                              ;INTERRUPTS COULD NOT BE ENABLED
 905  003716  000437                    BR      TST13             ;;<GO TO NEXT TEST>
 906  003720  042762  000004  000004 3$: BIC    #BIT2,4(R2)       ;DISABLE THE MAINT MODE
 907  003726  116267  000002  175250    MOVB    2(R2),STMP1       ;GET THE RECEIVED DATA
 908  003734  042712  000100            BIC     #BIT6,(R2)        ;TURN OFF RCVR INTR. ENAB
 909  003740  005167  175500            COM     INTFLG            ;SET THE SOFTWARE FLAG
 910  003744  000002                    RTI                       ;RETURN TO MAINLINE
 911  003746  005004            4$:     CLR     R4                ;[RCSR] S/B=0
 912  003750  005712                    TST     (R2)              ;IS IT ALL ZEROES ??
 913  003752  001403                    BEQ     5$                ;;<BR IF YES>
```

```
 914  003754  004767  010072            JSR     PC,SUER2          ;GO SET UP ERROR INFO
 915  003760  104002                    ERROR+2                   ;RCVR INTR NOT SERVICED PROPERLY
 916  003762  016701  175424    5$:     MOV     DLRDBR,R1         ;SAVE WAS ADDRESS
 917  003766  016702  175424            MOV     DLXDBR,R2         ;SAVE THE S/B ADDRESS
 918  003772  004767  010746            JSR     PC,UPMASK         ;GET THE WAS DATA AND
 919                                                              ;GO MASK OFF BITS AS A FUNCTION OF
 920                                                              ;CHARACTER LENGTH ( 5, 6, 7, OR 8 BITS)
 921  003776  116703  175230            MOVB    STMP14,R3         ;SET UP FOR ERROR CHECKING
 922  004002  110004                    MOVB    R0,R4             ;GET THE S/B DATA
 923  004004  020403                    CMP     R4,R3             ;WAS = S/B ??
 924  004006  001403                    BEQ     TST13             ;;<BR IF YES>
 925  004010  004767  010064            JSR     PC,SUERR1         ;GO SET UP THE ERROR INFO
 926  004014  104003                    ERROR+3                   ;DATA COMPARE ERROR
 927                               ;;***************************************************************
 928                               ;*TEST 13      TEST THAT "REQ TO SEND" ASSERTS "RING"
 929                               ;;***************************************************************
 930  004016  000004            TST13:  SCOPE
 931  004020  032777  010000  175112    BIT     #SW12,@SWR        ;ARE WE TESTING /C OR /D MODEL?
 932  004026  001047                    BNE     TST14             ;;<BRANCH IF YES>
 933  004030  012704  140004            MOV     #140004,R4        ;RESULT IN RCSR S/B = 140004
 934  004034  016702  175350            MOV     DLRCSR,R2         ;REGADR = RCSR ADR
 935  004040  005012                    CLR     (R2)              ;INIT THE RCSR TO 000000
 936  004042  052712  000004            BIS     #BIT2,(R2)        ;SET "REQ TO SEND"
 937  004046  032777  100000  175334    BIT     #BIT15,@DLRCSR    ;DID "RING" SET "DATA SET INT" ?
 938  004054  001003                    BNE     1$                ;;<BR IF YES>
 939  004056  004767  007770            JSR     PC,SUER2          ;GO SET UP ERROR INFO.
 940  004062  104002                    ERROR+2           ;"RING" TRANSITION FAILED TO SET "DATA SET INT"
 941                                                       ;NOTE: "BIT  #BIT15,(R2)" RESETS BIT15
 942  004064  012704  040004    1$:     MOV     #40004,R4         ;RESULT IN RCSR S/B = 40004
 943  004070  020412                    CMP     R4,(R2)           ;BOTH "RING" AND "REQ TO SEND" ASSERTED ?
 944  004072  001403                    BEQ     2$                ;;<BR IF YES>
 945  004074  004767  007752            JSR     PC,SUER2          ;GO SET UP ERROR INFO.
 946  004100  104002                    ERROR+2           ;"RING" OR "REQ TO SEND" FAILED TO SET
 947  004102  005004            2$:     CLR     R4                ;RESULT IN RCSR S/B = 000000
 948  004104  042712  000004            BIC     #BIT2,(R2)        ;CLEAR "REQ TO SEND"
 949  004110  032777  100000  175272    BIT     #BIT15,@DLRCSR    ;DID "DATA SET INT" GET SET ??
 950  004116  001403                    BEQ     3$                ;;<BR IF NOT>
 951  004120  004767  007726            JSR     PC,SUER2          ;GO SET UP ERROR INFO
 952  004124  104002                    ERROR+2                   ;CLEARING "RING" SET "DATA SET INT"
 953  004126  020412            3$:     CMP     R4,(R2)           ;RCSR CONTAIN ALL ZEROES ??
 954  004130  001406                    BEQ     TST14             ;;<BR IF YES>
 955  004132  004767  007714            JSR     PC,SUER2          ;GO SET UP ERROR INFO.
 956  004136  016767  000002  175034    MOV     .+6,SREG7         ;SAVE THE ERROR PC
 957  004144  104002                    ERROR+2           ;CLEARING  "REQ TO SEND" FAILED TO CLEAR "RING"
 958                               ;;***************************************************************
 959                               ;*TEST 14      TEST THAT "SEC XMIT" ASSERTS "SEC REC" AND "DATA SET INT"
 960                               ;;***************************************************************
 961  004146  000004            TST14:  SCOPE
 962  004150  032777  010000  174762    BIT     #SW12,@SWR        ;ARE WE TESTING /C OR /D MODEL?
 963  004156  001046                    BNE     TST15             ;;<BRANCH IF YES>
 964  004160  016702  175224            MOV     DLRCSR,R2         ;REGADR = RCSR ADR
 965  004164  005012                    CLR     (R2)              ;INIT RCSR TO 000000
 966  004166  012704  102010            MOV     #102010,R4        ;CONTENTS OF RCSR S/B = 102010
 967  004172  052712  000010            BIS     #BIT3,(R2)        ;SET "SEC XMIT" BIT
 968  004176  032777  100000  175204    BIT     #BIT15,@DLRCSR    ;DID "DATA SET INT" SET ??
 969  004204  001003                    BNE     1$                ;;<BR IF YES>
```

```
 970  004206  004767  007640           JSR    PC,SUER2      ;GO SET UP ERROR INFO
 971  004212  104002                    ERROR+2             ;"DATA SET INT" FAILED TO SET-NOTE THAT
 972                                                         ;"BIT #BIT15,(R2)" RESETS BIT15
 973  004214  012704  002010     1$:    MOV    #2010,R4      ;RESULT IN RCSR S/B = 2010
 974  004220  020412                    CMP    R4,(R2)       ;ARE "SEC XMIT" AND "SEC REC" BOTH SET ?
 975  004222  001403                    BEQ    2$            ;;<BR IF YES>
 976  004224  004767  007622           JSR    PC,SUER2      ;GO SET UP ERROR INFO
 977  004230  104002                    ERROR+2             ;"SEC XMIT" OR "SEC REC" FAILED TO SET
 978                                                         ;OR "DATA SET INT" FAILED TO BE CLEARED
 979                                                         ;WHEN REFERENCING RCSR
 980  004232  012704  100000     2$:    MOV    #BIT15,R4     ;RESULT IN RCSR S/B = 100000
 981  004236  042712  000010           BIC    #BIT3,(R2)    ;CLEAR "SEC XMIT" BIT
 982  004242  032777  100000  175140   BIT    #BIT15,@DLRCSR ;DID CLEARING IT SET "DATA SET INT"??
 983  004250  001003                    BNE    3$            ;;<BR IF YES>
 984  004252  004767  007574           JSR    PC,SUER2      ;GO SET UP ERROR INFO,
 985  004256  104002                    ERROR+2             ;CLEARING "SEC XMIT" FAILED TO SET "DATA
 986                                                         ;SET INT, (NOTE THAT REFERENCING RCSR
 987                                                         ; CLEARS "DATA SET INT"
 988  004260  005004           3$:    CLR    R4            ;RESULT IN RCSR S/B = 000000
 989  004262  020412                    CMP    R4,(R2)       ;"SEC XMIT" AND "SEC REC" CLEAR ?
 990  004264  001403                    BEQ    TST15         ;;<BR IF YES>
 991  004266  004767  007560           JSR    PC,SUER2      ;GO SETUP ERROR INFO
 992  004272  104002                    ERROR+2             ;"SEC XMIT" OR "SEC REC" FAILED TO CLEAR
 993                                                         ;OR REFERENCING RCSR FAILED TO CLEAR "DATA SET INT"
 994                        ;;*************************************************************
 995                        ;*TEST 15       TEST THAT "DTR" CAN ASSERT "CLR TO SEND" AND "CAR DET"
 996                        ;;*************************************************************
 997  004274  000004           TST15: SCOPE
 998  004276  032777  010000  174634   BIT    #SW12,@SWR    ;ARE WE TESTING /C OR /D MODEL?
 999  004304  001046                    BNE    TST16         ;;<BRANCH IF YES>
1000  004306  016702  175076           MOV    DLRCSR,R2     ;REGADR = RCSR ADR
1001  004312  005012                    CLR    (R2)          ;INIT RCSR TO 000000
1002  004314  012704  130002           MOV    #130002,R4    ;RESULT IN RCSR S/B = 130002
1003  004320  052712  000002           BIS    #BIT1,(R2)    ;SET "DTR" BIT
1004  004324  032777  100000  175056   BIT    #BIT15,@DLRCSR ;DID "DATA SET INT" SET ??
1005  004332  001003                    BNE    1$            ;;<BR IF YES>
1006  004334  004767  007512           JSR    PC,SUER2      ;GO SET UP ERROR INFO,
1007  004340  104002                    ERROR+2             ;"DATA SET INT" FAILED TO SET -
1008                                                         ;NOTE: THE REFERENCE TO RCSR ABOVE WILL
1009                                                         ;WILL UNCONDITIONALLY CLEAR RCSR BIT 15.
1010  004342  012704  030002     1$:    MOV    #30002,R4     ;RESULT IN RCSR S/B = 30002
1011  004346  020412                    CMP    R4,(R2)       ;"DTR","CLR TO SEND", AND "CAR DET" ALLSET
1012  004350  001403                    BEQ    2$            ;;<BR IF ALL SET>
1013  004352  004767  007474           JSR    PC,SUER2      ;GO SET UP ERROR INFO
1014  004356  104002                    ERROR+2             ;"DTR","CLR TO SEND" OR "CAR DET" FAILED
1015                                                         ;TO SET OR "DATA SET INT" FAILED TO CLEAR
1016  004360  012704  100000     2$:    MOV    #BIT15,R4     ;RESULT IN RCSR S/B = 100000
1017  004364  042712  000002           BIC    #BIT1,(R2)    ;NOW CLEAR "DTR"
1018  004370  032777  100000  175012   BIT    #BIT15,@DLRCSR ;DID "DATA SET INT" SET ??
1019  004376  001003                    BNE    3$            ;;<BR IF YES>
1020  004400  004767  007446           JSR    PC,SUER2      ;GO SETUP ERROR INFO
1021  004404  104002                    ERROR+2             ;"DATA SET INT" FAILED TO SET WHEN "DTR"
1022                                                         ;WENT TO A ZERO,
1023  004406  005004           3$:    CLR    R4            ;RESULT IN RCSR S/B = 000000
1024  004410  020412                    CMP    R4,(R2)       ;DID ALL BITS CLEAR??
1025  004412  001403                    BEQ    TST16         ;;<BR IF YES>
```

```
1026  004414  004767  007432           JSR    PC,SUER2      ;GO SET UP ERROR INFO
1027  004420  104002                    ERROR+2             ;"DTR","CLR TO SEND" OR "CAR DET" FAILED
1028                                                         ;TO CLEAR PROPERLY
1029                        ;;*************************************************************
1030                        ;*TEST 16       TEST THAT "DATA SET INT ENAB" CAN SET AND CLEAR
1031                        ;;*************************************************************
1032  004422  000004           TST16: SCOPE
1033  004424  032777  010000  174506   BIT    #SW12,@SWR    ;ARE WE TESTING /C OR /D MODEL?
1034  004432  001023                    BNE    TST17         ;;<BRANCH IF YES>
1035  004434  016702  174750           MOV    DLRCSR,R2     ;REGADR = RCSR ADR
1036  004440  012704  000040           MOV    #40,R4        ;RESULT IN RCSR S/B = 000040
1037  004444  052712  000040           BIS    #BIT5,(R2)    ;SET THE "DATA SET I.E." BIT
1038  004450  020412                    CMP    R4,(R2)       ;DID IT SET OK ??
1039  004452  001403                    BEQ    1$            ;;<BR IF YES>
1040  004454  004767  007372           JSR    PC,SUER2      ;GO SET UP ERROR INFO
1041  004460  104002                    ERROR+2             ;"DAT SET I. E." FAILED TO SET
1042  004462  005004           1$:    CLR    R4            ;MAKE S/B DATA = 000000
1043  004464  042712  000040           BIC    #BIT5,(R2)    ;NOW CLEAR THE "DATA SET I.E." BIT
1044  004470  020412                    CMP    R4,(R2)       ;DID IT CLEAR OK ??
1045  004472  001403                    BEQ    TST17         ;;<BR IF YES>
1046  004474  004767  007352           JSR    PC,SUER2      ;GO SET UP ERROR INFO,
1047  004500  104002                    ERROR+2             ;"DATA SET I.E." FAILED TO CLEAR
1048                        ;;*************************************************************
1049                        ;*TEST 17       TEST THE "DATA SET I.E." CAN CAUSE A RCVR INTR
1050                        ;;*************************************************************
1051  004502  000004           TST17: SCOPE
1052  004504  032777  010000  174426   BIT    #SW12,@SWR    ;ARE WE TESTING A /C OR /D MODEL?
1053  004512  001054                    BNE    TST20         ;;<BRANCH IF YES>
1054  004514  016705  174700           MOV    DLVECT,R5     ;GET THE VECTOR ADDR
1055  004520  012725  004606           MOV    #3$,(R5)+     ;GO TO 3$ ON RCVR INTR.
1056  004524  016715  174552           MOV    DLPR1,(R5)    ;AT LEVEL 4
1057  004530  005005                    CLR    R5            ;INIT INTR, TIMER
1058  004532  005067  174706           CLR    INTFLG        ;INIT SOFTWARE FLAG
1059  004536  005004                    CLR    R4            ;RESULT IN RCSR S/B = 0 AFTER INTR,
1060  004540  016702  174644           MOV    DLRCSR,R2     ;REGADR = RCSR ADR
1061  004544  052712  000040           BIS    #BIT5,(R2)    ;SET THE "DATA SET I.E." BIT
1062  004550  052712  000002           BIS    #BIT1,(R2)    ;NOW SET "DTR" TO GEN INTR,
1063  004554  005767  174664     1$:    TST    INTFLG        ;DID INTR OCCUR YET ??
1064  004560  001016                    BNE    4$            ;BR IF YES
1065  004562  005305                    DEC    R5            ;COUNT THE TIMER
1066  004564  001373                    BNE    1$            ;BR IF NO TIMEOUT
1067  004566  004767  007260           JSR    PC,SUER2      ;GO.SET UP ERROR INFO
1068  004572  005012                    CLR    (R2)          ;TURN IT ALL OFF
1069  004574  012767  004604  174442   MOV    #2$,$ESCAPE   ;COME BACK TO 2$ IN ALL CASES
1070  004602  104002                    ERROR+2             ;"DATA SET" INTR FAILED TO OCCUR
1071  004604                     2$:
1072  004604  000417                    BR     TST20         ;;<GO TO NEXT TEST>
1073  004606  005012           3$:    CLR    (R2)          ;ZERO THE RCSR
1074  004610  005167  174630           COM    INTFLG        ;SET THE SOFTWARE FLAG
1075  004614  000002                    RTI                  ;RETURN TO SENDER
1076  004616  032712  100000     4$:    BIT    #BIT15,(R2)   ;DID "DATA SET INT" GET SET BY INTR. SERVICE ??
1077  004622  001003                    BNE    5$            ;;<BR IF YES>
1078  004624  004767  007222           JSR    PC,SUER2      ;GO SET UP ERROR INFO
1079  004630  104002                    ERROR+2             ;DATA SET INTR. NOT SERVICED PROPERLY
1080  004632  020412           5$:    CMP    R4,(R2)       ;ALL BITS IN RCSR CLEAR ??
1081  004634  001403                    BEQ    TST20         ;;<BR IF YES>
```

```
1082  004636  004767  007210           JSR     PC,SUER2        ;GO SET UP ERROR INFO
1083  004642  104002                    ERROR+2                 ;INTR. SERVICE FAILED TO CLEAR RCSR
1084                           ;;*******************************************************************
1085                           ;*TEST 20        TEST THAT THE "BREAK" BIT CAN BE SET AND CLEARED
1086                           ;;*******************************************************************
1087  004644  000004           TST20:  SCOPE
1088  004646  032777  010000  174264    BIT     #SW12,@SWR      ;ARE WE TESTING /C OR /D MODEL?
1089  004654  001024                    BNE     TST21           ;;<BRANCH IF YES>
1090  004656  012704  000201            MOV     #201,R4         ;RESULT S/B = 201 IN XCSR
1091  004662  016702  174526            MOV     DLXCSR,R2       ;SET UP REGADR
1092  004666  052712  000001            BIS     #B1T0,(R2)      ;SET THE "BREAK" BIT
1093  004672  020412                    CMP     R4,(R2)         ;DID IT SET PROPERLY ??
1094  004674  001403                    BEQ     1$              ;;<BR IF YES>
1095  004676  004767  007150            JSR     PC,SUER2        ;GO SET UP ERROR INFO.
1096  004702  104002                    ERROR+2                 ;"BREAK" BIT FAILED TO SET PROPERLY
1097  004704  012704  000200    1$:     MOV     #200,R4         ;RESULT S/B = 200 IN XCSR
1098  004710  042712  000001            BIC     #BIT0,(R2)      ;CLEAR THE "BREAK" BIT
1099  004714  020412                    CMP     R4,(R2)         ;DID IT CLEAR PROPERLY ??
1100  004716  001403                    BEQ     TST21           ;;<BR IF YES>
1101  004720  004767  007126            JSR     PC,SUER2        ;GO SET UP ERROR INFO
1102  004724  104002                    ERROR+2                 ;"BREAK" FAILED TO CLEAR PROPERLY
1103
1104
1105                           ;;*******************************************************************
1106                           ;*TEST 21        TEST THAT A "RESET" CLEARS THE "BREAK" BIT
1107                           ;;*******************************************************************
1108  004726  000004           TST21:  SCOPE
1109  004730  012704  000200            MOV     #200,R4         ;RESULT S/B = 200
1110  004734  016702  174454            MOV     DLXCSR,R2       ;SET UP REGADR
1111  004740  052712  000001            BIS     #BIT0,(R2)      ;SET THE "BREAK" BIT
1112  004744  000005                    RESET                   ;CLEAR IT WITH A "RESET"
1113  004746  020412                    CMP     R4,(R2)         ;DID IT CLEAR ??
1114  004750  001403                    BEQ     TST22           ;;<BR IF YES>
1115  004752  004767  007074            JSR     PC,SUER2        ;GO SET UP ERROR INFO.
1116  004756  104002                    ERROR+2                 ;RESET INSTR. FAILED TO CLEAR "BREAK"
1117
```

```
1118                           ;;*******************************************************************
1119                           ;*TEST 22        TEST TO TURN AROUND NULL-DEL-NULL PATTERN
1120                           ;;*******************************************************************
1121  004760  000004           TST22:  SCOPE
1122  004762  012767  000001  174252    MOV     #1,STIMES       ;;DO 1 ITERATION
1123  004770  004767  007212            JSR     PC,SUVEC        ;GO SET UP VECTORS
1124  004774  005067  174430            CLR     RTRY            ;INITIALIZE RETRY FLAG
1125  005000  012767  014362  174430  1$: MOV   #LDOUT1,LDOUT   ;SET POINTER TO LOAD ROUTINE
1126  005006  004767  007222            JSR     PC,PRIME        ;GO SET UP BUFFERS AND DEVICE
1127  005012  005767  174404    2$:     TST     XFLG0           ;ANY HARD XMIT ERROS ??
1128  005016  001040                    BNE     5$              ;BR IF YES
1129  005020  005767  174400            TST     RFLG0           ;ANY HARD RECEIVER ERROR ??
1130  005024  001053                    BNE     7$              ;BR IF YES
1131  005026  005767  174374            TST     RFLG1           ;ANY SOFT RECEIVER ERRORS ??
1132  005032  001065                    BNE     9$              ;BR IF YES
1133  005034  022767  022260  174372    CMP     #BUFEND,IPTR    ;RECEIVED 256. BYTES ??
1134  005042  001003                    BNE     3$              ;BR IF NOT
1135  005044  004767  007456            JSR     PC,CHKDAT       ;GO CHECK THE DATA BUFFERS
1136  005050  000500                    BR      TST23           ;;<GO TO NEXT TEST>
1137  005052  005367  174362    3$:     DEC     TIMR1           ;DEC TIMEOUT COUNTER 1
1138  005056  001355                    BNE     2$              ;BR IF NO TIMEOUT
1139  005060  005367  174356            DEC     TIMR2           ;DEC TIMEOUT COUNTER 2
1140  005064  001352                    BNE     2$              ;BR IF NO TIMEOUT
1141  005066  042777  000100  174314    BIC     #100,@DLRCSR    ;TURN OFF THE INTRS.
1142  005074  042777  000104  174312    BIC     #104,@DLXCSR
1143  005102  104401  016342            TYPE    ,XMSG1          ;GO TYPE TIMEOUT MESSAGE
1144  005106  012767  005116  174130    MOV     #4$,SESCAPE     ;GO TO 4$ AFTER ERROR PRINT
1145  005114  104000                    ERROR                   ;PRINT ERROR PC
1146  005116                    4$:
1147  005116  000455                    BR      TST23           ;;<GO TO NEXT TEST>
1148  005120  016701  174264    5$:     MOV     DLRCSR,R1       ;PUT DEVADR IN R1
1149  005124  016702  174264            MOV     DLXCSR,R2       ;PUT REGADR IN R2
1150  005130  011203                    MOV     (R2),R3         ;GET THE WAS DATA
1151  005132  012704  000204            MOV     #204,R4         ;PUT S/B DATA IN R4
1152  005136  004767  006736            JSR     PC,SUERR1       ;GO SET UP ERROR INFO
1153  005142  012767  005152  174074    MOV     #6$,SESCAPE     ;GO TO 6$ AFTER PRINTING ERROR
1154  005150  104002                    ERROR+2                 ;TRANSMITTER FALSE INTERRUPT
1155  005152                    6$:
1156  005152  000437                    BR      TST23           ;;<GO TO NEXT TEST>
1157  005154  016701  174230    7$:     MOV     DLRCSR,R1       ;SAVE THE DEVADR
1158  005160  010102                    MOV     R1,R2           ;SAVE THE REGADR
1159  005162  011203                    MOV     (R2),R3         ;GET THE WAS DATA
1160  005164  012704  000200            MOV     #200,R4         ;RESULT S/B = 200
1161  005170  004767  006704            JSR     PC,SUERR1       ;GO SET UP ERROR INFO
1162  005174  012767  005204  174042    MOV     #8$,SESCAPE     ;GO TO 8$ AFTER ERROR PRINT
1163  005202  104002                    ERROR+2                 ;RECEIVER FALSE INTERRUPT
1164  005204                    8$:
1165  005204  000422                    BR      TST23           ;;<GO TO NEXT TEST>
1166  005206  016701  174176    9$:     MOV     DLRCSR,R1       ;SAVE THE DEVADR
1167  005212  016702  174174            MOV     DLRDBR,R2       ;SAVE REGADR
1168  005216  016703  173762            MOV     STMP1,R3        ;GET CONTENTS OF ERROR RDBR
1169  005222  004767  006652            JSR     PC,SUERR1       ;GO SETUP ERROR INFO
1170  005226  012767  005236  174010    MOV     #10$,SESCAPE    ;GO TO 10$ AFTER ERROR PRINT
1171  005234  104005                    ERROR+5                 ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN
1172  005236  005267  174166    10$:    INC     RTRY            ;COUNT ONE TRY
1173  005242  022767  000003  174160    CMP     #3,RTRY         ;TRIED THREE TIMES
```

```
1174  005250 001253                          BNE    1$              ;BR IF NOT
```

```
1175                                  ;;********************************************************
1176                                  ;*TEST 23        TEST TO TURN AROUND BINARY UP COUNT PATTERN
1177                                  ;;********************************************************
1178  005252 000004          TST23:   SCOPE
1179  005254 012767 000001 173760     MOV    #1,$TIMES       ;;DO 1 ITERATION
1180  005262 004767 006720            JSR    PC,SUVEC        ;GO SET UP VECTORS
1181  005266 005067 174136            CLR    RTRY            ;INITIALIZE RETRY FLAG
1182  005272 012767 014404 174136 1$: MOV    #LDOUT2,LDOUT   ;SET POINTER TO LOAD ROUTINE
1183  005300 004767 006730            JSR    PC,PRIME        ;GO SET UP BUFFERS AND DEVICE
1184  005304 005767 174112     2$:    TST    XFLG0           ;ANY HARD XMIT ERRORS ??
1185  005310 001040                   BNE    5$              ;BR IF YES
1186  005312 005767 174106            TST    RFLG0           ;ANY HARD RECEIVER ERROR ??
1187  005316 001053                   BNE    7$              ;BR IF YES
1188  005320 005767 174102            TST    RFLG1           ;ANY SOFT RECEIVER ERRORS ??
1189  005324 001065                   BNE    9$              ;BR IF YES
1190  005326 022767 022260 174100     CMP    #BUFEND,IPTR    ;RECEIVED 256. BYTES ??
1191  005334 001003                   BNE    3$              ;BR IF NOT
1192  005336 004767 007164            JSR    PC,CHKDAT       ;GO CHECK THE DATA BUFFERS
1193  005342 000500                   BR     TST24           ;;<GO TO NEXT TEST>
1194  005344 005367 174070     3$:    DEC    TIMR1           ;DEC TIMEOUT COUNTER 1
1195  005350 001355                   BNE    2$              ;BR IF NO TIMEOUT
1196  005352 005367 174064            DEC    TIMR2           ;DEC TIMEOUT COUNTER 2
1197  005356 001352                   BNE    2$              ;BR IF NO TIMEOUT
1198  005360 042777 000100 174022     BIC    #100,@DLRCSR    ;TURN OFF THE INTRS.
1199  005366 042777 000104 174020     BIC    #104,@DLXCSR
1200  005374 104401 016421            TYPE   ,XMSG2          ;GO TYPE TIMEOUT MESSAGE
1201  005400 012767 005410 173636     MOV    #4$,$ESCAPE     ;GO TO 4$ AFTER ERROR PRINT
1202  005406 104000                   ERROR                  ;PRINT ERROR PC
1203  005410                   4$:
1204  005410 000455                   BR     TST24           ;;<GO TO NEXT TEST>
1205  005412 016701 173772     5$:    MOV    DLRCSR,R1       ;PUT DEVADR IN R1
1206  005416 016702 173772            MOV    DLXCSR,R2       ;PUT REGADR IN R2
1207  005422 011203                   MOV    (R2),R3         ;GET THE WAS DATA
1208  005424 012704 000204            MOV    #204,R4         ;PUT S/B DATA IN R4
1209  005430 004767 006444            JSR    PC,SUERR1       ;GO SET UP ERROR INFO
1210  005434 012767 005444 173602     MOV    #6$,$ESCAPE     ;GO TO 6$ AFTER PRINTING ERROR
1211  005442 104002                   ERROR+2                ;TRANSMITTER FALSE INTERRUPT
1212  005444                   6$:
1213  005444 000437                   BR     TST24           ;;<GO TO NEXT TEST>
1214  005446 016701 173736     7$:    MOV    DLRCSR,R1       ;SAVE THE DEVADR
1215  005452 010102                   MOV    R1,R2           ;SAVE THE REGADR
1216  005454 011203                   MOV    (R2),R3         ;GET THE WAS DATA
1217  005456 012704 000200            MOV    #200,R4         ;RESULT S/B = 200
1218  005462 004767 006412            JSR    PC,SUERR1       ;GO SET UP ERROR INFO
1219  005466 012767 005476 173550     MOV    #8$,$ESCAPE     ;GO TO 8$ AFTER ERROR PRINT
1220  005474 104002                   ERROR+2                ;RECEIVER FALSE INTERRUPT
1221  005476                   8$:
1222  005476 000422                   BR     TST24           ;;<GO TO NEXT TEST>
1223  005500 016701 173704     9$:    MOV    DLRCSR,R1       ;SAVE THE DEVADR
1224  005504 016702 173702            MOV    DLRDBR,R2       ;SAVE REGADR
1225  005510 016703 173470            MOV    $TMP1,R3        ;GET CONTENTS OF ERROR RDBR
1226  005514 004767 006360            JSR    PC,SUERR1       ;GO SETUP ERROR INFO
1227  005520 012767 005530 173516     MOV    #10$,$ESCAPE    ;GO TO 10$ AFTER ERROR PRINT
1228  005526 104005                   ERROR+5                ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN
1229  005530 005267 173674     10$:   INC    RTRY            ;COUNT ONE TRY
1230  005534 022767 000003 173666     CMP    #3,RTRY         ;TRIED THREE TIMES
```

```
1231  005542  001253                         BNE    1$              ;BR IF NOT
```

```
1232                                    ;;************************************************************
1233                                    ;*TEST 24        TEST TO TURN AROUND BINARY DOWN COUNT PATTERN
1234                                    ;;************************************************************
1235  005544  000004              TST24:  SCOPE
1236  005546  012767  000001 173466        MOV    #1,$TIMES        ;;DO 1 ITERATION
1237  005554  004767  006426               JSR    PC,$UVEC         ;GO SET UP VECTORS
1238  005560  005067  173644               CLR    RTRY             ;INITIALIZE RETRY FLAG
1239  005564  012767  014424 173644  1$:   MOV    #LDOUT3,LDOUT     ;SET POINTER TO LOAD ROUTINE
1240  005572  004767  006436               JSR    PC,PRIME         ;GO SET UP BUFFERS AND DEVICE
1241  005576  005767  173620       2$:     TST    XFLG0            ;ANY HARD XMIT ERRORS ??
1242  005602  001040                       BNE    5$               ;BR IF YES
1243  005604  005767  173614               TST    RFLG0            ;ANY HARD RECEIVER ERROR ??
1244  005610  001053                       BNE    7$               ;BR IF YES
1245  005612  005767  173610               TST    RFLG1            ;ANY SOFT RECEIVER ERRORS ??
1246  005616  001065                       BNE    9$               ;BR IF YES
1247  005620  022767  022260 173606        CMP    #BUFEND,IPTR     ;RECEIVED 256. BYTES ??
1248  005626  001003                       BNE    3$               ;BR IF NOT
1249  005630  004767  006672               JSR    PC,CHKDAT        ;GO CHECK THE DATA BUFFERS
1250  005634  000500                       BR     TST25            ;;<GO TO NEXT TEST>
1251  005636  005367  173576       3$:     DEC    TIMR1            ;DEC TIMEOUT COUNTER 1
1252  005642  001355                       BNE    2$               ;BR IF NO TIMEOUT
1253  005644  005367  173572               DEC    TIMR2            ;DEC TIMEOUT COUNTER 2
1254  005650  001352                       BNE    2$               ;BR IF NO TIMEOUT
1255  005652  042777  000100 173530        BIC    #100,@DLRCSR     ;TURN OFF THE INTRS.
1256  005660  042777  000104 173526        BIC    #104,@DLXCSR
1257  005666  104401  016502               TYPE   ,XMSG3           ;GO TYPE TIMEOUT MESSAGE
1258  005672  012767  005702 173344        MOV    #4$,$ESCAPE      ;GO TO 4$ AFTER ERROR PRINT
1259  005700  104000                       ERROR                   ;PRINT ERROR PC
1260  005702                       4$:
1261  005702  000455                       BR     TST25            ;;<GO TO NEXT TEST>
1262  005704  016701  173500       5$:     MOV    DLRCSR,R1        ;PUT DEVADR IN R1
1263  005710  016702  173500               MOV    DLXCSR,R2        ;PUT REGADR IN R2
1264  005714  011203                       MOV    (R2),R3          ;GET THE WAS DATA
1265  005716  012704  000204               MOV    #204,R4          ;PUT S/B DATA IN R4
1266  005722  004767  006152               JSR    PC,$UERR1        ;GO SET UP ERROR INFO
1267  005726  012767  005736 173310        MOV    #6$,$ESCAPE      ;GO TO 6$ AFTER PRINTING ERROR
1268  005734  104002                       ERROR+2                 ;TRANSMITTER FALSE INTERRUPT
1269  005736                       6$:
1270  005736  000437                       BR     TST25            ;;<GO TO NEXT TEST>
1271  005740  016701  173444       7$:     MOV    DLRCSR,R1        ;SAVE THE DEVADR
1272  005744  010102                       MOV    R1,R2            ;SAVE THE REGADR
1273  005746  011203                       MOV    (R2),R3          ;GET THE WAS DATA
1274  005750  012704  000200               MOV    #200,R4          ;RESULT S/B = 200
1275  005754  004767  006120               JSR    PC,$UERR1        ;GO SET UP ERROR INFO
1276  005760  012767  005770 173256        MOV    #8$,$ESCAPE      ;GO TO 8$ AFTER ERROR PRINT
1277  005766  104002                       ERROR+2                 ;RECEIVER FALSE INTERRUPT
1278  005770                       8$:
1279  005770  000422                       BR     TST25            ;;<GO TO NEXT TEST>
1280  005772  016701  173412       9$:     MOV    DLRCSR,R1        ;SAVE THE DEVADR
1281  005776  016702  173410               MOV    DLRDBR,R2        ;SAVE REGADR
1282  006002  016703  173176               MOV    $TMP1,R3         ;GET CONTENTS OF ERROR RDBR
1283  006006  004767  006066               JSR    PC,$UERR1        ;GO SET UP ERROR INFO
1284  006012  012767  006022 173224        MOV    #10$,$ESCAPE     ;GO TO 10$ AFTER ERROR PRINT
1285  006020  104005                       ERROR+5                 ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN
1286  006022  005267  173402      10$:     INC    RTRY             ;COUNT ONE TRY
1287  006026  022767  000003 173374        CMP    #3,RTRY          ;TRIED THREE TIMES
```

```
 1288  006034  001253                         BNE     1$              ;BR IF NOT
```

```
 1289                                    ;;****************************************************************
 1290                                    ;*TEST 25        TEST TO TURN AROUND WORST CASE PATTERN
 1291                                    ;;****************************************************************
 1292  006036  000004              TST25:  SCOPE
 1293  006040  012767  000001  173174         MOV     #1,STIMES       ;;DO 1 ITERATION
 1294  006046  004767  006134             JSR     PC,SUVEC         ;GO SET UP VECTORS
 1295  006052  005067  173352             CLR     RTRY             ;INITIALIZE RETRY FLAG
 1296  006056  012767  014460  173352  1$:  MOV     #LDOUT4,LDOUT    ;SET POINTER TO LOAD ROUTINE
 1297  006064  004767  006144             JSR     PC,PRIME         ;GO SET UP BUFFERS AND DEVICE
 1298  006070  005767  173326         2$:  TST     XFLG0            ;ANY HARD XMIT ERRORS ??
 1299  006074  001042                      BNE     5$               ;BR IF YES
 1300  006076  005767  173322             TST     RFLG0            ;ANY HARD RECEIVER ERROR ??
 1301  006102  001056                      BNE     7$               ;BR IF YES
 1302  006104  005767  173316             TST     RFLG1            ;ANY SOFT RECEIVER ERRORS ??
 1303  006110  001071                      BNE     9$               ;BR IF YES
 1304  006112  022767  022260  173314         CMP     #BUFEND,IPTR    ;RECEIVED 256, BYTES ??
 1305  006120  001004                      BNE     3$               ;BR IF NOT
 1306  006122  004767  006400             JSR     PC,CHKDAT        ;GO CHECK THE DATA BUFFERS
 1307  006126  000167  002012             JMP     $EOP             ;GO TO NEXT TEST
 1308  006132  005367  173302         3$:  DEC     TIMR1            ;DEC TIMEOUT COUNTER 1
 1309  006136  001354                      BNE     2$               ;BR IF NO TIMEOUT
 1310  006140  005367  173276             DEC     TIMR2            ;DEC TIMEOUT COUNTER 2
 1311  006144  001351                      BNE     2$               ;BR IF NO TIMEOUT
 1312  006146  042777  000100  173234         BIC     #100,@DLRCSR    ;TURN OFF THE INTRS.
 1313  006154  042777  000104  173232         BIC     #104,@DLXCSR
 1314  006162  016565  016565             TYPE    ,XMSG4           ;GO TYPE TIMEOUT MESSAGE
 1315  006166  012767  006176  173050         MOV     #4$,$ESCAPE     ;GO TO 4$ AFTER ERROR PRINT
 1316  006174  104000                      ERROR                    ;PRINT ERROR PC
 1317  006176  000167  001742         4$:  JMP     $EOP             ;GO TO NEXT TEST
 1318  006202  016701  173202         5$:  MOV     DLRCSR,R1        ;PUT DEVADR IN R1
 1319  006206  016702  173202             MOV     DLXCSR,R2        ;PUT REGADR IN R2
 1320  006212  011203                      MOV     (R2),R3          ;GET THE WAS DATA
 1321  006214  012704  000204             MOV     #204,R4          ;PUT S/B DATA IN R4
 1322  006220  004767  005654             JSR     PC,SUERR1        ;GO SET UP ERROR INFO
 1323  006224  012767  006234  173012         MOV     #6$,$ESCAPE     ;GO TO 6$ AFTER PRINTING ERROR
 1324  006232  104002                      ERROR+2                  ;TRANSMITTER FALSE INTERRUPT
 1325  006234  000167  001704         6$:  JMP     $EOP             ;GO TO NEXT TEST
 1326  006240  016701  173144         7$:  MOV     DLRCSR,R1        ;SAVE THE DEVADR
 1327  006244  010102                      MOV     R1,R2            ;SAVE THE REGADR
 1328  006246  011203                      MOV     (R2),R3          ;GET THE WAS DATA
 1329  006250  012704  000200             MOV     #200,R4          ;RESULT S/B = 200
 1330  006254  004767  005620             JSR     PC,SUERR1        ;GO SET UP ERROR INFO
 1331  006260  012767  006270  172756         MOV     #8$,$ESCAPE     ;GO TO 8$ AFTER ERROR PRINT
 1332  006266  104002                      ERROR+2                  ;RECEIVER FALSE INTERRUPT
 1333  006270  000167  001650         8$:  JMP     $EOP             ;GO TO NEXT TEST
 1334  006274  016701  173110         9$:  MOV     DLRCSR,R1        ;SAVE THE DEVADR
 1335  006300  016702  173106             MOV     DLRDBR,R2        ;SAVE REGADR
 1336  006304  016703  172674             MOV     $TMP1,R3         ;GET CONTENTS OF ERROR RDBR
 1337  006310  004767  005564             JSR     PC,SUERR1        ;GO SETUP ERROR INFO
 1338  006314  012767  006324  172722         MOV     #10$,$ESCAPE    ;GO TO 10$ AFTER ERROR PRINT
 1339  006322  104005                      ERROR+5                  ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN
 1340  006324  005267  173100         10$: INC     RTRY             ;COUNT ONE TRY
 1341  006330  022767  000003  173072         CMP     #3,RTRY         ;TRIED THREE TIMES
 1342  006336  001247                      BNE     1$               ;BR IF NOT
 1343  006340  000167  001600             JMP     $EOP             ;GO TO END OF PASS ROUTINE
 1344
```

```
 1345                                              ;THIS IS PROGRAM #2
 1346                                              ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
 1347                                              ;          A)  SELECTION OF A TRANSMITTER DATA BUFFER
 1348                                              ;          B)  SELECTION OF A CHARACTER FOR CONTINUOUS TRANSFER
 1349                                              ;          C)  SELECTION OF AN EXPIRATION TIME IN MILLISECONDS
 1350                                              ;              BETWEEN EACH TRANSMITTER DATA BUFFER CHARACTER TRANSFER
 1351                                              ;          D)  A TIGHT SCOPE LOOP LOCK ON A SPECIFIC CHARACTER
 1352                                              ;
 1353
 1354  006344  012706  001100           PRG2:    MOV     #STACK,SP        ;INITIALIZE THE STACK POINTER
 1355  006350  012737  013066  000034            MOV     #8TRAP,@#TRAPVEC         ;TRAP VECTOR FOR TRAP CALLS
 1356  006356  012737  000340  000036            MOV     #340,@#TRAPVEC+2         ;LEVEL 7
 1357  006364  012737  010746  000030            MOV     #$ERROR,@#EMTVEC        ;EMT VECTOR FOR ERROR ROUTINE
 1358  006372  012737  000340  000032            MOV     #340,@#EMTVEC+2         ;LEVEL 7
 1359  006400  104401  016700            TYPE    ,PROG2M          ;INDICATE THAT USER SELECTED
 1360                                                               ;PROGRAM #2
 1361  006404  104401  020322           PRG2A:   TYPE    ,LIMTAD          ;ASK USER FOR THE TRANSMITTER
 1362                                                               ;DATA BUFFER ADDRESS OF THE DEVICE
 1363                                                               ;HE WISHES TO TEST
 1364  006410  104410                            RDOCT            ;ACCEPT THE ANSWER TYPED BY USER
 1365                                                               ;AND STORE ON TOP OF STACK
 1366                                              ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
 1367  006412  012602                            MOV     (SP)+,R2         ;GET THE ANSWER TYPED
 1368  006414  020227  176176                    CMP     R2,#176176       ;IS THE NUMBER TOO HIGH?
 1369  006420  101065                            BHI     REDO1            ;IF YES - GO TO RETRY SITUATION
 1370  006422  020227  175616                    CMP     R2,#175616       ;IS THE NUMBER TOO LOW?
 1371  006426  103462                            BLO     REDO1            ;IF YES - GO TO RETRY SITUATION
 1372  006430  132702  000001                    BITB    #BIT0,R2         ;NUMBER IS IN RANGE BUT IS IT
 1373                                                               ;ON AN EVEN BOUNDARY?
 1374  006434  001057                            BNE     REDO1            ;IF NOT GO TO RETRY SITUATION
 1375                                              ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
 1376  006436  010203                            MOV     R2,R3            ;GET THE USER RESPONSE
 1377  006440  142703  000370                    BICB    #370,R3          ;MASK OFF LOWER BYTE EXCEPT FOR
 1378                                                               ;LEAST SIGNIFICANT DIGIT
 1379  006444  122703  000006                    CMPB    #6,R3            ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
 1380                                                               ;USER RESPONSE EQUAL TO A SIX?
 1381  006450  001051                            BNE     REDO1            ;BRANCH IF NOT
 1382  006452  010267  172524                    MOV     R2,STMP0         ;THE TRANSMITTER ADDRESS
 1383                                                               ;TYPED IS OK - STORE FOR
 1384                                                               ;FUTURE USE
 1385                                              ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
 1386  006456  016746  171322                    MOV     ERRVEC,-(SP)     ;SAVE THE TIMEOUT VECTOR
 1387  006462  012767  006474  171314            MOV     #2$,ERRVEC       ;SET UP TIMEOUT SERVICE ADDRESS
 1388  006470  005712                            TST     (R2)             ;IF PRESENT WE WILL EXECUTE THE
 1389                                                               ;NEXT INSTRUCTION - IF NOT
 1390                                                               ;WE GO TO 2$:
 1391  006472  000412                            BR      4$               ;BRANCH IF PRESENT
 1392  006474  004767  005460           2$:      JSR     PC,SUERT2        ;GO SET UP FOR ERROR INFORMATION
 1393  006500  012767  006510  172536            MOV     #3$,SESCAPE      ;POINT OF RETURN AFTER ERROR REPORT
 1394  006506  104006                            ERROR   +6               ;XOBR REFERENCE CAUSED TIMEOUT
 1395  006510  022626                   3$:      CMP     (SP)+,(SP)+      ;CLEAN STACK FROM TIMEOUT
 1396  006514  012667  171266                    MOV     (SP)+,ERRVEC     ;RESTORE TIMEOUT VECTOR
 1397  006516  000426                            BR      REDO1            ;GO TO RETRY SITUATION
 1398  006520  012667  171260           4$:      MOV     (SP)+,ERRVEC     ;DEVICE REGISTER IS PRESENT!
 1399                                                               ;RESTORE TIMEOUT VECTOR
 1400                                              ;WE ARE NOW READY FOR THE CHARACTER TO BE TRANSMITTED, AND THE
```

```
 1401                                              ;DELAY TIME (IN MILLISECONDS) THAT IS TO TRANSPIRE BETWEEN
 1402                                              ;SUCCESSIVE CHARACTER TRANSFERS
 1403  006524  104401  020403           PRG2B:   TYPE    ,SELCAR          ;ASK USER FOR THE CHARACTER HE
 1404                                                               ;WISHES TO TRANSFER
 1405  006530  104410                            RDOCT            ;ACCEPT THE ANSWER TYPED BY
 1406                                                               ;USER AND STORE ON TOP OF STACK
 1407  006532  012667  172446                    MOV     (SP)+,STMP1      ;GET THE ANSWER TYPED
 1408                                              ;NOTE:   THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE
 1409                                              ;        OCTAL ASCII EQUIVALENT OF THE CHARACTER E.G. A=101
 1410  006536  104401  020511                    TYPE    ,SELDLY          ;ASK THE USER FOR THE DELAY
 1411                                                               ;IN MSEC (OCTAL NO.) BETWEEN
 1412                                                               ;CHARACTER TRANSFERS
 1413  006542  104410                            RDOCT            ;ACCEPT THE ANSWER TYPED BY
 1414                                                               ;USER AND STORE ON TOP OF STACK
 1415  006544  012667  172436                    MOV     (SP)+,STMP2      ;GET THE ANSWER TYPED
 1416  006550  116767  172432  000012  1$:       MOVB    STMP2,2$         ;SET THE DELAY COUNT ARGUMENT
 1417                                                               ;FOR TIMER ROUTINE
 1418  006556  116777  172422  172416            MOVB    STMP1,@STMP0     ;LOAD THE TRANSMITTER DATA
 1419                                                               ;BUFFER WITH THE CHARACTER
 1420  006564  004767  004750                    JSR     PC,DELAY         ;GO OFF TO WAIT THE SPECIFIED
```

```
1421                                                        ;NO. OF MSEC. BEFORE ISSUING
1422                                                        ;ANOTHER CHARACTER
1423  006570  000000              2$:    .WORD   0           ;THIS IS WHERE THE DELAY COUNT RESIDES
1424  006572  000766              BR      1$           ;GO BACK TO ISSUE ANOTHER CHARACTER
1425  006574  104401  001252      REDO01: TYPE   ,$QUES      ;TYPE A QUESTION MARK(?)
1426  006600  000167  177600              JMP    PRG2A        ;REITERATE THE XDBR QUESTION TO USER
1427                              ;THIS IS PROGRAM #3
1428                              ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
1429                              ;       A)  SELECTION OF A TRANSMITTER DATA BUFFER
1430                              ;       B)  SELECTION OF A CHARACTER FOR CONTINUOUS TRANSFER
1431                              ;            IN MAINTENANCE MODE
1432                              ;       C)  SELECTION OF AN EXPIRATION TIME IN MILLISECONDS
1433                              ;            BETWEEN EACH TRANSMITTER DATA BUFFER CHARACTER TRANSFER
1434                              ;       D)  A TIGHT SCOPE LOOP LOCK ON A SPECIFIC CHARACTER
1435                              ;
1436
1437  006604  012706  001100      PRG3:   MOV    #STACK,SP     ;INITIALIZE THE STACK POINTER
1438  006610  012737  013066  000034      MOV    #$TRAP,@#TRAPVEC     ;TRAP VECTOR FOR TRAP CALLS
1439  006616  012737  000340  000036      MOV    #340,@#TRAPVEC+2     ;LEVEL 7
1440  006624  012737  010746  000030      MOV    #$ERROR,@#EMTVEC     ;EMT VECTOR FOR ERROR ROUTINE
1441  006632  012737  000340  000032      MOV    #340,@#EMTVEC+2     ;LEVEL 7
1442  006640  104401  016744              TYPE   ,PROG3M       ;INDICATE THAT USER SELECTED
1443                                                        ;PROGRAM #3
1444  006644  104401  020322      PRG3A:  TYPE   ,LINTAD       ;ASK USER FOR THE TRANSMITTER DATA
1445                                                        ;BUFFER ADDRESS OF THE DEVICE
1446                                                        ;HE WISHES TO TEST
1447  006650  104410              RDOCT                ;ACCEPT THE ANSWER TYPED BY
1448                                                        ;USER AND STORE ON TOP OF STACK
1449                              ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1450  006652  012602              MOV    (SP)+,R2      ;GET THE ANSWER TYPED
1451  006654  020227  176176      CMP    R2,#176176    ;IS THE NUMBER TOO HIGH?
1452  006660  101071              BHI    REDO2         ;IF YES - GO TO RETRY SITUATION
1453  006662  020227  175616      CMP    R2,#175616    ;IS THE NUMBER TOO LOW?
1454  006666  103466              BLO    REDO2         ;IF YES - GO TO RETRY SITUATION
1455  006670  132702  000001      BITB   #BIT0,R2      ;NUMBER IS IN RANGE BUT IS IT
1456                                                        ;ON AN EVEN BOUNDARY?
1457  006674  001063              BNE    REDO2         ;IF NOT - GO TO RETRY SITUATION
1458                              ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XDBR DBR ADDRESS
1459  006676  010203              MOV    R2,R3         ;GET THE USER RESPONSE
1460  006700  142703  000370      BICB   #370,R3       ;MASK OFF LOWER BYTE EXCEPT FOR
1461                                                        ;LEAST SIGNIFICANT DIGIT
1462  006704  122703  000006      CMPB   #6,R3         ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
1463                                                        ;USER RESPONSE EQUAL TO A TWO?
1464  006710  001055              BNE    REDO2         ;BRANCH IF NOT
1465  006712  010267  172264      MOV    R2,STMP0      ;THE TRANSMITTER ADDRESS TYPED IS
1466                                                        ;OK - STORE FOR FUTURE USE
1467                              ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
1468  006716  016746  171062      MOV    ERRVEC,-(SP)  ;SAVE THE TIMEOUT VECTOR
1469  006722  012767  006734  171054      MOV    #2$,ERRVEC    ;SET UP TIMEOUT SERVICE ADDRESS
1470  006730  005712              TST    (R2)          ;IF PRESENT WE WILL EXECUTE THE
1471                                                        ;NEXT INSTRUCTION - IF NOT WE
1472                                                        ;GO TO 2$:
1473  006732  000412              BR     4$            ;BRANCH IF PRESENT
1474  006734  004767  005220      2$:    JSR    PC,SUERT2     ;GO SET UP FOR ERROR INFORMATION
1475  006740  012767  006750  172276      MOV    #3$,$ESCAPE   ;POINT OF RETURN AFTER ERROR REPORT
1476  006746  104006              ERROR  +6            ;XDBR REFERENCE CAUSED TIMEOUT
```

```
1477  006750  022626              3$:    CMP    (SP)+,(SP)+   ;CLEAN STACK FROM TIMEOUT
1478  006752  012667  171026      MOV    (SP)+,ERRVEC  ;RESTORE TIMEOUT VECTOR
1479  006756  000432              BR     REDO2         ;GO TO RETRY SITUATION
1480  006760  012667  171020      4$:    MOV    (SP)+,ERRVEC  ;DEVICE REGISTER IS PRESENT!
1481                                                        ;RESTORE TIMEOUT VECTOR
1482                              ;WE ARE NOW READY FOR THE CHARACTER TO BE TRANSMITTED, AND THE
1483                              ;DELAY TIME (IN MILLISECONDS) THAT IS TO TRANSPIRE BETWEEN SUCCESSIVE
1484                              ;CHARACTER TRANSFERS
1485  006764  104401  020403      PRG3B:  TYPE   ,SELCAR       ;ASK USER FOR THE CHARACTER
1486                                                        ;HE WISHES TO TRANSFER
1487  006770  104410              RDOCT                ;ACCEPT THE ANSWER TYPED BY USER
1488                                                        ;AND STORE ON TOP OF STACK
1489  006772  012667  172206      MOV    (SP)+,STMP1   ;GET THE ANSWER TYPED
1490                              ;NOTE:  THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE
1491                              ;        OCTAL ASCII EQUIVALENT OF THE CHARACTER E.G. B=102
1492  006776  104401  020511      TYPE   ,SELDLY       ;ASK THE USER FOR THE DELAY
1493                                                        ;IN MSEC (OCTAL NO.) BETWEEN
1494                                                        ;CHARACTER TRANSFERS
1495  007002  104410              RDOCT                ;ACCEPT THE ANSWER TYPED BY
1496                                                        ;USER AND STORE ON TOP OF STACK
1497  007004  012667  172176      MOV    (SP)+,STMP2   ;GET THE ANSWER TYPED
1498  007010  162702  000002      SUB    #2,R2         ;GET THE CORRESPONDING XCSR
1499                                                        ;ADDRESS FOR TRANSMITTER UNDER-
1500                                                        ;GOING TEST
1501  007014  052712  000004      1$:    BIS    #BIT2,(R2)    ;SET MAINTENANCE BIT IN XCSR
1502  007020  116767  172162  000012      MOVB   STMP2,2$     ;SET THE DELAY COUNT ARGUMENT
1503                                                        ;FOR TIMER ROUTINE
1504  007026  116777  172152  172146      MOVB   STMP1,@STMP0 ;LOAD THE TRANSMITTER DATA BUFFER
1505                                                        ;WITH THE CHARACTER
1506  007034  004767  004500              JSR    PC,DELAY      ;GO OFF TO WAIT THE SPECIFIED
1507                                                        ;NO. OF MSEC. BEFORE ISSUING
1508                                                        ;ANOTHER CHARACTER
1509  007040  000000              2$:    .WORD   0           ;THIS IS WHERE THE DELAY COUNT RESIDES
1510  007042  000764              BR     1$            ;GO BACK TO ISSUE ANOTHER CHARACTER
1511  007044  104401  001252      REDO2:  TYPE   ,$QUES        ;TYPE A QUESTION MARK(?)
1512  007050  000167  177570              JMP    PRG3A        ;REITERATE THE XDBR QUESTION TO
1513                                                        ;USER
1514
1515                              ;THIS IS PROGRAM #4
1516                              ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
1517                              ;       A)  SELECTION OF A TRANSMITTER DATA BUFFER
1518                              ;       B)  SELECTION OF A SINGLE CHARACTER TO BE SENT, RECEIVED
1519                              ;            AND CHECKED WITH MAINTENANCE BIT SET
1520                              ;
1521
1522  007054  012706  001100      PRG4:   MOV    #STACK,SP     ;INITIALIZE THE STACK POINTER
1523  007060  012737  013066  000034      MOV    #$TRAP,@#TRAPVEC     ;TRAP VECTOR FOR TRAP CALLS
1524  007066  012737  000340  000036      MOV    #340,@#TRAPVEC+2     ;LEVEL 7
1525  007074  012737  010746  000030      MOV    #$ERROR,@#EMTVEC     ;EMT VECTOR FOR ERROR ROUTINE
1526  007102  012737  000340  000032      MOV    #340,@#EMTVEC+2     ;LEVEL 7
1527  007110  104401  017010              TYPE   ,PROG4M       ;INDICATE THAT USER SELECTED
1528                                                        ;PROGRAM #4
1529  007114  104401  020322      PRG4A:  TYPE   ,LINTAD       ;ASK USER FOR THE TRANSMITTER
1530                                                        ;DATA BUFFER ADDRESS OF THE
1531                                                        ;DEVICE HE WISHES TO TEST
1532  007120  104410              RDOCT                ;ACCEPT THE ANSWER TYPED BY
```

```
1533                                                                    ;USER AND STORE ON TOP OF STACK
1534                                      ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
1535   007122  012602                          MOV     (SP)+,R2         ;GET THE ANSWER TYPED
1536   007124  020227  176176                  CMP     R2,#176176       ;IS THE NUMBER TOO HIGH?
1537   007130  101136                          BHI     REDO3            ;IF YES - GO TO RETRY SITUATION
1538   007132  020227  175616                  CMP     R2,#175616       ;IS THE NUMBER TOO LOW?
1539   007136  103533                          BLO     REDO3            ;IF YES - GO TO RETRY SITUATION
1540   007140  132702  000001                  BITB    #BIT0,R2         ;NUMBER IS IN RANGE BUT IS IT
1541                                                                    ;ON AN EVEN BOUNDARY?
1542   007144  001130                          BNE     REDO3            ;IF NO - GO TO RETRY SITUATION
1543                                      ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
1544   007146  010203                          MOV     R2,R3            ;GET THE USER RESPONSE
1545   007150  142703  000370                  BICB    #370,R3          ;MASK OFF LOWER BYTE EXCEPT FOR
1546                                                                    ;LEAST SIGNIFICANT DIGIT
1547   007154  122703  000006                  CMPB    #6,R3            ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
1548                                                                    ;USER RESPONSE EQUAL TO A SIX?
1549   007160  001122                          BNE     REDO3            ;BRANCH IF NOT
1550   007162  010267  172014                  MOV     R2,STMP0         ;THE TRANSMITTER ADDRESS TYPED
1551                                                                    ;IS OK - STORE FOR FUTURE USE
1552                                      ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
1553   007166  016746  170612                  MOV     ERRVEC,-(SP)     ;SAVE THE TIMEOUT VECTOR
1554   007172  012767  007204  170604          MOV     #2$,ERRVEC       ;SET UP TIMEOUT SERVICE ADDRESS
1555   007200  005712                          TST     (R2)             ;IF PRESENT WE WILL EXECUTE THE
1556                                                                    ;NEXT INSTRUCTION - IF NOT WE
1557                                                                    ;GO TO 2$;
1558   007202  000412                          BR      4$               ;BRANCH IF PRESENT
1559   007204  004767  004750          2$:     JSR     PC,SUERT2        ;GO SET UP FOR ERROR INFORMATION
1560   007210  012767  007220  172026          MOV     #3$,SESCAPE      ;POINT OF RETURN AFTER ERROR REPORT
1561   007216  104006                          ERROR   +6               ;XDBR REFERENCE CAUSED TIMEOUT
1562   007220  022626                  3$:     CMP     (SP)+,(SP)+      ;CLEAN STACK FROM TIMEOUT
1563   007222  012667  170556                  MOV     (SP)+,ERRVEC     ;RESTORE TIMEOUT VECTOR
1564   007226  000477                          BR      REDO3            ;GO TO RETRY SITUATION
1565   007230  012667  170550          4$:     MOV     (SP)+,ERRVEC     ;DEVICE REGISTER IS PRESENT!
1566                                                                    ;RESTORE TIMEOUT VECTOR
1567   007234  104401  020604                  TYPE    ,PSTALL          ;ASK THE USER IF HE DESIRES SOME
1568                                                                    ;RANDOM NO. OF MSEC. WAIT TIME
1569                                                                    ;BEFORE CHECKING FOR XCSR DONE
1570                                                                    ;FLAG
1571   007240  104410                          RDOCT                    ;ACCEPT THE ANSWER TYPED BY USER
1572                                                                    ;AND STORE ON TOP OF STACK
1573   007242  012667  171740                  MOV     (SP)+,STMP2      ;GET THE ANSWER TYPED
1574                                      ;WE ARE NOW READY FOR THE CHARACTER TO BE TRANSMITTED
1575   007246  104401  020403          PRG4B:  TYPE    ,SELCAR          ;ASK USER FOR THE CHARACTER HE
1576                                                                    ;WISHES TO TRANSFER
1577   007252  104410                          RDOCT                    ;ACCEPT THE ANSWER TYPED BY USER
1578                                                                    ;AND STORE ON TOP OF STACK
1579   007254  012667  171724                  MOV     (SP)+,STMP1      ;GET THE ANSWER TYPED
1580                                      ;NOTE:   THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE OCTAL
1581                                      ;        ASCII EQUIVALENT OF THE CHARACTER E.G. C=103
1582                                      ;
1583   007260  104401  017233          PRG4C:  TYPE,   LENGTH ;ASK USER FOR THE CHARACTER LENGTH
1584                                                                    ;FOR WHICH HIS DEVICE IS SET
1585   007264  104411                          RDDEC                    ;ACCEPT THE ANSWER TYPED BY USER
1586                                      ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1587   007266  012600                          MOV     (SP)+,R0         ;GET THE ANSWER TYPED
1588   007270  020027  000010                  CMP     R0,#8.           ;IS THE NUMBER TOO HIGH?
```

```
1589   007274  101060                          BHI     REDO3A           ;IF YES - GO TO RETRY SITUATION
1590   007276  020027  000005                  CMP     R0,#5,           ;IS THE NUMBER TOO LOW?
1591   007302  103455                          BLO     REDO3A           ;IF YES - GO TO RETRY SITUATION
1592   007304  010067  171724                  MOV     R0,STMP15        ;THE VALUE TYPED IS OK
1593                                                                    ;STORE FOR FUTURE USE
1594   007310  016767  171666  171672          MOV     STMP0,STMP3      ;GET THE XDBR ADDRESS
1595   007316  162767  000002  171664          SUB     #2,STMP3         ;FORM THE XCSR ADDRESS
1596   007324  005767  171656          1$:     TST     STMP2            ;DO WE RANDOM STALL?
1597   007330  001402                          BEQ     2$               ;BRANCH IF IT WASN'T DESIRED
1598   007332  004767  004246                  JSR     PC,STALL         ;GO STALL RANDOM VALUE OF MSEC,
1599   007336  004767  004352          2$:     JSR     PC,TIMETX        ;GO WAIT FOR TRANSMITTER DONE
1600                                                                    ;BIT TO SET
1601   007342  104401  017120                  TYPE    ,XDB             ;TYPE TRANSMITTER DONE BIT MESSAGE
1602   007346  104000                          ERROR   +0               ;XCSR DONE BIT NEVER SET
1603   007350  052777  000004  171632          BIS     #BIT2,@STMP3     ;SET THE MAINTENANCE BIT IN THE
1604                                                                    ;TRANSMITTER CONTROL STATUS REGISTER
1605   007356  016777  171622  171616          MOV     STMP1,@STMP0     ;LOAD TRANSMITTER DATA BUFFER
1606                                                                    ;WITH SELECTED CHARACTER
1607   007364  004767  004306                  JSR     PC,TIMERX        ;GO WAIT FOR RECEIVER DONE BIT
1608                                                                    ;TO SET
1609   007370  104401  017167                  TYPE    ,RDB             ;TYPE RECEIVER DONE BIT MESSAGE
1610   007374  104000                          ERROR   +0               ;RCSR DONE BIT NEVER SET
1611   007376  016767  171606  171606          MOV     STMP3,STMP4      ;GET THE TRANSMITTER CONTROL
1612                                                                    ;STATUS REGISTER ADDRESS
1613   007404  162767  000002  171600          SUB     #2,STMP4         ;FORM THE RECEIVER DATA BUFFER
1614                                                                    ;ADDRESS
1615   007412  017767  171574  171574          MOV     @STMP4,STMP5     ;STORE THE CHARACTER FROM THE
1616                                                                    ;RECEIVER BUFFER + REST OF CONTENTS
1617   007420  004767  004326                  JSR     PC,DATCHK        ;GO TO COMPARE EXPECTED & RECEIVED
1618                                                                    ;DATA
1619   007424  000737                          BR      1$               ;GO BACK TO ISSUE ANOTHER CHARACTER
1620   007426  104401  001252          REDO3:  TYPE    ,SQUES           ;TYPE A QUESTION MARK(?)
1621   007432  000167  177456                  JMP     PRG4A            ;REITERATE THE XDBR QUESTION TO USER
1622   007436  104401  001252          REDO3A: TYPE,   SQUES            ;TYPE '?' INDICATING USER TYPED
1623                                                                    ;SOMETHING WRONG FOR CHARACTER LENGTH
1624   007442  000167  177612                  JMP     PRG4C            ;GO BACK TO REISSUE QUESTION
1625
1626                                      ;THIS IS PROGRAM #5
1627                                      ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW USER PARAMETERS
1628                                      ;FOR RUNNING A BINARY COUNT IN MAINTENANCE MODE
1629                                      ;
1630
1631   007446  012706  001100          PRG5:   MOV     #STACK,SP        ;INITIALIZE THE STACK POINTER
1632   007452  012737  013066  000034          MOV     #STRAP,@#TRAPVEC         ;TRAP VECTOR FOR TRAP CALLS
1633   007460  012737  000340  000036          MOV     #340,@#TRAPVEC+2         ;LEVEL 7
1634   007466  012737  010746  000030          MOV     #SERROR,@#EMTVEC        ;EMT VECTOR FOR ERROR ROUTINE
1635   007474  012737  000340  000032          MOV     #340,@#EMTVEC+2         ;LEVEL 7
1636   007502  104401  017054                  TYPE    ,PROG5M          ;INDICATE THAT USER SELECTED
1637                                                                    ;PROGRAM #5
1638   007506  104401  020322          PRG5A:  TYPE    ,LINTAD          ;ASK USER FOR THE TRANSMITTER DATA
1639                                                                    ;BUFFER ADDRESS OF THE DEVICE
1640                                                                    ;HE WISHES TO TEST
1641   007512  104410                          RDOCT                    ;ACCEPT THE ANSWER TYPED BY USER
1642                                                                    ;AND STORE ON TOP OF STACK
1643                                      ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
1644   007514  012602                          MOV     (SP)+,R2         ;GET THE ANSWER TYPED
```

```
1645  007516  020227  176176                CMP     R2,#176176     ;IS THE NUMBER TOO HIGH?
1646  007522  101152                         BHI     REDO4          ;IF YES - GO TO RETRY SITUATION
1647  007524  020227  175616                CMP     R2,#175616     ;IS THE NUMBER TOO LOW?
1648  007530  103547                         BLO     REDO4          ;IF YES - GO TO RETRY SITUATION
1649  007532  132702  000001                BITB    #BIT0,R2       ;NUMBER IS IN RANGE BUT IS IT
1650                                                                ;ON AN EVEN BOUNDARY?
1651  007536  001144                         BNE     REDO4          ;IF NOT - GO TO RETRY SITUATION
1652                                  ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
1653  007540  010203                         MOV     R2,R3          ;GET THE USER RESPONSE
1654  007542  142703  000370                BICB    #370,R3 ;MASK OFF LOWER BYTE EXCEPT FOR
1655                                                                ;LEAST SIGNIFICANT DIGIT
1656  007546  122703  000006                CMPB    #6,R3          ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
1657                                                                ;USER RESPONSE EQUAL TO A SIX?
1658  007552  001136                         BNE     REDO4          ;BRANCH IF NOT
1659  007554  010267  171422                MOV     R2,$TMPO       ;THE TRANMITTER ADDRESS TYPED
1660                                                                ;IS OK - STORE FOR FUTURE USE
1661                                  ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
1662  007560  016746  170220                MOV     ERRVEC,-(SP)   ;SAVE THE TIMOUT VECTOR
1663  007564  012767  007576  170212        MOV     #2$,ERRVEC     ;SET UP TIMEOUT SERVICE ADDRESS
1664  007572  005712                         TST     (R2)           ;IF PRESENT WE WILL EXECUTE THE
1665                                                                ;NEXT INSTRUCTION - IF NOT WE
1666                                                                ;GO TO 2$:
1667  007574  000412                         BR      4$             ;BRANCH IF PRESENT
1668  007576  004767  004356         2$:     JSR     PC,SUERT2      ;GO SETUP FOR ERROR INFORMATION
1669  007602  012767  007612  171434        MOV     #3$,$ESCAPE    ;POINT OF RETURN AFTER ERROR REPORT
1670  007610  104006                         ERROR   +6             ;XDBR REFERENCE CAUSED TIMEOUT
1671  007612  022626         3$:     CMP     (SP)+,(SP)+    ;CLEAN STACK FROM TIMEOUT
1672  007614  012667  170164                MOV     (SP)+,ERRVEC   ;RESTORE TIMEOUT VECTOR
1673  007620  000513                         BR      REDO4          ;GO TO RETRY SITUATION
1674  007622  012667  170156         4$:     MOV     (SP)+,ERRVEC   ;DEVICE REGISTER IS PRESENT!
1675                                                                ;ASK THE USER IF HE DESIRES SOME
1676                                                                ;RANDOM NO. OF MSEC. WAIT TIME
1677                                                                ;BEFORE CHECKING XCSR DONE FLAG
1678  007626  104401  017233         PRG5C:  TYPE,   LENGTH ;ASK USER FOR THE CHARACTER LENGTH
1679                                                                ;FOR WHICH HIS DEVICE IS SET
1680  007632  104411                         RDDEC                  ;ACCEPT THE ANSWER TYPED BY USER
1681                                  ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1682  007634  012600                         MOV     (SP)+,R0       ;GET THE ANSWER TYPED
1683  007636  020027  000010                CMP     R0,#8.         ;IS THE NUMBER TOO HIGH?
1684  007642  101106                         BHI     REDO4A         ;IF YES - GO TO RETRY SITUATION
1685  007644  020027  000005                CMP     P0,#5.         ;IS THE NUMBER TOO LOW?
1686  007650  103503                         BLO     REDO4A         ;IF YES - GO TO RETRY SITUATION
1687  007652  010067  171356                MOV     R0,$TMP15      ;THE VALUE TYPED IS OK
1688                                                                ;STORE FOR FUTURE USE
1689  007656  104401  020604         TYPE    ,RSTALL        ;RANDOM NO. OF MSEC. WAIT TIME
1690  007662  104410                         RDOCT                  ;ACCEPT THE ANSWER TYPED BY USER
1691                                                                ;AND STORE ON TOP OF STACK
1692  007664  012667  171316                MOV     (SP)+,$TMP2    ;GET THE ANSWER TYPED
1693                                  ;WE ARE NOW READY TO INITIALIZE THE BINARY COUNT AND GET
1694                                  ;THE BINARY CHARACTER
1695                                  ;
1696  007670  012767  177777  171326        MOV     #-1,$TMP11     ;SET LEAD IN VARIABLE TO -1
1697  007676  016767  171322  171322 PRG5B:  MOV     $TMP11,$TMP12  ;STORE PREVIOUS BINARY CHARACTER
1698  007704  005267  171316                INC     $TMP12         ;FLIP BINARY CHARACTER AGAIN
1699  007710  042767  177400  171310        BIC     #177400,$TMP12 ;MASK TO 8 BITS
1700  007716  016767  171304  171300        MOV     $TMP12,$TMP11  ;STORE BINARY CHARACTER
```

```
1701  007724  016767  171276  171252        MOV     $TMP12,$TMP1   ;STORE BINARY CHARACTER
1702                                                                ;FOR FUTURE USE
1703  007732  016767  171244  171250        MOV     $TMPO,$TMP3    ;GET THE XDBR ADDRESS
1704  007740  162767  000002  171242        SUB     #2,$TMP3       ;FORM THE XCSR ADDRESS
1705  007746  005767  171234         1$:     TST     $TMP2          ;DO WE RANDOM STALL?
1706  007752  001402                         BEQ     2$             ;BRANCH IF IT WASN'T DESIRED
1707  007754  004767  003624                JSR     PC,STALL       ;GO STALL RANDOM VALUE OF MSEC.
1708  007760  004767  003730         2$:     JSR     PC,TIMETX      ;GO WAIT FOR TRANSMITTER DONE
1709                                                                ;BIT TO SET
1710  007764  104401  017120                TYPE    ,XDB           ;TYPE TRANSMITTER DONE BIT MESSAGE
1711  007770  104000                         ERROR   +0             ;XCSR DONE BIT NEVER SET
1712  007772  052777  000004  171210        BIS     #BIT2,@$TMP3   ;SET THE MAINTENANCE BIT IN THE
1713                                                                ;TRANSMITTER CONTROL STATUS REGISTER
1714  010000  016777  171200  171174        MOV     $TMP1,@$TMPO   ;LOAD TRANSMITTER DATA BUFFER
1715                                                                ;WITH SELECTED CHARACTER
1716  010006  004767  003664                JSR     PC,TIMERX      ;GO WAIT FOR RECEIVER DONE BIT TO SET
1717  010012  104401  017167                TYPE    ,RDB           ;TYPE RECEIVER DONE BIT MESSAGE
1718  010016  104000                         ERROR   +0             ;RCSR DONE BIT NEVER SET
1719  010020  016767  171164  171164        MOV     $TMP3,$TMP4    ;GET THE TRANSMITTER CONTROL
1720                                                                ;STATUS REGISTER ADDRESS
1721  010026  162767  000002  171156        SUB     #2,$TMP4       ;FORM THE RECEIVER DATA BUFFER
1722                                                                ;ADDRESS
1723  010034  017767  171152  171152        MOV     @$TMP4,$TMP5   ;STORE THE CHARACTER FROM THE
1724                                                                ;RECEIVER BUFFER + REST OF CONTENTS
1725  010042  004767  003704                JSR     PC,DATCHK      ;GO TO COMPARE EXPECTED & RECEIVED
1726                                                                ;DATA
1727  010046  000713                         BR      PRG5B          ;GO BACK TO ISSUE ANOTHER BINARY
1728                                                                ;CHARACTER
1729  010050  104401  001252         REDO4:  TYPE    ,$QUES         ;TYPE A QUESTION MARK(?)
1730  010054  000167  177426                JMP     PRG5A          ;GO BACK TO REITERATE XDBR QUESTION
1731  010060  104401  001252         REDO4A: TYPE,   $QUES          ;TYPE '?' INDICATING USER TYPED
1732                                                                ;SOMETHING WRONG FOR CHARACTER LENGTH
1733  010064  000167  177536                JMP     PRG5C          ;GO BACK TO REISSUE QUESTION
1734
1735                                  ;THIS ROUTINE WILL SET UP:
1736                                  ;       RCSR - RECEIVER STATUS REGISTER
1737                                  ;       RBUF - RECEIVER BUFFER REGISTER
1738                                  ;       XCSR - TRANSMITTER STATUS REGISTER
1739                                  ;       XBUF - TRANSMITTER BUFFER REGISTER
1740                                  ;INITIALLY, IN RESPONSE TO USER REPLY TO 1ST DEVICE HE WANTS
1741                                  ;TESTED, AND THEREAFTER, AT THE END OF A PROGRAM TO CYCLE THRU
1742                                  ;ALL DEVICES FOR MULTIPLE DEVICE TESTING (IF REQUIRED)
1743  010070  016767  171164  171312 DLADDR: MOV     DLBASE,DLRCSR  ;STORE RECEIVER STATUS REGISTER
1744                                                                ;OF CURRENT DEVICE
1745  010076  062767  000002  171154        ADD     #2,DLBASE      ;FORM RECEIVER BUFFER REGISTER
1746                                                                ;OF CURRENT DEVICE
1747  010104  016767  171150  171300        MOV     DLBASE,DLRDBR  ;STORE RECEIVER BUFFER REGISTER
1748                                                                ;OF CURRENT DEVICE
1749  010112  062767  000002  171140        ADD     #2,DLBASE      ;FORM TRANSMITTER STATUS REGISTER
1750                                                                ;OF OF CURRENT DEVICE
1751  010120  016767  171134  171266        MOV     DLBASE,DLXCSR  ;STORE TRANSMITTER STATUS REGISTER
1752                                                                ;OF CURRENT DEVICE
1753  010126  062767  000002  171124        ADD     #2,DLBASE      ;FOR TRANSMITTER BUFFER REGISTER
1754                                                                ;OF CURRENT DEVICE
1755  010134  016767  171120  171254        MOV     DLBASE,DLXDBR  ;STORE TRANSMITTER BUFFER REGISTER
1756                                                                ;OF CURRENT DEVICE
```

```
1757   010142  000207                         RTS    PC              ;RETURN
1758
1759                                  .SBTTL  END OF PASS ROUTINE
1760
1761                                  ;;*****************************************************
1762                                  ;*INCREMENT THE PASS NUMBER (SPASS)
1763                                  ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
1764                                  ;*IF THERES A MONITOR GO TO IT
1765                                  ;*IF THERE ISN'T JUMP TO RESTRT
1766
1767   010144                         $EOP:
1768                                  ;THIS NEXT SECTION UP TO THE NEXT LINE OF ASTERISKS WAS
1769                                  ;SUPPLIED BY THE MACRO 'EOPBEG'.  THE MACRO NAME APPEARS IN
1770                                  ;THE SOURCE PROGRAM AS ONE OF THE ARGUMENTS TO THE .$EOP
1771                                  ;SYSMAC UTILITY ROUTINE CALL.
1772   010144  000004                         SCOPE
1773   010146  105767  171120                 TSTB   MULTD           ;ARE WE RUNNING MULTIPLE DEVICES?
1774   010152  001501                         BEQ    3$              ;BRANCH IF NOT FOR NORMAL
1775                                                                 ;'END PASS # XX' TYPEOUT
1776   010154  005767  171114                 TST    ACTREG          ;ARE ANY DEVICES ACTIVE?
1777   010160  001011                         BNE    1$              ;BRANCH IF YES
1778   010162  104401  020110                 TYPE   ,FOULUP         ;INDICATE SOMETHING WRONG!
1779                                                                 ;MULTIPLE DEVICES ARE BEING
1780                                                                 ;RUN SUPPOSEDLY, BUT NONE ARE
1781                                                                 ;SHOWN ACTIVE
1782   010166  000000                         HALT                   ;WAIT FOR A USER RESPONSE
1783   010170  005067  171076                 CLR    MULTD           ;CLEAR MULTIPLE DEVICE FLAG
1784   010174  005067  171056                 CLR    TABFLG          ;CLEAR TABLE CREATION FLAG
1785   010200  000167  171566                 JMP    RESTRT          ;GET READY TO START ALL OVER
1786                                                                 ;AGAIN - ALL DEVICES WERE
1787                                                                 ;DESELECTED SOMEHOW!
1788   010204  062767  000010  171052 1$:     ADD    #10,BASEADD     ;FORM A NEW BASE
1789                                                                 ;ADDRESS FOR START OF NEXT BLOCK
1790                                                                 ;OF REGISTERS FOR NEXT DEVICE
1791   010212  062767  000010  171050         ADD    #10,BASEIV      ;FORM A NEW BASE ADDRESS FOR
1792                                                                 ;START OF NEXT BLOCK OF VECTORS
1793                                                                 ;FOR NEXT DEVICE
1794   010220  000241                         CLC                    ;CLEAR LAST DEVICE INDICATOR
1795   010222  006167  171050                 ROL    ROTADD          ;UPDATE NEXT POSSIBLE DEVICE ACTIVE
1796                                                                 ;POINTER
1797   010226  103431                         BCS    2$              ;BRANCH IF THIS WAS THE
1798                                                                 ;LAST DEVICE TO BE TESTED ON
1799                                                                 ;THIS PASS
1800   010230  036767  171042  171036         BIT    ROTADD,ACTREG   ;IS THIS DEVICE TRULY ACTIVE
1801                                                                 ;(AS PER USER)
1802   010236  001762                         BEQ    1$              ;BRANCH IF NOT TO SEE IF NEXT
1803                                                                 ;ONE POSSIBLE IS
1804   010240  016767  171020  171012         MOV    BASEADD,DLBASE  ;FORM THE RECEIVER STATUS REGISTER
1805                                                                 ;ADDRESS OF NEXT DEVICE
1806   010246  016767  171016  171144         MOV    BASEIV,DLVECT   ;GET NEXT DEVICE RCVR VECTOR
1807   010254  000240                         NOP
1808   010256  004767  177606                 JSR    PC,DLADDR       ;GO FORM DL ADDRESSES FOR NEXT
1809                                                                 ;DEVICE SELECTED
1810   010262  005067  170614                 CLR    $TSTNM          ;INITIALIZE TEST NO. FOR A PROGRAM
1811                                                                 ;PASS OVER THE NEXT DEVICE ACTIVE
1812   010266  005077  171122                 CLR    @DLXCSR         ;CLEAR OUT BOTH CSR'S
```

```
1813   010272  005077  171112                 CLR    @DLRCSR
1814   010276  005777  171110                 TST    @DLRDBR         ;FLUSH RCVR "DONE" BIT
1815   010302  005777  171104                 TST    @DLRDBR
1816   010306  000167  172456                 JMP    TST1            ;START TESTING THIS DEVICE
1817   010312                          2$:
1818                                  ;IF WE TAKE THIS PATH WE HAVE MADE A CYCLE THRU THE PROGRAM ONCE
1819                                  ;PER DEVICE I.E. - WE HAVE MADE A COMPLETE PASS
1820                                  ;NOW WE NEED TO RESTORE EVERYTHING FOR THE NEXT COMPLETE PASS
1821   010312  012767  000001  170756         MOV    #1,ROTADD       ;SET UP ROTATING POINTER FOR NEXT
1822                                                                 ;MULTIPLE PASS
1823   010320  016767  170736  170736         MOV    KEEPADD,BASEADD ;RESTORE BASE ADDRESS
1824   010326  016767  170734  170734         MOV    KEEPIV,BASEIV   ;RESTORE BASE INTERRUPT VECTOR
1825   010334  016767  170724  170716         MOV    BASEADD,DLBASE  ;RESTORE 1ST DEVICE BASE ADDRESS
1826   010342  016767  170722  171050         MOV    BASEIV,DLVECT   ;RESTORE 1ST DEVICE VECTOR ADDRESS
1827   010350  000240                         NOP
1828   010352  004767  177512                 JSR    PC,DLADDR       ;FORM ADDRESSES FOR 1ST DEVICE
1829   010356                          3$:
1830                                  ;;*****************************************************
1831   010356  005067  170520                 CLR    $TSTNM          ;;ZERO THE TEST NUMBER
1832   010362  005067  170654                 CLR    $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
1833   010366  005267  170506                 INC    $PASS           ;;INCREMENT THE PASS NUMBER
1834   010372  042767  100000  170500         BIC    #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
1835   010400  005327                         DEC    (PC)+           ;;LOOP?
1836   010402  000001                 $EOPCT: .WORD  1
1837   010404  003022                         BGT    $DOAGN          ;;YES
1838   010406  012737                         MOV    (PC)+,@(PC)+    ;;RESTORE COUNTER
1839   010410  000001                 $ENDCT: .WORD  1
1840   010412  010402                         $EOPCT
1841   010414  104401  010461                 TYPE   ,$ENDMG         ;;TYPE "END PASS #"
1842   010420  016746  170454                 MOV    $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
1843   010424  104405                         TYPDS                  ;;GO TYPE-DECIMAL ASCII WITH SIGN
1844   010426  104401  010456                 TYPE   ,$ENULL         ;;TYPE A NULL CHARACTER
1845   010432  013700  000042         $GET42: MOV    @#42,R0         ;;GET MONITOR ADDRESS
1846   010436  001405                         BEQ    $DOAGN          ;;BRANCH IF NO MONITOR
1847   010440  000005                         RESET                  ;;CLEAR THE WORLD
1848   010442  004710                 $ENDAD: JSR    PC,(R0)         ;;GO TO MONITOR
1849   010444  000240                         NOP                    ;;SAVE ROOM
1850   010446  000240                         NOP                    ;;FOR
1851   010450  000240                         NOP                    ;;ACT11
1852   010452                         $DOAGN:
1853   010452  000137                         JMP    @(PC)+          ;;RETURN
1854   010454  001772                 $RTNAD: .WORD  RESTRT
1855   010456     377     377     000 $ENULL: .BYTE  -1,-1,0         ;;NULL CHARACTER STRING
1856   010461     015  042412  042116 $ENDMG: .ASCIZ <15><12>/END PASS #/
1857   010466  050040  051501  020123
1858   010474  000043
1859                                  .SBTTL  SCOPE HANDLER ROUTINE
1860
1861                                  ;;*****************************************************
1862                                  ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
1863                                  ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
1864                                  ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
1865                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1866                                  ;*SW14=1         LOOP ON TEST
1867                                  ;*SW11=1         INHIBIT ITERATIONS
1868                                  ;*SW09=1         LOOP ON ERROR
```

```
 1869                                    ;*SW08=1         LOOP ON TEST IN SWR<7:0>
 1870                                    ;*CALL
 1871                                    ;*     SCOPE            ;;SCOPE=IOT
 1872
 1873   010476                          $SCOPE:
 1874   010476  032777  040000  170434  1$:    BIT     #BIT14,@SWR      ;;LOOP ON PRESENT TEST?
 1875   010504  001111                         BNE     5$               ;;YES IF SW14=1
 1876                                    ;#####START OF CODE FOR THE XOR TESTER#####
 1877   010506  000416                  $XTSTR: BR      6$               ;;IF RUNNING ON THE "XOR" TESTER CHANGE
 1878                                                                    ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
 1879   010510  013746  000004                  MOV     @#ERRVEC,-(SP)   ;;SAVE THE CONTENTS OF THE ERROR VECTOR
 1880   010514  012737  010534  000004          MOV     #5$,@#ERRVEC     ;;SET FOR TIMEOUT
 1881   010522  005737  177060                  TST     @#177060         ;;TIME OUT ON XOR?
 1882   010526  012637  000004                  MOV     (SP)+,@#ERRVEC   ;;RESTORE THE ERROR VECTOR
 1883   010532  000463                          BR      $SVLAD           ;;GO TO THE NEXT TEST
 1884   010534  022626                  5$:     CMP     (SP)+,(SP)+      ;;CLEAR THE STACK AFTER A TIME OUT
 1885   010536  012637  000004                  MOV     (SP)+,@#ERRVEC   ;;RESTORE THE ERROR VECTOR
 1886   010542  000423                          BR      7$               ;;LOOP ON THE PRESENT TEST
 1887   010544                          6$:;#####END OF CODE FOR THE XOR TESTER#####
 1888   010544  032777  000400  170366          BIT     #BIT08,@SWR      ;;LOOP ON SPEC. TEST?
 1889   010552  001404                          BEQ     2$               ;;BR IF NO
 1890   010554  127767  170360  170320          CMPB    @SWR,$TSTNM      ;;ON THE RIGHT TEST?    SWR<7:0>
 1891   010562  001462                          BEQ     $OVER            ;;BR IF YES
 1892   010564  105767  170313          2$:     TSTB    $ERFLG           ;;HAS AN ERROR OCCURRED?
 1893   010570  001421                          BEQ     3$               ;;BR IF NO
 1894   010572  126767  170317  170303          CMPB    $ERMAX,$ERFLG    ;;MAX. ERRORS FOR THIS TEST OCCURRED?
 1895   010600  101015                          BHI     3$               ;;BR IF NO
 1896   010602  032777  001000  170330          BIT     #BIT09,@SWR      ;;LOOP ON ERROR?
 1897   010610  001404                          BEQ     4$               ;;BR IF NO
 1898   010612  016767  170272  170266  7$:     MOV     $LPERR,$LPADR    ;;SET LOOP ADDRESS TO LAST SCOPE
 1899   010620  000443                          BR      $OVER
 1900   010622  105067  170255          4$:     CLRB    $ERFLG           ;;ZERO THE ERROR FLAG
 1901   010626  005067  170410                  CLR     $TIMES           ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
 1902   010632  000415                          BR      1$               ;;ESCAPE TO THE NEXT TEST
 1903   010634  032777  004000  170276  3$:     BIT     #BIT11,@SWR      ;;INHIBIT ITERATIONS?
 1904   010642  001011                          BNE     1$               ;;BR IF YES
 1905   010644  005767  170230                  TST     $PASS            ;;IF FIRST PASS OF PROGRAM
 1906   010650  001406                          BEQ     1$               ;;   INHIBIT ITERATIONS
 1907   010652  005267  170226                  INC     $ICNT            ;;INCREMENT ITERATION COUNT
 1908   010656  026767  170360  170220          CMP     $TIMES,$ICNT     ;;CHECK THE NUMBER OF ITERATIONS MADE
 1909   010664  002021                          BGE     $OVER            ;;BR IF MORE ITERATION REQUIRED
 1910   010666  012767  000001  170210  1$:     MOV     #1,$ICNT         ;;REINITIALIZE THE ITERATION COUNTER
 1911   010674  016767  000044  170340          MOV     $MXCNT,$TIMES    ;;SET NUMBER OF ITERATIONS TO DO
 1912   010702  105267  170174          $SVLAD: INCB    $TSTNM           ;;COUNT TEST NUMBERS
 1913   010706  011667  170174                  MOV     (SP),$LPADR      ;;SAVE SCOPE LOOP ADDRESS
 1914   010712  011667  170172                  MOV     (SP),$LPERR      ;;SAVE ERROR LOOP ADDRESS
 1915   010716  005067  170322                  CLR     $ESCAPE          ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 1916   010722  112767  000001  170165          MOVB    #1,$ERMAX        ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
 1917   010730  016777  170146  170204  $OVER:  MOV     $TSTNM,@DISPLAY  ;;DISPLAY TEST NUMBER
 1918   010736  016716  170144                  MOV     $LPADR,(SP)      ;;FUDGE RETURN ADDRESS
 1919   010742  000002                          RTI                      ;;FIXES PS
 1920   010744  000100                  $MXCNT: 100                      ;;MAX. NUMBER OF ITERATIONS
 1921
 1922                                    .SBTTL  ERROR HANDLER ROUTINE
 1923
 1924                                    ;;*****************************************************************
```

```
 1925                                    ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
 1926                                    ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
 1927                                    ;*AND GO TO $ERRTYP ON ERROR
 1928                                    ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 1929                                    ;*SW15=1          HALT ON ERROR
 1930                                    ;*SW13=1          INHIBIT ERROR TYPEOUTS
 1931                                    ;*SW10=1          BELL ON ERROR   ,
 1932                                    ;*SW09=1          LOOP ON ERROR
 1933                                    ;*CALL
 1934                                    ;*     ERROR   N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
 1935
 1936   010746                          $ERROR:
 1937   010746  105267  170131          7$:    INCB    $ERFLG           ;;SET THE ERROR FLAG
 1938   010752  001775                         BEQ     7$               ;;DON'T LET THE FLAG GO TO ZERO
 1939   010754  016777  170122  170160         MOV     $TSTNM,@DISPLAY  ;;DISPLAY TEST NUMBER AND ERROR FLAG
 1940   010762  032777  002000  170150         BIT     #BIT10,@SWR      ;;BELL ON ERROR?
 1941   010770  001402                         BEQ     1$               ;;NO - SKIP
 1942   010772  104401  001246                 TYPE    ,$BELL           ;;RING BELL
 1943   010776  005267  170110         1$:     INC     $ERTTL           ;;COUNT THE NUMBER OF ERRORS
 1944   011002  011667  170110                 MOV     (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
 1945   011006  162767  000002  170102         SUB     #2,$ERRPC
 1946   011014  117767  170076  170072         MOVB    @$ERRPC,$ITEMB   ;;STRIP AND SAVE THE ERROR ITEM CODE
 1947   011022  032777  020000  170110         BIT     #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
 1948   011030  001004                         BNE     20$              ;;SKIP TYPEOUTS
 1949   011032  004767  000044                 JSR     PC,$ERRTYP       ;;GO TO USER ERROR ROUTINE
 1950   011036  104401  001253                 TYPE    ,$CRLF
 1951   011042                         20$:
 1952   011042  005777  170072         2$:     TST     @SWR             ;;HALT ON ERROR
 1953   011046  100001                         BPL     3$               ;;SKIP IF CONTINUE
 1954   011050  000000                         HALT                     ;;HALT ON ERROR!
 1955   011052  032777  001000  170060  3$:    BIT     #BIT09,@SWR      ;;LOOP ON ERROR SWITCH SET?
 1956   011060  001402                         BEQ     4$               ;;BR IF NO
 1957   011062  016716  170022                 MOV     $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
 1958   011066  005767  170152         4$:     TST     $ESCAPE          ;;CHECK FOR AN ESCAPE ADDRESS
 1959   011072  001402                         BEQ     5$               ;;BR IF NONE
 1960   011074  016716  170144                 MOV     $ESCAPE,(SP)     ;;FUDGE RETURN ADDRESS FOR ESCAPE
 1961   011100                         5$:
 1962   011100  000002                         RTI                      ;;RETURN
 1963
 1964                                    .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
 1965
 1966                                    ;;*****************************************************************
 1967                                    ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
 1968                                    ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
 1969                                    ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
 1970
 1971   011102                          $ERRTYP:
 1972   011102  104401  001253                 TYPE    ,$CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
 1973   011106  010046                          MOV     R0,-(SP)         ;;SAVE R0
 1974   011110  005000                          CLR     R0               ;;PICKUP THE ITEM INDEX
 1975   011112  153700  001114                  BISB    @#$ITEMB,R0
 1976   011116  001004                          BNE     1$               ;;IF ITEM NUMBER IS ZERO, JUST
 1977                                                                    ;;TYPE THE PC OF THE ERROR
 1978   011120  016746  167772                  MOV     $ERRPC,-(SP)     ;;SAVE $ERRPC FOR TYPEOUT
 1979                                                                    ;;ERROR ADDRESS
 1980   011124  104402                          TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
```

```
1981  011126  000426                        BR      6$              ;;GET OUT
1982  011130  005300             1$:        DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
1983  011132  006300                        ASL     R0              ;;       WORK FOR THE ERROR TABLE
1984  011134  006300                        ASL     R0
1985  011136  006300                        ASL     R0
1986  011140  062700  001310                ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
1987  011144  012067  000004                MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
1988  011150  001404                         BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
1989  011152  104401                        TYPE                    ;;TYPE THE "ERROR MESSAGE"
1990  011154  000000             2$:        .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
1991  011156  104401  001253                TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
1992  011162  012067  000004     3$:        MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
1993  011166  001404                         BEQ     5$              ;;SKIP TYPEOUT IF 0
1994  011170  104401                        TYPE                    ;;TYPE THE "DATA HEADER"
1995  011172  000000             4$:        .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
1996  011174  104401  001253                TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
1997  011200  011000             5$:        MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
1998  011202  001004                         BNE     7$              ;;GO TYPE THE DATA
1999  011204  012600             6$:        MOV     (SP)+,R0        ;;RESTORE R0
2000  011206  104401  001253                TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
2001  011212  000207                        RTS     PC              ;;RETURN
2002  011214                     7$:
2003  011214  013046                        MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
2004  011216  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2005  011220  005710                        TST     (R0)            ;;IS THERE ANOTHER NUMBER?
2006  011222  001770                         BEQ     6$              ;;BR IF NO
2007  011224  104401  011232                TYPE    ,8$             ;;TYPE TWO(2) SPACES
2008  011230  000771                        BR      7$              ;;LOOP
2009  011232  020040     000     8$:        .ASCIZ  /  /            ;;TWO(2) SPACES
2010  011236                                .EVEN
2011
2012                              .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
2013
2014                              ;;*****************************************************************
2015                              ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2016                              ;*OCTAL (ASCII) NUMBER AND TYPE IT.
2017                              ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2018                              ;*CALL:
2019                              ;*       MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
2020                              ;*       TYPOS                   ;;CALL FOR TYPEOUT
2021                              ;*       .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2022                              ;*       .BYTE   M               ;;M=1 OR 0
2023                              ;*                                ;;1=TYPE LEADING ZEROS
2024                              ;*                                ;;0=SUPPRESS LEADING ZEROS
2025                              ;*
2026                              ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2027                              ;*$TYPOS OR $TYPOC
2028                              ;*CALL:
2029                              ;*       MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
2030                              ;*       TYPON                   ;;CALL FOR TYPEOUT
2031                              ;*
2032                              ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2033                              ;*CALL:
2034                              ;*       MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
2035                              ;*       TYPOC                   ;;CALL FOR TYPEOUT
2036
```

```
2037  011236  017646  000000     $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
2038  011242  116667  000001  000211      MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
2039  011250  112667  000207              MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
2040  011254  062716  000002              ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
2041  011260  000406                      BR      $TYPON
2042  011262  112767  000001  000171  $TYPOC: MOVB    #1,$OFILL    ;;SET THE ZERO FILL SWITCH
2043  011270  112767  000006  000165      MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
2044  011276  112767  000005  000154  $TYPON: MOVB    #5,$OCNT     ;;SET THE ITERATION COUNT
2045  011304  010346                      MOV     R3,-(SP)        ;;SAVE R3
2046  011306  010446                      MOV     R4,-(SP)        ;;SAVE R4
2047  011310  010546                      MOV     R5,-(SP)        ;;SAVE R5
2048  011312  116704  000145              MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
2049  011316  005404                      NEG     R4
2050  011320  062704  000006              ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
2051  011324  110467  000132              MOVB    R4,$OMODE       ;;SAVE IT FOR USE
2052  011330  116704  000125              MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
2053  011334  016605  000012              MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
2054  011340  005003                      CLR     R3              ;;CLEAR THE OUTPUT WORD
2055  011342  006105             1$:      ROL     R5              ;;ROTATE MSB INTO "C"
2056  011344  000404                      BR      3$              ;;GO DO MSB
2057  011346  006105             2$:      ROL     R5              ;;FORM THIS DIGIT
2058  011350  006105                      ROL     R5
2059  011352  006105                      ROL     R5
2060  011354  010503                      MOV     R5,R3
2061  011356  006103             3$:      ROL     R3              ;;GET LSB OF THIS DIGIT
2062  011360  105367  000076              DECB    $OMODE          ;;TYPE THIS DIGIT?
2063  011364  100016                      BPL     7$              ;;BR IF NO
2064  011366  042703  177770              BIC     #177770,R3      ;;GET RID OF JUNK
2065  011372  001002                      BNE     4$              ;;TEST FOR 0
2066  011374  005704                      TST     R4              ;;SUPPRESS THIS 0?
2067  011376  001403                      BEQ     5$              ;;BR IF YES
2068  011400  005204             4$:      INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
2069  011402  052703  000060              BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
2070  011406  052703  000040     5$:      BIS     #'  ,R3         ;;MAKE ASCII IF NOT ALREADY
2071  011412  110367  000000              MOVB    R3,8$           ;;SAVE FOR TYPING
2072  011416  104401  011456              TYPE    ,8$             ;;GO TYPE THIS DIGIT
2073  011422  105367  000032     7$:      DECB    $OCNT           ;;COUNT BY 1
2074  011426  003347                      BGT     2$              ;;BR IF MORE TO DO
2075  011430  002402                      BLT     6$              ;;BR IF DONE
2076  011432  005204                      INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
2077  011434  000744                      BR      2$              ;;GO DO THE LAST DIGIT
2078  011436  012605             6$:      MOV     (SP)+,R5        ;;RESTORE R5
2079  011440  012604                      MOV     (SP)+,R4        ;;RESTORE R4
2080  011442  012603                      MOV     (SP)+,R3        ;;RESTORE R3
2081  011444  016666  000002  000004      MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
2082  011452  012616                      MOV     (SP)+,(SP)
2083  011454  000002                      RTI                     ;;RETURN
2084  011456  000        8$:      .BYTE   0               ;;STORAGE FOR ASCII DIGIT
2085  011457  000                 .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
2086  011460  000        $OCNT:   .BYTE   0               ;;OCTAL DIGIT COUNTER
2087  011461  000        $OFILL:  .BYTE   0               ;;ZERO FILL SWITCH
2088  011462  000000     $OMODE:  .WORD   0               ;;NUMBER OF DIGITS TO TYPE
2089
2090                      .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2091
2092                      ;;*****************************************************************
```

```
2093                                      ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
2094                                      ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
2095                                      ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
2096                                      ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
2097                                      ;*REPLACED WITH SPACES.
2098                                      ;*CALL:
2099                                      ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
2100                                      ;*      TYPDS                   ;;GO TO THE ROUTINE
2101
2102   011464                       $TYPDS:
2103   011464   010046                    MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2104   011466   010146                    MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2105   011470   010246                    MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2106   011472   010346                    MOV     R3,-(SP)        ;;PUSH R3 ON STACK
2107   011474   010546                    MOV     R5,-(SP)        ;;PUSH R5 ON STACK
2108   011476   012746   020200           MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
2109   011502   016605   000020           MOV     20(SP),R5       ;;GET THE INPUT NUMBER
2110   011506   100004                    BPL     1$              ;;BR IF INPUT IS POS.
2111   011510   005405                    NEG     R5              ;;MAKE THE BINARY NUMBER POS.
2112   011512   112766   000055 000001    MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
2113   011520   005000           1$:      CLR     R0              ;;ZERO THE CONSTANTS INDEX
2114   011522   012703   011700           MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
2115   011526   112723   000040           MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
2116   011532   005002           2$:      CLR     R2              ;;CLEAR THE BCD NUMBER
2117   011534   016001   011670           MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
2118   011540   160105           3$:      SUB     R1,R5           ;;FORM THIS BCD DIGIT
2119   011542   002402                    BLT     4$              ;;BR IF DONE
2120   011544   005202                    INC     R2              ;;INCREASE THE BCD DIGIT BY 1
2121   011546   000774                    BR      3$
2122   011550   060105           4$:      ADD     R1,R5           ;;ADD BACK THE CONSTANT
2123   011552   005702                    TST     R2              ;;CHECK IF BCD DIGIT=0
2124   011554   001002                    BNE     5$              ;;FALL THROUGH IF 0
2125   011556   105716                    TSTB    (SP)            ;;STILL DOING LEADING 0'S?
2126   011560   100407                    BMI     7$              ;;BR IF YES
2127   011562   106316           5$:      ASLB    (SP)            ;;MSD?
2128   011564   103003                    BCC     6$              ;;BR IF NO
2129   011566   116663   000001 177777    MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
2130   011574   052702   000060  6$:      BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
2131   011600   052702   000040  7$:      BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
2132   011604   110223                    MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
2133   011606   005720                    TST     (R0)+           ;;JUST INCREMENTING
2134   011610   020027   000010           CMP     R0,#10          ;;CHECK THE TABLE INDEX
2135   011614   002746                    BLT     2$              ;;GO DO THE NEXT DIGIT
2136   011616   003002                    BGT     8$              ;;GO TO EXIT
2137   011620   010502                    MOV     R5,R2           ;;GET THE LSD
2138   011622   000764                    BR      6$              ;;GO CHANGE TO ASCII
2139   011624   105726           8$:      TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
2140   011626   100003                    BPL     9$              ;;BR IF NO
2141   011630   116663   177777 177776    MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
2142   011636   105013           9$:      CLRB    (R3)            ;;SET THE TERMINATOR
2143   011640   012605                    MOV     (SP)+,R5        ;;POP STACK INTO R5
2144   011642   012603                    MOV     (SP)+,R3        ;;POP STACK INTO R3
2145   011644   012602                    MOV     (SP)+,R2        ;;POP STACK INTO R2
2146   011646   012601                    MOV     (SP)+,R1        ;;POP STACK INTO R1
2147   011650   012600                    MOV     (SP)+,R0        ;;POP STACK INTO R0
2148   011652   104401   011700           TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
```

```
2149   011656   016666   000002 000004    MOV     2(SP),4(SP)     ;;ADJUST THE STACK
2150   011664   012616                    MOV     (SP)+,(SP)
2151   011666   000002                    RTI                     ;;RETURN TO USER
2152   011670   023420           $DTBL:   10000.
2153   011672   001750                    1000.
2154   011674   000144                    100.
2155   011676   000012                    10.
2156   011700   000004           $DBLK:   .BLKW   4
2157
2158                              .SBTTL  TTY INPUT ROUTINE
2159
2160                              ;;*********************************************************
2161                              .ENABL  LSB
2162
2163                              .DSABL  LSB
2164
2165
2166                              ;;*********************************************************
2167                              ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2168                              ;*CALL:
2169                              ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
2170                              ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
2171                              ;*                             ;;WITH PARITY BIT STRIPPED OFF
2172                              ;
2173
2174   011710   011646           $RDCHR:  MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
2175   011712   016666   000004 000002    MOV     4(SP),2(SP)     ;;SAVE THE PS
2176   011720   105777   167220  1$:      TSTB    @$TKS           ;;WAIT FOR
2177   011724   100375                    BPL     1$              ;;A CHARACTER
2178   011726   117766   167214 000004    MOVB    @$TKB,4(SP)     ;;READ THE TTY
2179   011734   042766   177600 000004    BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
2180   011742   026627   000004 000023    CMP     4(SP),#23       ;;IS IT A CONTROL-S?
2181   011750   001013                    BNE     3$              ;;BRANCH IF NO
2182   011752   105777   167166  2$:      TSTB    @$TKS           ;;WAIT FOR A CHARACTER
2183   011756   100375                    BPL     2$              ;;LOOP UNTIL ITS THERE
2184   011760   117746   167162           MOVB    @$TKB,-(SP)     ;;GET CHARACTER
2185   011764   042716   177600           BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
2186   011770   022627   000021           CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
2187   011774   001366                    BNE     2$              ;;IF NOT DISCARD IT
2188   011776   000750                    BR      1$              ;;YES, RESUME
2189   012000   026627   000004 000140 3$: CMP    4(SP),#140      ;;IS IT UPPER CASE?
2190   012006   002407                    BLT     4$              ;;BRANCH IF YES
2191   012010   026627   000004 000175    CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
2192   012016   003003                    BGT     4$              ;;BRANCH IF YES
2193   012020   042766   000040 000004    BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
2194   012026   000002           4$:      RTI                     ;;GO BACK TO USER
2195                              ;;*********************************************************
2196                              ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2197                              ;*CALL:
2198                              ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
2199                              ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2200                              ;*                             ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
2201
2202   012030   010346           $RDLIN:  MOV     R3,-(SP)        ;;SAVE R3
2203   012032   005046                    CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
2204   012034   012703   012264  1$:      MOV     #$TTYIN,R3      ;;GET ADDRESS
```

```
2205  012040  022703  012274        28:      CMP     #STTYIN+8.,R3   ;;BUFFER FULL?
2206  012044  101456                          BLOS    48              ;;BR IF YES
2207  012046  104406                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
2208  012050  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
2209  012052  122713  000177        108:      CMPB    #177,(R3)       ;;IS IT A RUBOUT
2210  012056  001022                          BNE     58              ;;BR IF NO
2211  012060  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
2212  012062  001007                          BNE     68              ;;BR IF NO
2213  012064  112767  000134  000170          MOVB    #'\,98          ;;TYPE A BACK SLASH
2214  012072  104401  012262                  TYPE    ,98
2215  012076  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
2216  012102  005303        68:               DEC     R3              ;;BACKUP BY ONE
2217  012104  020327  012264                  CMP     R3,#STTYIN      ;;STACK EMPTY?
2218  012110  103434                          BLO     48              ;;BR IF YES
2219  012112  111367  000144                  MOVB    (R3),98         ;;SETUP TO TYPEOUT THE DELETED CHAR.
2220  012116  104401  012262                  TYPE    ,98             ;;GO TYPE
2221  012122  000746                          BR      28              ;;GO READ ANOTHER CHAR.
2222  012124  005716        58:               TST     (SP)            ;;RUBOUT KEY SET?
2223  012126  001406                          BEQ     78              ;;BR IF NO
2224  012130  112767  000134  000124          MOVB    #'\,98          ;;TYPE A BACK SLASH
2225  012136  104401  012262                  TYPE    ,98
2226  012142  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
2227  012144  122713  000025        78:       CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
2228  012150  001003                          BNE     88              ;;BR IF NO
2229  012152  104401  012274                  TYPE    ,8CNTLU         ;;TYPE A CONTROL "U"
2230  012156  000726                          BR      18              ;;GO START OVER
2231  012160  122713  000022        88:       CMPB    #22,(R3)        ;;IS CHARACTER A "^R"?
2232  012164  001011                          BNE     38              ;;BRANCH IF NO
2233  012166  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
2234  012170  104401  001253                  TYPE    ,8CRLF          ;;TYPE A "CR" & "LF"
2235  012174  104401  012264                  TYPE    ,8TTYIN         ;;TYPE THE INPUT STRING
2236  012200  000717                          BR      28              ;;GO PICKUP ANOTHER CHACTER
2237  012202  104401  001252        48:       TYPE    ,8QUES          ;;TYPE A "?"
2238  012206  000712                          BR      18              ;;CLEAR THE BUFFER AND LOOP
2239  012210  111367  000046        38:       MOVB    (R3),98         ;;ECHO THE CHARACTER
2240  012214  104401  012262                  TYPE    ,98
2241  012220  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
2242  012224  001305                          BNE     28              ;;LOOP IF NOT RETURN
2243  012226  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
2244  012232  104401  001254                  TYPE    ,8LF            ;;TYPE A LINE FEED
2245  012236  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
2246  012240  012603                          MOV     (SP)+,R3        ;;RESTORE R3
2247  012242  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
2248  012244  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
2249  012252  012766  012264  000004          MOV     #STTYIN,4(SP)
2250  012260  000002                          RTI                     ;;RETURN
2251  012262  000           98:       .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
2252  012263  000                     .BYTE   0               ;;TERMINATOR
2253  012264  000010        STTYIN:   .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
2254  012274  052536  005015  000  8CNTLU:  .ASCIZ  /^U/<15><12>   ;;CONTROL "U"
2255  012301  136     006507  000012  8CNTLG:  .ASCIZ  /^G/<15><12>   ;;CONTROL "G"
2256  012306  005015  053523  020122  8MSWR:  .ASCIZ  <15><12>/SWR = /
2257  012314  020075  000
2258  012317  040     047040  053505  8MNEW:  .ASCIZ  / NEW = /
2259  012324  036440  000040
2260
```

```
2261                          .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
2262
2263                          ;;******************************************************************
2264                          ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2265                          ;*CHANGE IT TO BINARY.
2266                          ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
2267                          ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
2268                          ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
2269                          ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
2270                          ;*CALL:
2271                          ;*      RDOCT                   ;;READ AN OCTAL NUMBER
2272                          ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
2273                          ;*                              ;;HIGH ORDER BITS ARE IN 8HIOCT
2274
2275  012330  011646        8RDOCT:  MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
2276  012332  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
2277  012340  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2278  012342  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2279  012344  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2280  012346  104407        18:       RDLIN                   ;;READ AN ASCIZ LINE
2281  012350  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
2282  012352  010067  000100                  MOV     R0,58           ;;AND SAVE IT
2283  012356  005001                          CLR     R1              ;;CLEAR DATA WORD
2284  012360  005002                          CLR     R2
2285  012362  112046        28:       MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
2286  012366  001420                          BEQ     38              ;;IF ZERO GET OUT
2287  012366  122716  000060                  CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
2288  012372  003026                          BGT     48              ;;IS AN OCTAL DIGIT
2289  012374  122716  000067                  CMPB    #'7,(SP)
2290  012400  002423                          BLT     48
2291  012402  006301                          ASL     R1              ;;*2
2292  012404  006102                          ROL     R2
2293  012406  006301                          ASL     R1              ;;*4
2294  012410  006102                          ROL     R2
2295  012412  006301                          ASL     R1              ;;*8
2296  012414  006102                          ROL     R2
2297  012416  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
2298  012422  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
2299  012424  000756                          BR      28              ;;LOOP
2300  012426  005726        38:       TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
2301  012430  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
2302  012434  010267  000026                  MOV     R2,8HIOCT
2303  012440  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
2304  012442  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
2305  012444  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
2306  012446  000002                          RTI                     ;;RETURN
2307  012450  005726        48:       TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
2308  012452  105010                          CLRB    (R0)            ;;SET A TERMINATOR
2309  012454  104401                          TYPE                    ;;TYPE UP THRU THE BAD CHAR.
2310  012456  000000        58:       .WORD   0
2311  012460  104401  001252          TYPE    ,8QUES          ;;"?" "CR" & "LF"
2312  012464  000730                          BR      18              ;;TRY AGAIN
2313  012466  000000        8HIOCT:   .WORD   0               ;;HIGH ORDER BITS GO HERE
2314
2315                          .SBTTL  TYPE ROUTINE
2316
```

```
2317                                     ;;************************************************************
2318                                     ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
2319                                     ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
2320                                     ;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
2321                                     ;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
2322                                     ;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
2323                                     ;*
2324                                     ;*CALL:
2325                                     ;*1) USING A TRAP INSTRUCTION
2326                                     ;*       TYPE    ,MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2327                                     ;*OR
2328                                     ;*       TYPE
2329                                     ;*       MESADR
2330                                     ;*
2331
2332  012470  105767  166463            $TYPE: TSTB    $TPFLG           ;;IS THERE A TERMINAL?
2333  012474  100002                           BPL     1$               ;;BR IF YES
2334  012476  000000                           HALT                     ;;HALT HERE IF NO TERMINAL
2335  012500  000407                           BR      3$               ;;LEAVE
2336  012502  010046                    1$:    MOV     R0,-(SP)         ;;SAVE R0
2337  012504  017600  000002                   MOV     @2(SP),R0        ;;GET ADDRESS OF ASCIZ STRING
2338  012510  112046                    2$:    MOVB    (R0)+,-(SP)      ;;PUSH CHARACTER TO BE TYPED ONTO STACK
2339  012512  001005                           BNE     4$               ;;BR IF IT ISN'T THE TERMINATOR
2340  012514  005726                           TST     (SP)+            ;;IF TERMINATOR POP IT OFF THE STACK
2341  012516  012600                    60$:   MOV     (SP)+,R0         ;;RESTORE R0
2342  012520  062716  000002            3$:    ADD     #2,(SP)          ;;ADJUST RETURN PC
2343  012524  000002                           RTI                      ;;RETURN
2344  012526  122716  000011            4$:    CMPB    #HT,(SP)         ;;BRANCH IF <HT>
2345  012532  001430                           BEQ     8$
2346  012534  122716  000200                   CMPB    #CRLF,(SP)       ;;BRANCH IF NOT <CRLF>
2347  012540  001006                           BNE     5$
2348  012542  005726                           TST     (SP)+            ;;POP <CR><LF> EQUIV
2349  012544  104401                           TYPE                     ;;TYPE A CR AND LF
2350  012546  001253                           $CRLF
2351  012550  105067  000130                   CLRB    $CHARCNT         ;;CLEAR CHARACTER COUNT
2352  012554  000755                           BR      2$               ;;GET NEXT CHARACTER
2353  012556  004767  000056            5$:    JSR     PC,$TYPEC        ;;GO TYPE THIS CHARACTER
2354  012562  126726  166370            6$:    CMPB    $FILLC,(SP)+     ;;IS IT TIME FOR FILLER CHARS.?
2355  012566  001350                           BNE     2$               ;;IF NO GO GET NEXT CHAR.
2356  012570  016746  166360                   MOV     $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
2357                                                                    ;;AND THE NULL CHAR.
2358  012574  105366  000001            7$:    DECB    1(SP)            ;;DOES A NULL NEED TO BE TYPED?
2359  012600  002770                           BLT     6$               ;;BR IF NO--GO POP THE NULL OFF OF STACK
2360  012602  004767  000032                   JSR     PC,$TYPEC        ;;GO TYPE A NULL
2361  012606  105367  000072                   DECB    $CHARCNT         ;;DO NOT COUNT AS A COUNT
2362  012612  000770                           BR      7$               ;;LOOP
2363
2364                                     ;HORIZONTAL TAB PROCESSOR
2365
2366  012614  112716  000040            8$:    MOVB    #' ,(SP)         ;;REPLACE TAB WITH SPACE
2367  012620  004767  000014            9$:    JSR     PC,$TYPEC        ;;TYPE A SPACE
2368  012624  132767  000007  000052           BITB    #7,$CHARCNT      ;;BRANCH IF NOT AT
2369  012632  001372                           BNE     9$               ;;TAB STOP
2370  012634  005726                           TST     (SP)+            ;;POP SPACE OFF STACK
2371  012636  000724                           BR      2$               ;;GET NEXT CHARACTER
2372  012640  105777  166304            $TYPEC: TSTB   @$TPS            ;;WAIT UNTIL PRINTER IS READY
```

```
2373  012644  100375                           BPL     $TYPEC
2374  012646  116677  000002  166276           MOVB    2(SP),@$TPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
2375  012654  122766  000015  000002           CMPB    #CR,2(SP)        ;;IS CHARACTER A CARRIAGE RETURN?
2376  012662  001003                           BNE     1$               ;;BRANCH IF NO
2377  012664  105067  000014                   CLRB    $CHARCNT         ;;YES--CLEAR CHARACTER COUNT
2378  012670  000406                           BR      $TYPEX           ;;EXIT
2379  012672  122766  000012  000002   1$:     CMPB    #LF,2(SP)        ;;IS CHARACTER A LINE FEED?
2380  012700  001402                           BEQ     $TYPEX           ;;BRANCH IF YES
2381  012702  105227                           INCB    (PC)+            ;;COUNT THE CHARACTER
2382  012704  000000            $CHARCNT:.WORD 0                        ;;CHARACTER COUNT STORAGE
2383  012706  000207            $TYPEX: RTS     PC
2384
2385
2386                                     .SBTTL  READ A DECIMAL NUMBER FROM THE TTY
2387
2388                                     ;;************************************************************
2389                                     ;*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
2390                                     ;*CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
2391                                     ;*ARE READ A "?" FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
2392                                     ;*THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
2393                                     ;*USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
2394                                     ;*POSITIVE 32767 TO NEGATIVE 32768.
2395                                     ;*CALL:
2396                                     ;*       RDDEC                   ;;READ A DECIMAL NUMBER
2397                                     ;*       RETURN HERE             ;;NUMBER IS ON TOP OF THE STACK
2398                                     ;
2399
2400  012710  011646                    $RDDEC: MOV    (SP),-(SP)       ;;PROVIDE SPACE FOR
2401  012712  016666  000004  000002           MOV     4(SP),2(SP)      ;;THE INPUT NUMBER
2402  012720  010046                           MOV     R0,-(SP)         ;;PUSH R0 ON STACK
2403  012722  010146                           MOV     R1,-(SP)         ;;PUSH R1 ON STACK
2404  012724  010246                           MOV     R2,-(SP)         ;;PUSH R2 ON STACK
2405  012726  104407                    1$:    RDLIN                    ;;READ AN ASCIZ LINE
2406  012730  012600                           MOV     (SP)+,R0         ;;ADDRESS OF 1ST CHAR.
2407  012732  010067  000120                   MOV     R0,6$            ;;SAVE INCASE OF BAD INPUT
2408  012736  005046                           CLR     -(SP)            ;;CLEAR DATA WORD
2409  012740  005002                           CLR     R2               ;;SIGN SET POSITIVE
2410  012742  122710  000055                   CMPB    #'-,(R0)         ;;SEE IF A MINUS SIGN WAS TYPED
2411  012746  001001                           BNE     2$               ;;BR IF NO MINUS SIGN
2412  012750  112002                           MOVB    (R0)+,R2         ;;SAVE FOR LATER USE
2413  012752  112001                    2$:    MOVB    (R0)+,R1         ;;PICKUP THIS CHARACTER
2414  012754  001424                           BEQ     3$               ;;GET OUT IF ZERO
2415  012756  122701  000060                   CMPB    #'0,R1           ;;MAKE SURE THIS CHARACTER
2416  012762  003032                           BGT     5$               ;;IS A DIGIT BETWEEN 0 & 9
2417  012764  122701  000071                   CMPB    #'9,R1
2418  012770  002427                           BLT     5$
2419  012772  032716  170000                   BIT     #^C7777,(SP)     ;;DON'T LET NUMBER GET TO BIG
2420  012776  001024                           BNE     5$               ;;BR IF NUMBER WOULD OVERFLOW
2421  013000  006316                           ASL     (SP)             ;;*2
2422  013002  011646                           MOV     (SP),-(SP)       ;;SAVE FOR LATER
2423  013004  006316                           ASL     (SP)             ;;*4
2424  013006  006316                           ASL     (SP)             ;;*8.
2425  013010  062616                           ADD     (SP)+,(SP)       ;;*10.
2426  013012  102416                           BVS     5$               ;;OVERFLOW ISN'T ALLOWED
2427  013014  162701  000060                   SUB     #'0,R1           ;;STRIP AWAY THE ASCII JUNK
2428  013020  060116                           ADD     R1,(SP)          ;;ADD IN THIS DIGIT
```

```
2429  013022  102412                      BVS     5$              ;;OVERFLOW ISN'T ALLOWED
2430  013024  000752                      BR      2$              ;;LOOP
2431  013026  005702           3$:        TST     R2              ;;CHECK IF NUMBER IS NEG
2432  013030  001401                      BEQ     4$              ;;BR IF NO
2433  013032  005416                      NEG     (SP)            ;;YES--NEGATE THE NUMBER
2434  013034  012666  000012   4$:        MOV     (SP)+,12(SP)    ;;SAVE THE RESULT
2435  013040  012602                      MOV     (SP)+,R2        ;;POP STACK INTO R2
2436  013042  012601                      MOV     (SP)+,R1        ;;POP STACK INTO R1
2437  013044  012600                      MOV     (SP)+,R0        ;;POP STACK INTO R0
2438  013046  000002                      RTI                     ;;RETURN
2439
2440  013050  005726           5$:        TST     (SP)+           ;;CLEAN PARTIAL NUMBER FROM STACK
2441  013052  105010                      CLRB    (R0)            ;;SET A TERMINATOR
2442  013054  104401                      TYPE                    ;;TYPE THE INPUT UP TO BAD CHAR.
2443  013056  000000           6$:        .WORD   0               ;;POINTER GOES HERE
2444  013060  104401  001252              TYPE    ,$QUES          ;;"?" "CR" &"LF"
2445  013064  000720                      BR      1$              ;;TRY AGAIN
2446
2447                           .SBTTL  TRAP DECODER
2448
2449                           ;;************************************************************
2450                           ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
2451                           ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
2452                           ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
2453                           ;*GO TO THAT ROUTINE.
2454
2455  013066  010046           $TRAP:     MOV     R0,-(SP)        ;;SAVE R0
2456  013070  016600  000002              MOV     2(SP),R0        ;;GET TRAP ADDRESS
2457  013074  005740                      TST     -(R0)           ;;BACKUP BY 2
2458  013076  111000                      MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
2459  013100  006300                      ASL     R0              ;;POSITION FOR INDEXING
2460  013102  016000  013122              MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
2461  013106  000200                      RTS     R0              ;;GO TO ROUTINE
2462
2463
2464                           ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
2465
2466  013110  011646           $TRAP2:    MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
2467  013112  016666  000004 000002       MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
2468  013120  000002                      RTI                     ;;RESTORE THE PSW
2469
2470                           .SBTTL  TRAP TABLE
2471
2472                           ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
2473                           ;*BY THE "TRAP" INSTRUCTION.
2474
2475                           ;         ROUTINE
2476                           ;         -------
2477  013122  013110           $TRPAD:    .WORD   $TRAP2
2478  013124  012470                      $TYPE   ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
2479  013126  011262                      $TYPOC  ;;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
2480  013130  011236                      $TYPOS  ;;CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
2481  013132  011276                      $TYPON  ;;CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
2482  013134  011464                      $TYPDS  ;;CALL=TYPDS     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
2483
2484
```

```
2485  013136  011710                      $RDCHR  ;;CALL=RDCHR     TRAP+6(104406)  TTY TYPEIN CHARACTER ROUTINE
2486  013140  012030                      $RDLIN  ;;CALL=RDLIN     TRAP+7(104407)  TTY TYPEIN STRING ROUTINE
2487  013142  012330                      $RDOCT  ;;CALL=RDOCT     TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY
2488  013144  012710                      $RDDEC  ;;CALL=RDDEC     TRAP+11(104411) READ A DECIMAL NUMBER FROM TTY
2489
2490                           .SBTTL  POWER DOWN AND UP ROUTINES
2491
2492                           ;;************************************************************
2493                           ;POWER DOWN ROUTINE
2494  013146  012737  013312 000024  $PWRDN: MOV   #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
2495  013154  012737  000340 000026      MOV     #340,@#PWRVEC+2 ;;PRIO:7
2496  013162  010046                      MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2497  013164  010146                      MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2498  013166  010246                      MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2499  013170  010346                      MOV     R3,-(SP)        ;;PUSH R3 ON STACK
2500  013172  010446                      MOV     R4,-(SP)        ;;PUSH R4 ON STACK
2501  013174  010546                      MOV     R5,-(SP)        ;;PUSH R5 ON STACK
2502  013176  017746  165736              MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
2503  013202  010667  000110              MOV     SP,$SAVR        ;;SAVE SP
2504  013206  012737  013220 000024       MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
2505  013214  000000                      HALT
2506  013216  000776                      BR      .-2             ;;HANG UP
2507
2508                           ;;************************************************************
2509                           ;POWER UP ROUTINE
2510  013220  012737  013312 000024  $PWRUP: MOV   #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
2511  013226  016706  000064              MOV     $SAVR6,SP       ;;GET SP
2512  013232  005067  000060              CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
2513  013236  005267  000054   1$:        INC     $SAVR6          ;;WAIT FOR THE INC
2514  013242  001375                      BNE     1$              ;;OF WORD
2515  013244  012737  165670              MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
2516  013250  012605                      MOV     (SP)+,R5        ;;POP STACK INTO R5
2517  013252  012604                      MOV     (SP)+,R4        ;;POP STACK INTO R4
2518  013254  012603                      MOV     (SP)+,R3        ;;POP STACK INTO R3
2519  013256  012602                      MOV     (SP)+,R2        ;;POP STACK INTO R2
2520  013260  012601                      MOV     (SP)+,R1        ;;POP STACK INTO R1
2521  013262  012600                      MOV     (SP)+,R0        ;;POP STACK INTO R0
2522  013264  012737  013146 000024       MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
2523  013272  012737  000340 000026       MOV     #340,@#PWRVEC+2 ;;PRIO:7
2524  013300  104401                      TYPE                    ;;REPORT THE POWER FAILURE
2525  013302  013320           $PWRMG:    .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
2526  013304  012716                      MOV     (PC)+,(SP)      ;;RESTART AT RESTRT
2527  013306  001772           $PWRAD:    .WORD   RESTRT          ;;RESTART ADDRESS
2528  013310  000002                      RTI
2529  013312  000000           $ILLUP:    HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
2530  013314  000776                      BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
2531  013316  000000           $SAVR6:    0                       ;;PUT THE SP HERE
2532  013320  005015  047520 042527  $POWER: .ASCIZ  <15><12>"POWER"
2533  013326  000122
2534                                      .EVEN
2535
2536                           ;;************************************************************
2537                           ;TRANSMIT INTERRUPT SERVICE ROUTINE FOR 256. BYTE BLOCK TRANSFERS
2538                           ;;************************************************************
2539
2540  013330  105777  166060   XINT:      TSTB    @DLXCSR         ;"READY" SET ??
```

```
2541  013334  100416                        BMI     1$                  ;BR IF YES
2542  013336  013767  177776  165636        MOV     @#PSW,$TMP0         ;SAVE THE ERROR PSW
2543  013344  010667  165626                MOV     SP,$REG6            ;SAVE THE ERROR STACK POINTER
2544  013350  005167  166046                COM     XFLG0               ;SET XMIT SOFTWARE ERROR FLAG
2545  013354  042777  000100  166026        BIC     #100,@DLRCSR        ;TURN OFF THE INTERRUPT ENABLES
2546  013362  042777  000100  166024        BIC     #100,@DLXCSR
2547  013370  000411                        BR      2$                  ;GO TO EXIT
2548  013372  022767  021660  166032   1$:  CMP     #DLBUFI,OPTR        ;XMITTED 256, BYTES YET ??
2549  013400  001405                        BEQ     2$                  ;BR IF YES
2550  013402  117777  166024  166006        MOVB    @OPTR,@DLXDBR       ;OUTPUT A BYTE
2551  013410  005267  166016                INC     OPTR                ;UPDATE BUFFER POINTER
2552  013414  000002                   2$:  RTI                         ;RETURN TO MAINLINE TEST
2553
2554                                         ;;*************************************************************
2555                                         ;RECEIVER INTERRUPT SERVICE ROUTINE FOR 256, BYTE BLOCK TRANSFERS
2556                                         ;;*************************************************************
2557
2558  013416  105777  165766        RINT:    TSTB    @DLRCSR ;"DONE" SET ??
2559  013422  100410                         BMI     1$                  ;BR IF YES
2560  013424  013767  177776  165550         MOV     @#PSW,$TMP0         ;SAVE THE ERROR PSW
2561  013432  010667  165540                 MOV     SP,$REG6            ;SAVR THE ERROR STACK POINTER
2562  013436  005167  165762                 COM     RFLG0               ;SET HARD RCVR ERROR FLAG
2563  013442  000415                         BR      2$                  ;GO EXIT
2564  013444  005777  165742        1$:      TST     @DLRDBR             ;ANY SOFT ERRORS ??
2565  013450  100021                         BPL     3$                  ;BR IF NOT
2566  013452  013767  177776  165522         MOV     @#PSW,$TMP0         ;SAVE THE ERROR PSW
2567  013460  010667  165512                 MOV     SP,$REG6            ;SAVE THE ERROR STACK POINTER
2568  013464  017767  165722  165512         MOV     @DLRDBR,$TMP1       ;SAVE THE ERROR REGISTER IN TMP1
2569  013472  005167  165730                 COM     RFLG1               ;SET THE SOFT ERROR FLAG
2570  013476  042777  000100  165710  2$:    BIC     #100,@DLXCSR        ;TURN OFF THE INTR, ENABLES
2571  013504  042777  000100  165676         BIC     #100,@DLRCSR
2572  013512  000411                         BR      4$                  ;GO TO EXIT
2573  013514  022767  022260  165712  3$:    CMP     #BUFEND,IPTR        ;RECEIVED 256, BYTES YET ??
2574  013522  001405                         BEQ     4$                  ;BR IF YES
2575  013524  117777  165662  165702         MOVB    @DLRDBR,@IPTR       ;INPUT A BYTE FROM THE DL11
2576  013532  005267  165676                 INC     IPTR                ;UPDATE BUFFER POINTER
2577  013536  000002                   4$:   RTI                         ;RETURN TO MAINLINE TEST
2578
2579                                          ;THE FOLLOWING ROUTINE IS USED BY THE USER UTILITY PROGRAMS TO WAIT
2580                                          ;A SPECIFIED NO, OF MILLISECONDS BETWEEN CHARACTER TRANSFERS
2581
2582  013540  017667  000000  000034  DELAY: MOV     @(R6),DELCNT        ;GET THE NO, OF MSEC, DELAY COUNT
2583                                                                      ;TYPED IN BY USER
2584  013546  062716  000002                 ADD     #2,(R6)             ;SET UP THIS ROUTINE'S EXIT ADDRESS
2585  013552  005767  000024                 TST     DELCNT              ;IS THE DELAY COUNT ZERO?
2586  013556  001410                         BEQ     3$                  ;BRANCH IF YES
2587  013560  012746  000226        1$:      MOV     #226,-(SP)          ;PUSH A 1 MSEC, COUNT TO STACK
2588  013564  005316                2$:      DEC     (SP)                ;DECREMENT THE 1 MSEC, COUNT BY 1
2589  013566  001376                         BNE     2$                  ;BRANCH IF 1 MSEC, NOT EATEN
2590                                                                      ;AWAY YET
2591  013570  005726                         TST     (SP)+               ;RESET STACK AFTER 1 MSEC, TIME UP
2592  013572  005367  000004                 DEC     DELCNT              ;DECREMENT THE TOTAL NO, OF
2593                                                                      ;MSECS, COUNT
2594  013576  001370                         BNE     1$                  ;BRANCH IF WE HAVE MORE MSECS,
2595                                                                      ;TO WAIT
2596  013600  000207                3$:      RTS     PC                  ;GO BACK TO REISSUE A CHARACTER
```

```
2597  013602  000000                DELCNT:  .WORD   0                   ;THE NO, OF MSECS, NEEDED TO
2598                                                                      ;TRANSPIRE RESIDES HERE
2599                                          ;THE FOLLOWING ROUTINE IS USED BY USER PROGRAM #4 AND WILL ALLOW
2600                                          ;A RANDOM NUMBER OF MILLISECONDS BEFORE TRANSMISSION OF CHARACTER
2601                                          ;
2602  013604  016700  000062        STALL:   MOV     NUMONE,R0           ;GET THE LOW LIMIT
2603  013610  006100                         ROL     R0                  ;MULTIPLY BY 4
2604  013612  006100                         ROL     R0                  ;    .
2605  013614  066700  000054                 ADD     NUMTWO,R0           ;ADD IN THE HIGH LIMIT
2606  013620  010067  000046                 MOV     R0,NUMONE           ;STORE THIS AS NEW LOW LIMIT
2607  013624  006100                         ROL     R0                  ;MULTIPLY NEW LOW LIMIT BY 4
2608  013626  006100                         ROL     R0                  ;    .
2609  013630  066700  000040                 ADD     NUMTWO,R0           ;ADD IN THE HIGH LIMIT
2610  013634  006100                         ROL     R0                  ;MULTIPLY BY 4 AGAIN
2611  013636  006100                         ROL     R0                  ;    .
2612  013640  010067  000030                 MOV     R0,NUMTWO           ;STORE THIS AS NEW HIGH LIMIT
2613  013644  016700  000022                 MOV     NUMONE,R0           ;SAVE THE RANDOMLY GENERATED NO,
2614  013650  046700  165432                 BIC     STLMSK,R0           ;STRIP ALL BUT 1ST 5 BITS SO AS
2615                                                                      ;NOT TO ALLOW THE STALL TO BE TOO
2616                                                                      ;LARGE
2617  013654  001405                         BEQ     2$                  ;BRANCH IF RESULT WAS ZERO
2618  013656  010067  000004                 MOV     R0,1$               ;SET STALL TIME FOR DELAY ROUTINE
2619  013662  004767  177652                 JSR     PC,DELAY            ;GO OFF TO STALL
2620  013666  000000                1$:      .WORD   0                   ;THIS IS WHERE STALL TIME RESIDES
2621  013670  000207                2$:      RTS     PC                  ;RETURN TO ISSUE CHARACTER
2622  013672  001233                NUMONE:  001233                      ;LOW LIMIT FOR RANDOM NO,
2623  013674  007622                NUMTWO:  007622                      ;HIGH LIMIT FOR RANDOM NO,
2624                                          ;THE FOLLOWING ROUTINE CHECKS THE 'DONE' BIT FOR BOTH THE RECEIVER
2625                                          ;AND TRANSMITTER, THIS ROUTINE IS USED BY PROGRAM #4
2626                                          ;
2627  013676  016767  165306  000044  TIMERX: MOV    $TMP3,DUT           ;GET THE TRANSMITTER CONTROL
2628                                                                      ;STATUS REGISTER ADDRESS
2629  013704  162767  000004  000036         SUB     #4,DUT              ;FORM THE RECEIVER CONTROL
2630                                                                      ;STATUS REGISTER ADDRESS
2631  013712  000403                         BR      TCONT               ;GO TO TIME OUT THE RECEIVERS'
2632                                                                      ;DONE BIT
2633  013714  016767  165270  000026  TIMETX: MOV    $TMP3,DUT           ;GET THE TRANSMITTER CONTROL
2634                                                                      ;STATUS REGISTER ADDRESS
2635  013722  005067  165270        TCONT:   CLR     $TMP6               ;INITIALIZE A TIME COUNT
2636  013726  005267  165264        1$:      INC     $TMP6               ;INCREMENT THE TIME COUNT
2637  013732  001405                         BEQ     2$                  ;BRANCH IF TIME COUNTER OVERFLOWED
2638                                                                      ;INDICATING DONE BIT NEVER SET
2639                                                                      ;WITH PLENTY OF TIME ELAPSED
2640  013734  105777  000010                 TSTB    @DUT                ;SEE IF DONE BIT IS SET YET
2641  013740  100372                         BPL     1$                  ;WAIT SOME MORE IF IT ISN'T
2642  013742  062716  000006                 ADD     #6,@R6              ;DONE BIT IS SET - SET UP EXIT
2643                                                                      ;RETURN TO SKIP ERROR REPORT
2644  013746  000207                2$:      RTS     PC                  ;RETURN TO PROGRAM #4
2645  013750  000000                DUT:     .WORD   0                   ;THIS IS WHERE THE RCSR OR XCSR
2646                                                                      ;ADDRESS RESIDES
2647                                          ;THIS ROUTINE IS USED BY PROGRAMS #4 & 5, AND WILL CHECK FOR CORRECT
2648                                          ;EXPECTED AND RECEIVED DATA, IN ADDITION TO ANY ERROR BITS
2649                                          ;
2650  013752  016767  165236  165236  DATCHK: MOV    $TMP5,$TMP6         ;GET THE CONTENTS OF THE RECEIVER
2651                                                                      ;BUFFER
2652  013760  016767  165226  165200         MOV     $TMP4,$REG2         ;STORE THE ADDRESS OF THE RECEIVER
```

```
 2653                                                       ;DATA BUFFER
 2654   013766  016767  165174  165170        MOV    $REG2,$REG1     ;GET THE ADDRESS OF THE RECEIVER .
 2655                                                       ;DATA BUFFER
 2656   013774  162767  000002  165162        SUB    #2,$REG1        ;FORM THE ADDRESS OF THE RECEIVER
 2657                                                       ;STATUS REGISTER FROM IT
 2658   014002  016767  165206  165160        MOV    $TMP5,$REG3     ;STORE THE CONTENTS OF THE RECEIVER
 2659                                                       ;DATA BUFFER
 2660   014010  032767  170000  165200        BIT    #170000,$TMP6   ;ARE ANY ERROR BITS SET?
 2661   014016  001013                        BNE    1$              ;BRANCH IF YES
 2662   014020  004767  000720                JSR    PC,UPMASK       ;GO TO MASK OFF BITS AS A FUNCTION OF
 2663                                                       ;CHARACTER LENGTH( 5, 6, 7, OR 8 BITS)
 2664   014024  026767  165164  165200        CMP    $TMP5,$TMP14    ;WAS RECEIVED CHARACTER THE
 2665                                                       ;SAME AS THE ONE TRANSMITTED?
 2666   014032  001406                        BEQ    2$              ;BRANCH IF YES
 2667   014034  016767  165172  165130        MOV    $TMP14,$REG4    ;STORE WHAT THE CONTENTS OF THE
 2668                                                       ;RECEIVER DATA BUFFER SHOULD BE
 2669   014042  104010                        ERROR  +10             ;DATA RECEIVED WRONG!
 2670   014044  000401                        BR     2$              ;GET SET TO RETURN AFTER ERROR REPORT
 2671   014046  104007                   1$:  ERROR  +7              ;ERROR BIT/S SET FROM TRANSMISSION
 2672   014050  000207                   2$:  RTS    PC              ;RETURN TO PROGRAM #4
 2673
 2674                                   ;;************************************************************
 2675                                   ;SUBROUTINE TO SETUP ERROR INFORMATION FOR ERROR MESSAGES
 2676                                   ;;************************************************************
 2677
```

```
 2678   014052  013767  177776  165122  SUER2: MOV    @#PSW,$TMP0    ;SAVE THE [PSW]
 2679   014060  016701  165324                 MOV    DLRCSR,R1      ;PUT DEVADR IN R1
 2680   014064  011203                         MOV    (R2),R3        ;PUT WAS INFO IN R3
 2681   014066  010667  165104                 MOV    SP,$REG6       ;SAVE THE [SP]
 2682   014072  062767  000002  165076         ADD    #2,$REG6       ;CORRECT FOR CALLING JSR
 2683   014100  116700  164776  SUERR1: MOVB   $TSTNM,R0      ;PUT TEST NO. IN R0
 2684   014104  010067  165052                 MOV    R0,$REG0       ;SAVE [R0] THRU [R4]
 2685   014110  010167  165050                 MOV    R1,$REG1
 2686   014114  010267  165046                 MOV    R2,$REG2
 2687   014120  010367  165044                 MOV    R3,$REG3
 2688   014124  010467  165042                 MOV    R4,$REG4
 2689   014130  000207                         RTS    PC             ;RETURN TO CALLING TEST
 2690
 2691   014132  013767  177776  165042  SUERT1: MOV   @#PSW,$TMP0    ;SAVE THE [PSW]
 2692   014140  116700  164736                 MOVB   $TSTNM,R0      ;PUT TEST NO. IN R0
 2693   014144  016701  165240                 MOV    DLRCSR,R1      ;PUT DEVADR IN R1
 2694   014150  010067  165006                 MOV    R0,$REG0       ;SAVE [R0]
 2695   014154  010167  165004                 MOV    R1,$REG1       ;SAVE [R1]
 2696   014160  013767  177776  165016  SUERT2: MOV   @#PSW,$TMP1    ;SAVE THE [PSW]
 2697   014166  010667  165004                 MOV    SP,$REG6       ;SAVE THE [SP]
 2698   014172  062767  000002  164776         ADD    #2,$REG6       ;CORRECT FOR CALLING JSR
 2699   014200  010267  164762                 MOV    R2,$REG2       ;SAVE [R2]
 2700   014204  000207                         RTS    PC             ;RETURN
 2701
 2702                                   ;SUBROUTINE TO SETUP VECTORS FOR 256. BYTE BLOCK TRANSFER TESTS
 2703
 2704   014206  016705  165206  SUVEC: MOV    DLVECT,R5      ;GET FIRST VECTOR ADDRESS
 2705   014212  012725  013416                 MOV    #RINT,(R5)+    ;SET UP RCVR VECTOR
 2706   014216  016725  165060                 MOV    DLPRI,(R5)+
 2707   014222  012725  013330                 MOV    #XINT,(R5)+    ;SET UP XMIT VECTOR
 2708   014226  016715  165050                 MOV    DLPRI,(R5)
 2709   014232  000207                         RTS    PC             ;RETURN TO CALLER
 2710
 2711                                   ;SUBROUTINE TO PRIME DATA BUFFERS AND DEVICE FOR 256. BYTE TRANSFER
 2712
 2713   014234  005077  165154  PRIME: CLR    @DLXCSR        ;CLEAR XMIT AND RCVR CSR'S
 2714   014240  005077  165144                 CLR    @DLRCSR
 2715   014244  005067  165152                 CLR    XFLG0          ;INITIALIZE ERROR FLAGS
 2716   014250  005067  165150                 CLR    RFLG0
 2717   014254  005067  165146                 CLR    RFLG1
 2718   014260  012767  021260  165144         MOV    #DLBUFO,OPTR   ;SET UP OUTPUT POINTER
 2719   014266  012767  021660  165140         MOV    #DLBUFI,IPTR   ;SET UP INPUT POINTER
 2720   014274  004767  000044                 JSR    PC,CLDLBF      ;GO CLEAR THE BUFFERS
 2721   014300  004777  165132                 JSR    PC,@LDOUT      ;GO SET UP THE PATTERN
 2722   014304  005067  165130                 CLR    TIMR1          ;INIT TIMEOUT COUNTERS
 2723   014310  012767  000036  165124         MOV    #30.,TIMR2
 2724   014316  005777  165070                 TST    @DLRDBR        ;FLUSH "DONE" BIT IN RCVR CSR
 2725   014322  005777  165064                 TST    @DLRDBR
 2726   014326  052777  000100  165054         BIS    #100,@DLRCSR   ;ENABLE RCVR INTR.
 2727   014334  052777  000104  165052         BIS    #104,@DLXCSR   ;ENABLE XMIT INTR. AND MAINT MODE
 2728   014342  000207                         RTS    PC
 2729
 2730
 2731
 2732
 2733                                   ;THIS ROUTINE IS CALLED TO CLEAR THE INPUT AND OUTPUT BUFFERS
```

```
2734
2735   014344   012705   021260          CLDLBF: MOV    #DLBUFO,R5       ;R5 POINTS TO BEGINNING OF BUFFER AREA
2736   014350   005025           1$:    CLR    (R5)+            ;CLEAR A WORD
2737   014352   022705   022260          CMP    #BUFEND,R5       ;DONE ALL WORDS ??
2738   014356   001374                   BNE    1$               ;BR IF NOT
2739   014360   000207                   RTS    PC               ;RETURN TO CALLER
2740
2741                                     ;THIS ROUTINE IS CALLED TO SET UP THE NULL-DEL-NULL PATTERN
2742
2743   014362   012705   021260          LDOUT1: MOV    #DLBUFO,R5       ;R5 POINTS TO OUTPUT BUFFER
2744   014366   105025           1$:    CLRB   (R5)+            ;MOVE A NULL CHAR
2745   014370   112725   000377          MOVB   #377,(R5)+       ;MOV A DEL CHAR
2746   014374   022705   021660          CMP    #DLBUFI,R5       ;ALL DONE ??
2747   014400   001372                   BNE    1$               ;BR IF NOT
2748   014402   000207                   RTS    PC               ;RETURN TO CALLER
2749
2750                                     ;THIS ROUTINE IS USED TO LOAD AN ASCENDING BINARY COUNT PATTERN
2751
2752   014404   005005          LDOUT2: CLR    R5               ;START WITH 000
2753   014406   110565   021260          1$:    MOVB   R5,DLBUFO(R5)    ;LOAD ONE BYTE
2754   014412   005205                   INC    R5               ;INCREMENT BYTE
2755   014414   022705   000400          CMP    #400,R5          ;DONE 000 THRU 377 ??
2756   014420   001372                   BNE    1$               ;BR IF NOT
2757   014422   000207                   RTS    PC               ;RETURN TO CALLER
2758
2759                                     ;THIS ROUTINE IS USED TO LOAD A DESCENDING BINARY COUNT PATTERN
2760
2761   014424   112767   000377  164566  LDOUT3: MOVB   #377,$TMP7       ;START WITH A 377 BYTE
2762   014432   012705   021260          MOV    #DLBUFO,R5       ;R5 POINTS TO OUTPUT BUFFER
2763   014436   116725   164556          1$:    MOVB   $TMP7,(R5)+      ;LOAD ONE BYTE
2764   014442   022705   021660          CMP    #DLBUFI,R5       ;ALL DONE ??
2765   014446   001403                   BEQ    2$               ;BR IF YES
2766   014450   105367   164544          DECB   $TMP7            ;GENERATE NEXT BYTE
2767   014454   000770                   BR     1$               ;GO MOVE IT
2768   014456   000207           2$:    RTS    PC               ;RETURN TO CALLER
2769
2770
2771                                     ;THIS ROUTINE LOADS A COMPLEMENTING WORST CASE PATTERN
2772
2773   014460   012705   021260          LDOUT4: MOV    #DLBUFO,R5       ;R5 POINTS TO OUTPUT BUFFER
2774   014464   005067   164530          CLR    $TMP7            ;INIT. BYTE GENERATOR
2775   014470   116725   164524          1$:    MOVB   $TMP7,(R5)+      ;MOVE A BYTE
2776   014474   105167   164520          COMB   $TMP7            ;COMPLEMENT IT
2777   014500   116725   164514          MOVB   $TMP7,(R5)+      ;NOW LOAD THE 1'S COMPLEMENT
2778   014504   105267   164511          INCB   $TMP7+1          ;INCREMENT THE BYTE
2779   014510   116767   164505  164502  MOVB   $TMP7+1,$TMP7    ;SET UP TO LOAD NEXT TWO
2780   014516   022705   021660          CMP    #DLBUFI,R5       ;ALL DONE ??
2781   014522   001362                   BNE    1$               ;BR IF NOT
2782   014524   000207                   RTS    PC               ;RETURN TO CALLER
2783
2784                                     ;THIS ROUTINE CHECKS FOR DATA COMPARE ERRORS IN 256. BYTE BLOCK TRANSFERS
2785
2786   014526   042777   000104  164660  CHKDAT: BIC    #104,@DLXCSR     ;DISABLE BOTH XMIT AND RCVR INTR. ENAB.
2787   014534   042777   000100  164646  BIC    #100,@DLRCSR
2788   014542   012702   021260          MOV    #DLBUFO,R2       ;R2 POINTS TO S/B DATA IN OUTPUT BUFFER
2789   014546   004767   000070          JSR    PC,MASKING       ;GO TO MASK OFF BITS AS A FUNCTION OF
```

```
2790                                                             ;CHARACTER LENGTH(5, 6, 7, OR 8 BITS)
2791   014552   012701   021660          MOV    #DLBUFI,R1       ;R1 POINTS TO WAS DATA IN RCVR. BUFFER
2792   014556   122221           1$:    CMPB   (R2)+,(R1)+      ;DID S/B = WAS ??
2793   014560   001004                   BNE    3$               ;BR IF NOT
2794   014562   022701   022260          2$:    CMP    #BUFEND,R1       ;CHECKED ALL BYTES ??
2795   014566   001373                   BNE    1$               ;BR IF NOT
2796   014570   000207                   RTS    PC               ;RETURN TO CALLER
2797   014572   013767   177776  164402  3$:    MOV    @#PSW,$TMP0      ;SAVE THE [PSW]
2798   014600   010667   164372          MOV    SP,$REG6         ;SAVE THE [SP]
2799   014604   114204                   MOVB   -(R2),R4         ;GET THE S/B DATA
2800   014606   042704   177400          BIC    #177400,R4       ;CLEAR JUNK FROM HI BYTE
2801   014612   114103                   MOVB   -(R1),R3         ;GET THE WAS DATA
2802   014614   042703   177400          BIC    #177400,R3       ;CLEAR JUNK FROM HI BYTE
2803   014620   004767   177254          JSR    PC,SUERR1        ;GO SET UP ERROR INFO.
2804   014624   012767   014634  164412  MOV    #4$,$ESCAPE      ;RETURN TO 4$ AFTER ERROR PRINT
2805   014632   104003                   ERROR+3                 ;DATA COMPARE ERROR
2806   014634   005202           4$:    INC    R2               ;REPOSITION BUFFER POINTERS
2807   014636   005201                   INC    R1
2808   014640   000750                   BR     2$               ;GO CHECK NEXT BYTE
2809
2810                                     ;THIS ROUTINE IS USED BY THE PATTERN TESTS
2811                                     ;IT WILL MASK OFF THE CHARACTER SENT OUT BY THE XMITTER
2812                                     ;BEFORE THE COMPARISON OF DATA OF WHAT WAS RECEIVED AND WHAT WAS TRANSMITTED
2813                                     ;IS DONE.  THE MASKING IS DONE AS A FUNCTION OF CHARACTER LENGTH WHICH
2814                                     ;CAN BE EITHER 5, 6, 7, OR 8 BITS .
2815
2816   014642   005005          MASKING: CLR    R5               ;INITIALIZE TABLE OFFSET
2817                                                             ;FOR PICKING UP MASK WORD
2818   014644   022767   000010  164362  CMP    #8.,$TMP15       ;IS THE CHARACTER LENGTH 8 BITS?
2819   014652   001427                   BEQ    3$               ;BRANCH IF IT IS
2820   014654   062705   000002          ADD    #2,R5            ;SET UP FOR NEXT MASK WORD
2821                                                             ;IT COULD BE THIS ONE
2822   014660   022767   000007  164346  CMP    #7.,$TMP15       ;IS THE CHARACTER LENGTH 7 BITS?
2823   014666   001410                   BEQ    1$               ;BRANCH IF IT IS
2824   014670   062705   000002          ADD    #2,R5            ;SET UP FOR NEXT MASK WORD
2825                                                             ;IT COULD BE THIS ONE
2826   014674   022767   000006  164332  CMP    #6.,$TMP15       ;IS THE CHARACTER LENGTH 6 BITS?
2827   014702   001402                   BEQ    1$               ;BRANCH IF IT IS
2828   014704   062705   000002          ADD    #2,R5            ;SET UP FOR NEXT MASK WORD
2829                                                             ;IT MUST BE THIS ONE!!!!
2830   014710   016505   014734          1$:    MOV    CHARL(R5),R5     ;PICK UP THE MASK WORD
2831   014714   005105                   COM    R5               ;FORM THE BITS THAT ARE TO BE MASKED
2832   014716   140522           2$:    BICB   R5,(R2)+         ;MASK A BYTE
2833   014720   022702   021660          CMP    #DLBUFI,R2       ;ARE WE AT THE END OF THE XMITTER
2834                                                             ;OUTPUT BUFFER
2835   014724   001374                   BNE    2$               ;BRANCH IF NO TO MASK NEXT BYTE
2836   014726   012702   021260          MOV    #DLBUFO,R2       ;RESTORE R2 BEFORE RETURNING
2837   014732   000207           3$:    RTS    PC               ;RETURN TO MAINLINE CODE
2838                                     ;TABLE OF MASK WORDS
2839   014734   000377          CHARL:  .WORD  377              ;8. BITS IN LENGTH
2840   014736   000177                   .WORD  177              ;7. BITS IN LENGTH
2841   014740   000077                   .WORD  77               ;6. BITS IN LENGTH
2842   014742   000037                   .WORD  37               ;5. BITS IN LENGTH
2843
2844                                     ;THIS ROUTINE IS USED BY PROGRAMS #4 & 5
2845                                     ;IT WILL MASK OFF THE CHARACTER SENT OUT BY THE TRANSMITTER
```

```
                                          ;BEFORE THE COMPARISON OF DATA OF WHAT WAS RECEIVED AND WHAT WAS
2846                                       ;TRANSMITTED IS DONE,  THE MASKING IS DONE AS A FUNCTION OF CHARACTER
2847                                       ;LENGTH WHICH CAN BE EITHER 5, 6, 7, OR 8 BITS.
2848
2849
2850  014744  016767  164234  164260 UPMASK: MOV    $TMP1,$TMP14        ;PICK UP THE CHARACTER THAT WAS
2851                                                                    ;SENT OUT FROM THE XMITTER
2852  014752  005005                          CLR    R5                 ;INITIALIZE TABLE OFFSET
2853                                                                    ;FOR PICKING UP MASK WORD
2854  014754  022767  000010  164252          CMP    #8.,$TMP15         ;IS THE CHARACTER LENGTH 8 BITS?
2855  014762  001423                          BEQ    2$                 ;BRANCH IF IT IS
2856  014764  062705  000002                  ADD    #2,R5.             ;SET UP FOR NEXT MASK WORD
2857                                                                    ;IT COULD BE THIS ONE
2858  014770  022767  000007  164236          CMP    #7.,$TMP15         ;IS THE CHARACTER LENGTH 7 BITS?
2859  014776  001410                          BEQ    1$                 ;BRANCH IF IT IS
2860  015000  062705  000002                  ADD    #2,R5              ;SET UP FOR NEXT MASK WORD
2861                                                                    ;IT COULD BE THIS ONE
2862  015004  022767  000006  164222          CMP    #6.,$TMP15         ;IS THE CHARACTER LENGTH 6 BITS?
2863  015012  001402                          BEQ    1$                 ;BRANCH IF IT IS
2864  015014  062705  000002                  ADD    #2,R5              ;SET UP FOR NEXT MASK WORD
2865                                                                    ;IT MUST BE THIS ONE!!!!
2866  015020  016505  014734       1$:       MOV    CHARL(R5),R5       ;PICK UP THE MASK WORD
2867  015024  005105                          COM    R5                 ;FORM THE BITS THAT ARE TO BE MASKED
2868  015026  140567  164200                  BICB   R5,$TMP14          ;MASK THE LOW BYTE
2869  015032  000207                2$:       RTS    PC                 ;RETURN TO MAINLINE CODE
2870
2871                                       ;ROUTINE TO SERVICE BUS ERROR TRAPS
2872
2873  015034  112767  000060  000632 BUSERR: MOVB   #60,EM4+46         ;SET UP ERROR MESSAGE
2874  015042  112767  000060  000625         MOVB   #60,EM4+47
2875  015050  112767  000064  000620         MOVB   #64,EM4+50
2876  015056  000412                          BR     TRPCOM             ;GO SET UP AND REPORT BUS ERROR
2877
2878                                       ;ROUTINE TO SERVICE RSVD INSTRUCTION TRAPS
2879
2880  015060  112767  000060  000606 RSVERR: MOVB   #60,EM4+46         ;SET UP ERROR MESSAGE
2881  015066  112767  000061  000601         MOVB   #61,EM4+47
2882  015074  112767  000060  000574         MOVB   #60,EM4+50
2883  015102  000400                          BR     TRPCOM             ;GO SET UP AND REPORT RSVD INSTR. ERROR
2884
2885                                       ;ROUTINE TO SET UP AND REPORT BUS ERROR AND RSVD INSTR ERRORS
2886
2887  015104  010667  164066       TRPCOM: MOV    SP,$REG6           ;SAVE THE TRAP SP
2888  015110  116700  163766               MOVB   $TSTNM,R0          ;PUT TEST NO. IN R0
2889  015114  010067  164042               MOV    R0,$REG0           ;SAVE TEST #
2890  015120  016667  000002  164054       MOV    2(SP),$TMP0        ;SAVE THE ERROR PSW
2891  015126  012767  015142  164110       MOV    #1$,$ESCAPE        ;GO TO 1$ AFTER ERROR PRINT
2892  015134  011667  164040               MOV    (SP),$REG7         ;SAVE THE ERROR PC
2893  015140  104004                        ERROR+4                  ;REPORTED TRAP ERROR
2894  015142  000137  001772       1$:     JMP    @#RESTRT           ;ATTEMPT TO RESTART THE PROGRAM
2895                                                                 ;AND TRY AGAIN
2896
2897
2898                                       ;;*************************************************************
2899                                       ;ERROR MESSAGE INFORMATION
2900                                       ;;*************************************************************
2901
```

```
2902                                       ;INFORMATION FOR ERROR MESSAGE 1
2903
2904  015146  046104  030461  051040 EM1:   .ASCIZ  'DL11 REGISTER REFERENCE CAUSED TIMEOUT'
2905  015154  043505  051511  042524
2906  015162  020122  042522  042506
2907  015170  042522  041516  020105
2908  015176  040503  051525  042105
2909  015204  052040  046511  047505
2910  015212  052125     000
2911  015215     040   050050  024503 DH1:   .ASCIZ  ' (PC)    (PS)    (SP)    TEST    DEVADR  REGADR'
2912  015222  020040  020040  050050
2913  015230  024523  020040  020040
2914  015236  051450  024520  020040
2915  015244  020040  042524  052123
2916  015252  020040  042040  053105
2917  015260  042101  020122  051040
2918  015266  043505  042101  000122
2919                                  .EVEN
2920  015274  001116  001202  001176 DT1:   .WORD   $ERRPC,$TMP0,$REG6,$REG0,$REG1,$REG2,0
2921  015302  001162  001164  001166
2922  015310  000000
2923
2924                                       ;INFORMATION FOR ERROR MESSAGE 2
2925
2926  015312  046104  030461  051040 EM2:   .ASCIZ  'DL11 REGISTER ERROR'
2927  015320  043505  051511  042524
2928  015326  020122  051105  047522
2929  015334  000122
2930  015336  024040  041520  020051 DH2:   .ASCIZ  ' (PC)    (PS)    (SP)    TEST    DEVADR  REGADR  WAS     S/B'
2931  015344  020040  024040  051520
2932  015352  020051  020040  024040
2933  015360  050123  020051  020040
2934  015366  052040  051505  020124
2935  015374  020040  042504  040526
2936  015402  051104  020040  042522
2937  015410  040507  051104  020040
2938  015416  053440  051501  020040
2939  015424  020040  051440  041057
2940  015432     000
2941          015434                 .EVEN
2942  015434  001116  001202  001176 DT2:   .WORD   $ERRPC,$TMP0,$REG6,$REG0,$REG1,$REG2,$REG3,$REG4,0
2943  015442  001162  001164  001166
2944  015450  001170  001172  000000
2945
2946                                       ;INFORMATION FOR MESSAGE 3
2947
2948  015456  046104  030461  042040 EM3:   .ASCIZ  'DL11 DATA COMPARE ERROR'
2949  015464  052101  020101  047503
2950  015472  050115  051101  020105
2951  015500  051105  047522  000122
2952  015506  024040  041520  020051 DH3:   .ASCIZ  ' (PC)    (PS)    (SP)    TEST    WASADR  SHBADR  WAS   S/B'
2953  015514  020040  024040  051520
2954  015522  020051  020040  024040
2955  015530  050123  020051  020040
2956  015536  052040  051505  020124
2957  015544  020040  040527  040523
```

```
2958  015552  051104  020040  044123
2959  015560  040502  051104  020040
2960  015566  020040  040527  020123
2961  015574  020040  020040  027523
2962  015602  000102
2963                                          .EVEN
2964  015604  001116  001202  001176    DT3:    .WORD    $ERRPC,$TMP0,$REG6,$REG0,$REG1,$REG2,$REG3,$REG4,0
2965  015612  001162  001164  001166
2966  015620  001170  001172  000000
2967
2968                                          ;INFORMATION FOR MESSAGE 4
2969
2970  015626  047125  054105  042520    EM4:    .ASCIZ  'UNEXPECTED TRAP TO VECTOR AT LOCATION
2971  015634  052103  042105  052040
2972  015642  040522  020120  047524
2973  015650  053040  041505  047524
2974  015656  020122  052101  046040
2975  015664  041517  052101  047511
2976  015672  020116  020040  000040
2977  015700  024040  041520  020051    DH4:    ..ASCIZ  ' (PC)      (PS)      (SP)       TEST'
2978  015706  020040  024040  051520
2979  015714  020051  020040  024040
2980  015722  050123  020051  020040
2981  015730  052040  051505  000124
2982                                          .EVEN
2983  015736  001200  001202  001176    DT4:    .WORD    $REG7,$TMP0,$REG6,$REG0,0
2984  015744  001162  000000
2985
2986                                          ;ERROR INFORMATION FOR ERROR MESSAGE 5
2987
2988  015750  046104  030461  051440    EM5:    .ASCIZ  'DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN)'
2989  015756  043117  020124  051105
2990  015764  047522  020122  050050
2991  015772  051101  052111  026131
2992  016000  051106  046501  047111
2993  016006  026107  047440  020122
2994  016014  053117  051105  052522
2995  016022  024516  000
2996  016025     040  050050  024503    DH5:    .ASCIZ  ' (PC)      (PS)      (SP)       TEST      DEVADR    REGADR      (REG)'
2997  016032  020040  020040  050050
2998  016040  024523  020040  020040
2999  016046  051450  024520  020040
3000  016054  020040  042524  052123
3001  016062  020040  042040  053105
3002  016070  042101  020122  051040
3003  016076  043505  042101  020122
3004  016104  020040  051050  043505
3005  016112  000051
3006                                          .EVEN
3007  016114  001116  001202  001176    DT5:    .WORD    $ERRPC,$TMP0,$REG6,$REG0,$REG1,$REG2,$REG3,0
3008  016122  001162  001164  001166
3009  016130  001170  000000
3010
3011                                          ;INFORMATION FOR ERROR MESSAGE 6
3012
3013  016134  024040  041520  020051    DH6:    .ASCIZ  ' (PC)      (PS)      (SP)       REGADR'
```

```
3014  016142  020040  024040  051520
3015  016150  020051  020040  024040
3016  016156  050123  020051  020040
3017  016164  042522  040507  051104
3018  016172     000
3019          016174                          .EVEN
3020  016174  001116  001204  001176    DT6:    .WORD    $ERRPC,$TMP1,$REG6,$REG2,0
3021  016202  001166  000000
3022
3023                                          ;INFORMATION FOR ERROR MESSAGE 7
3024
3025  016206  024040  041520  020051    DH7:    .ASCIZ  ' (PC)      DEVADR    REGADR      (REG)'
3026  016214  020040  042504  040526
3027  016222  051104  020040  042522
3028  016230  040507  051104  020040
3029  016236  024040  042522  024507
3030  016244     000
3031          016246                          .EVEN
3032  016246  001116  001164  001166    DT7:    .WORD    $ERRPC,$REG1,$REG2,$REG3,0
3033  016254  001170  000000
3034
3035                                          ;INFORMATION FOR ERROR MESSAGE 10
3036
3037  016260  024040  041520  020051    DH10:   .ASCIZ  ' (PC)      DEVADR    REGADR      (REG)      S/B'
3038  016266  020040  042504  040526
3039  016274  051104  020040  042522
3040  016302  040507  051104  020040
3041  016310  024040  042522  024507
3042  016316  020040  020040  027523
3043  016324  000102
3044                                          .EVEN
3045  016326  001116  001164  001166    DT10:   .WORD    $ERRPC,$REG1,$REG2,$REG3,$REG4,0
3046  016334  001170  001172  000000
3047                                          ;MISCELLANEOUS MESSAGES
3048
3049  016342  052516  046114  042055    XMSG1:  .ASCIZ  'NULL-DEL-NULL SEQUENCE TIMEOUT AT FOLLOWING PC'
3050  016350  046105  047055  046125
3051  016356  020114  042523  052521
3052  016364  047105  042503  052040
3053  016372  046511  047505  052125
3054  016400  040440  020124  047506
3055  016406  046114  053517  047111
3056  016414  020107  041520     000
3057  016421     102  047111  051101    XMSG2:  .ASCIZ  'BINARY UP COUNT SEQUENCE TIMEOUT AT FOLLOWING PC'
3058  016426  020131  050125  041440
3059  016434  052517  052116  051440
3060  016442  050505  042525  041516
3061  016450  020105  044524  042515
3062  016456  052517  020124  052101
3063  016464  043040  046117  047514
3064  016472  044527  043516  050040
3065  016500  000103
3066  016502  044502  040516  054522    XMSG3:  .ASCIZ  'BINARY DOWN COUNT SEQUENCE TIMEOUT AT FOLLOWING PC'
3067  016510  042040  053517  020116
3068  016516  047503  047125  020124
3069  016524  042523  052521  047105
```

```
3070  016532  042503  052040  046511
3071  016540  047505  052125  040440
3072  016546  020124  047506  046114
3073  016554  053517  047111  020107
3074  016562  041520     000
3075  016565     127  051117  052123  XMSG4:  .ASCIZ  'WORST CASE PATTERN SEQUENCE TIMEOUT AT FOLLOWING PC'
3076  016572  041440  051501  020105
3077  016600  040520  052124  051105
3078  016606  020116  042523  052521
3079  016614  047105  042503  052040
3080  016622  046511  047505  052125
3081  016630  040440  020124  047506
3082  016636  046114  053517  047111
3083  016644  020107  041520     000
3084
3085  016651     015  046412  044501  STMES:  .ASCIZ  <15><12>'MAINDEC-11-DZDLC-B'<15><12>
3086  016656  042116  041505  030455
3087  016664  026461  055104  046104
3088  016672  026503  006502  000012
3089
3090  016700  005015  047531  020125  PROG2M: .ASCIZ  <15><12>'YOU HAVE SELECTED PROGRAM NO. 2'<15><12>
3091  016706  040510  042526  051440
3092  016714  046105  041505  042524
3093  016722  020104  051120  043517
3094  016730  040522  020115  047516
3095  016736  020056  006462  000012
3096  016744  005015  047531  020125  PROG3M: .ASCIZ  <15><12>'YOU HAVE SELECTED PROGRAM NO. 3'<15><12>
3097  016752  040510  042526  051440
3098  016760  046105  041505  042524
3099  016766  020104  051120  043517
3100  016774  040522  020115  047516
3101  017002  020056  006463  000012
3102  017010  005015  047531  020125  PROG4M: .ASCIZ  <15><12>'YOU HAVE SELECTED PROGRAM NO. 4'<15><12>
3103  017016  040510  042526  051440
3104  017024  046105  041505  042524
3105  017032  020104  051120  043517
3106  017040  040522  020115  047516
3107  017046  020056  006464  000012
3108  017054  005015  047531  020125  PROG5M: .ASCIZ  <15><12>'YOU HAVE SELECTED PROGRAM NO. 5'<15><12>
3109  017062  040510  042526  051440
3110  017070  046105  041505  042524
3111  017076  020104  051120  043517
3112  017104  040522  020115  047516
3113  017112  020056  006465  000012
3114  017120  005015  051124  047101  XDB:    .ASCIZ  <15><12>'TRANSMITTER DONE BIT NEVER SET  PC= '
3115  017126  046523  052111  042524
3116  017134  020122  047504  042516
3117  017142  041040  052111  047040
3118  017150  053105  051105  051440
3119  017156  052105  020040  041520
3120  017164  020075     000
3121  017167     015  051012  041505  RDB:    .ASCIZ  <15><12>'RECEIVER DONE BIT NEVER SET  PC=
3122  017174  044505  042526  020122
3123  017202  047504  042516  041040
3124  017210  052111  047040  053105
3125  017216  051105  051440  052105
```

```
3126  017224  020040  041520  020075
3127  017232     000
3128                                  ;MESSAGES SEEKING USER RESPONSE
3129
3130  017233     015  053412  040510  LENGTH: .ASCIZ  <15><12>'WHAT IS THE CHARACTER LENGTH (5,6,7 OR 8 BITS)?'
3131  017240  020124  051511  052040
3132  017246  042510  041440  040510
3133  017254  040522  052103  051105
3134  017262  046040  047105  052107
3135  017270  020110  032450  033054
3136  017276  033454  047440  020122
3137  017304  020070  044502  051524
3138  017312  037451     000
3139  017315     015  042012  020117  DEFAULT:        .ASCII  <15><12>'DO YOU WISH TO TEST OTHER THAN THE'
3140  017322  047531  020125  044527
3141  017330  044123  052040  020117
3142  017336  042524  052123  047440
3143  017344  044124  051105  052040
3144  017352  040510  020116  044124
3145  017360     105
3146  017361     015  042012  043105          .ASCIZ  <15><12>'DEFAULT DEVICE (1/0 = YES/NO)?'
3147  017366  052501  052114  042040
3148  017374  053105  041511  020105
3149  017402  030450  030057  036440
3150  017410  054440  051505  047057
3151  017416  024517  000077
3152  017422  005015  044127  052101  MFIRSTD:.ASCIZ  <15><12>'WHAT IS THE 1ST RECEIVER STATUS REGISTER ADDRESS?
3153  017430  044440  020123  044124
3154  017436  020105  051461  020124
3155  017444  042522  042503  053111
3156  017452  051105  051440  040524
3157  017460  052524  020123  042522
3158  017466  044507  052123  051105
3159  017474  040440  042104  042522
3160  017502  051523  020077  000040
3161  017510  005015  044127  052101  MVECT:  .ASCIZ  <15><12>'WHAT IS THE 1ST RECEIVER''S VECTOR ADDRESS?
3162  017516  044440  020123  044124
3163  017524  020105  051461  020124
3164  017532  042522  042503  053111
3165  017540  051105  020123  042526
3166  017546  052103  051117  040440
3167  017554  042104  042522  051523
3168  017562  020077  000040
3169  017566  005015  047504  054440  MULDEV: .ASCIZ  <15><12>'DO YOU WANT TO TEST MULTIPLE DEVICES 1/0=YES/NO?
3170  017574  052517  053440  047101
3171  017602  020124  047524  052040
3172  017610  051505  020124  052515
3173  017616  052114  050111  042514
3174  017624  042040  053105  041511
3175  017632  051505  030440  030057
3176  017640  054475  051505  047057
3177  017646  037517  020040     000
3178  017653     015  053412  040510  MLASTD: .ASCIZ  <15><12>'WHAT IS THE STATUS REGISTER ADDRESS OF THE LAST RECEIVER?
3179  017660  020124  051511  052040
3180  017666  042510  051440  040524
3181  017674  052524  020123  042522
```

```
3182  017702  044507  052123  051105
3183  017710  040440  042104  042522
3184  017716  051523  047440  020106
3185  017724  044124  020105  040514
3186  017732  052123  051040  041505
3187  017740  044505  042526  037522
3188  017746  020040    000
3189  017751    015   051412  046517  MRANGE: .ASCIZ  <15><12>'SOMETHING WRONG-ANSWER THE LAST QUESTION AGAIN!
3190  017756  052105  044510  043516
3191  017764  053440  047522  043516
3192  017772  040455  051516  042527
3193  020000  020122  044124  020105
3194  020006  040514  052123  050440
3195  020014  042525  052123  047511
3196  020022  020116  043501  044501
3197  020030  020516  020040    000
3198  020035    015   053412  040510  PLEVEL: .ASCIZ  <15><12>'WHAT IS YOUR INTERRUPT PRIORITY LEVEL?
3199  020042  020124  051511  054440
3200  020050  052517  020122  047111
3201  020056  042524  051122  050125
3202  020064  020124  051120  047511
3203  020072  044522  054524  046040
3204  020100  053105  046105  020077
3205  020106  000040
3206  020110  005015  051120  043517  FOULUP: .ASCII  <15><12>'PROGRAM DEVICE ACTIVE LOCATION SHOWS NO DEVICE ACTIVE'
3207  020116  040522  020115  042504
3208  020124  044526  042503  040440
3209  020132  052103  053111  020105
3210  020140  047514  040503  044524
3211  020146  047117  051440  047510
3212  020154  051527  047040  020117
3213  020162  042504  044526  042503
3214  020170  040440  052103  053111
3215  020176    105
3216  020177    015   051412  052105         .ASCII  <15><12>'SET SWITCH 0 TO A ONE (1) AND'
3217  020204  051440  044527  041524
3218  020212  020110  020060  047524
3219  020220  040440  047440  042516
3220  020226  024040  024461  040440
3221  020234  042116
3222  020236  005015  044510  020124         .ASCII  <15><12>'HIT CONTINUE TO GO BACK TO DEVICE SELECTION AGAIN'
3223  020244  047503  052116  047111
3224  020252  042525  052040  020117
3225  020260  047507  041040  041501
3226  020266  020113  047524  042040
3227  020274  053105  041511  020105
3228  020302  042523  042514  052103
3229  020310  047511  020116  043501
3230  020316  044501  000116
3231  020322  005015  044127  052101  LINTAD: .ASCIZ  <15><12>'WHAT IS THE TRANSMITTER DATA BUFFER ADDRESS?
3232  020330  044440  020123  044124
3233  020336  020105  051124  047101
3234  020344  046523  052111  042524
3235  020352  020122  040504  040524
3236  020360  041040  043125  042506
3237  020366  020122  042101  051104
```

```
3238  020374  051505  037523  020040
3239  020402    000
3240  020403    015   053412  040510  SELCAR: .ASCII  <15><12>'WHAT IS THE CHARACTER TO BE TRANSMITTED (OCTAL ASCII E.G. A=101
3241  020410  020124  051511  052040
3242  020416  042510  041440  040510
3243  020424  040522  052103  051105
3244  020432  052040  020117  042502
3245  020440  052040  040522  051516
3246  020446  044515  052124  042105
3247  020454  024040  041517  040524
3248  020462  020114  051501  044503
3249  020470  020111  027105  027107
3250  020476  040440  030475  030460
3251  020504  037451  020040    000
3252  020511    015   053412  040510  SELDLY: .ASCII  <15><12>'WHAT IS THE DESIRED MSEC. DELAY (OCTAL E.G. 10=8(10))?
3253  020516  020124  051511  052040
3254  020524  042510  042040  051505
3255  020532  051111  042105  046440
3256  020540  042523  027103  042040
3257  020546  046105  054501  024040
3258  020554  041517  040524  020114
3259  020562  027105  027107  030440
3260  020570  036460  024070  030061
3261  020576  024451  020077  000040
3262  020604  005015  051511  040440  RSTALL: .ASCIZ  <15><12>'IS A RANDOM WAIT TIME (MSEC.) DESIRED  1/0=YES/NO?
3263  020612  051040  047101  047504
3264  020620  020115  040527  052111
3265  020626  052040  046511  020105
3266  020634  046450  042523  027103
3267  020642  020051  042504  044523
3268  020650  042522  020104  030440
3269  020656  030057  054475  051505
3270  020664  047057  037517  020040
3271  020672    000
3272  020673    015   054412  052517  FAILSA: .ASCII  <15><12>'YOU HAVE SWR8 SET INDICATING LOOP ON TEST'
3273  020700  044040  053101  020105
3274  020706  053523  034122  051440
3275  020714  052105  044440  042116
3276  020722  041511  052101  047111
3277  020730  020107  047514  050117
3278  020736  047440  020116  042524
3279  020744  052123
3280  020746  005015  040510  042526         .ASCII  <15><12>'HAVE YOU MODIFIED THE PROPER LOCATIONS FOR THE'
3281  020754  054440  052517  046440
3282  020762  042117  043111  042511
3283  020770  020104  044124  020105
3284  020776  051120  050117  051105
3285  021004  046040  041517  052101
3286  021012  047511  051516  043040
3287  021020  051117  052040  042510
3288  021026  005015  042504  044526         .ASCII  <15><12>'DEVICE THAT YOU WANT TO TEST?'
3289  021034  042503  052040  040510
3290  021042  020124  047531  020125
3291  021050  040527  052116  052040
3292  021056  020117  042524  052123
3293  021064    077
```

```
3294  021065      015  044412  020106          .ASCII  <15><12>'IF SO - PRESS THE CONTINUE SWITCH'
3295  021072  047523  026440  050040
3296  021100  042522  051523  052040
3297  021106  042510  041440  047117
3298  021114  044524  052516  020105
3299  021122  053523  052111  044103
3300  021130  005015  043111  047040          .ASCII  <15><12>'IF NOT - MODIFY THE PROPER LOCATIONS, THEN'
3301  021136  052117  026440  046440
3302  021144  042117  043111  020131
3303  021152  044124  020105  051120
3304  021160  050117  051105  046040
3305  021166  041517  052101  047511
3306  021174  051516  020054  044124
3307  021202  047105
3308  021204  005015  042522  052123          .ASCIZ  <15><12>'RESTART THE PROGRAM AT ADDRESS 200'
3309  021212  051101  020124  044124
3310  021220  020105  051120  043517
3311  021226  040522  020115  052101
3312  021234  040440  042104  042522
3313  021242  051523  031040  030060
3314  021250     000
3315
3316  021251      040  020075  041520          PCMSG:  .ASCII  ' = PC'
3317  021256  000040                                   .ASCIZ  ' '
3318
3319                                          .EVEN
3320                                          ;512. WORDS RESERVED FOR TWO 256. BYTE INPUT/OUTPUT DATA BUFFERS
3321
3322  021260  000400          DLBUFO: .BLKB   256.           ;RSVD FOR OUTPUT BUFFER
3323                                                         ;THIS IS THE DATA BEING SENT OUT
3324                                                         ;BY THE TRANSMITTER
3325  021660  000400          DLBUFI: .BLKB   256.           ;RSVD FOR INPUT BUFFER
3326                                                         ;THIS IS THE DATA THAT WAS PICKED
3327                                                         ;UP BY THE RECEIVER (I.E. DATA
3328                                                         ;SENT BY THE TRANSMITTER - HOPEFULLY)
3329  022260  000000          BUFEND: 0                      ;TAG MARKS END OF BUFFERS
3330
3331          000001          .END
```

```
ACTREG  001274       266#   621*   653*   654*   664*   1776   1800
BASEAD  001264       252#   573*   659*   660    667*   675*   1788*  1804   1823*  1825
BASFIV  001270       259#   593*   1791*  1806   1824*  1826
BEGIN   001446        46    393#
BIT0  = 000001       152#   560    583    641    1092   1098   1111   1372   1455   1540   1649
BIT00 = 000001       142#   152
BIT01 = 000002       141#   151
BIT02 = 000004       140#   150
BIT03 = 000010       139#   149
BIT04 = 000020       138#   148
BIT05 = 000040       137#   147
BIT06 = 000100       136#   146
BIT07 = 000200       135#   145
BIT08 = 000400       134#   144    1888
BIT09 = 001000       133#   143    1896   1955
BIT1  = 000002       151#   1003   1017   1062
BIT10 = 002000       132#   1940
BIT11 = 004000       131#   1903
BIT12 = 010000       130#
BIT13 = 020000       129#   1947
BIT14 = 040000       128#   1874
BIT15 = 100000       127#   937    949    968    980    982    1004   1016   1018   1076
BIT2  = 000004       150#   812    818    882    893    906    936    948    1501   1603   1712
BIT3  = 000010       149#   967    981
BIT4  = 000020       148#
BIT5  = 000040       147#   1037   1043   1061
BIT6  = 000100       146#   858    864    881    908
BIT7  = 000200       145#
BIT8  = 000400       144#
BIT9  = 001000       143#
BPTVEC= 000014       159#
BUFEND  022260       1133   1190   1247   1304   2573   2737   2794   3329#
BUSERR  015034       486    2873#
CHARL   014734       2830   2839#  2866
CHKDAT  014526       1135   1192   1249   1306   2786#
CLDLBF  014344       2720   2735#
CONQUE  002626       626    670    682#   723
CR    = 000015        67#   2375   2385
CRLF  = 000200        68#   2346   2385
DATCHK  013752       1617   1725   2650#
DDISP = 177570        74#   196    420
DEFAUL  017315       531    3139#
DELAY   013540       1420   1506   2582#  2619
DELCNT  013602       2582*  2585   2592*  2597#
DH1     015215       317    2911#
DH10    016260       366    3037#
DH2     015336       324    2930#
DH3     015506       331    2952#
DH4     015700       338    2977#
DH5     016025       345    2996#
DH6     016134       352    3013#
DH7     016206       359    3025#
DISPLA  001142       196#   420*   428*   1917*  1939*
DISPRE  000174        43#   428
DLADDR  010070       480    571    1743#  1808   1828
DLBASE  001260       246#   479*   567*   570    1743   1745*  1747   1749*  1751   1753*  1755   1804*  1825*
```

```
DLBUFI  021660      2548    2719    2746    2764    2780    2791    2833    3325#
DLBUFO  021260      2718    2735    2743    2753*   2762    2773    2788    2836    3322#
DLPRI   001302      282#    440*    696*    702*    703*    704*    705*    706*    707     830     875     1056    2706
                    2708
DLRCSR  001410      374#    508*    714*    732     791     857     879     934     937     949     964     968     982
                    1000    1004    1018    1035    1060    1141*   1148    1157    1166    1198*   1205    1214    1273
                    1255*   1262    1271    1280    1312*   1318    1326    1334    1743*   1813*   2545*   2558    2571*
                    2679    2693    2714*   2726*   2787*
DLRDBR  001412      375#    509     510     715     716     762     916     1167    1224    1281    1335    1747*   1814
                    1815    2564    2568    2575    2724    2725
DLVECT  001420      378#    482*    590*    592     593     828     873     1054    1806*   1826*   2704
DLXCSR  001414      376#    507*    747     801     811     833     1091    1110    1142*   1149    1199*   1206
                    1256*   1263    1313*   1319    1751*   1812*   2540    2546*   2570*   2713*   2727*   2786*
DLXDBR  001416      377#    777     917     1755*   2550*
DSWR =  177570      73#     195     419
DT1     015274      318     2920#
DT10    016326      367     3045#
DT2     015434      325     2942#
DT3     015604      332     2964#
DT4     015736      339     2983#
DT5     016114      346     3007#
DT6     016174      353     3020#
DT7     016246      360     3032#
DUT     013750      2627*   2629#   2633*   2640    2645#
EMTVEC= 000030      162#    404*    405*    1357*   1358*   1440*   1441*   1525*   1526*   1634*   1635*
EM1     015146      316     351     2904#
EM2     015312      323     2926#
EM3     015456      330     365     2948#
EM4     015626      337     2873*   2874*   2880*   2881*   2882*   2970#
EM5     015750      344     358     2988#
ERRVEC= 000004      155#    417     418*    429*    486*    487*    730     731*    739*    745     746*    754*    760
                    761*    769*    775     776*    784*    1386    1387*   1396*   1398*   1468    1469*   1478*   1480*
                    1553    1554*   1563*   1565*   1662    1663*   1672*   1674*   1879    1880*   1882*   1885*
FAILSA  020673      463     3272#
FIRSTD  002200      549#    601
FLUSH   002724      536     713#
FOULUP  020110      1778    3206#
GNS  =  ****** U    42      2478    2479    2480    2481    2482    2485    2486    2487    2488
GO      002100      506     520#    598
HOWMAN  002376      594     605#
HT   =  000011      65#     2344    2385
INTFLG  001444      388#    827*    835     845*    876*    888     909*    1058*   1063    1074*
IOTVEC= 000020      160#    402*    403*
IPTR    001434      384#    1133    1190    1247    1304    2573    2575*   2576*   2719*
KEEPAD  001262      248#    570*    573     667     675     1823
KEEPIV  001266      255#    592*    1824
LASTAD  001300      278#    648*    660
LASTD   002446      620     627#    720
LDOUT   001436      385#    1125*   1182*   1239*   1296*   2721
LDOUT1  014362      1125    2743#
LDOUT2  014404      1182    2752#
LDOUT3  014424      1239    2761#
LDOUT4  014460      1296    2773#
LENGTH  017233      520     1583    1678    3130#
LESS1   001304      284#    707*    709*    711*
LF   =  000012      66#     2379    2385
```

```
LINTAD  020322      1361    1444    1529    1638    3231#
MASKIN  014642      2789    2816#
MFIRST  017422      549     3152#
MLASTD  017653      630     3178#
MRANGE  017751      677     3189#
MULDEV  017566      605     3169#
MULTD   001272      262#    431*    612*    615*    618     1773    '1783*
MVECT   017510      574     3161#
NUMONE  013672      2602    2606*   2613    2622#
NUMTWO  013674      2605    2609    2612*   2623#
ONCE    002054      467     496     507#
OPTR    001432      383#    2548    2550    2551*   2718*
PCMSG   021251      465     3316#
PIRQ =  177772      72#
PIRQVE= 000240      166#
PLEVEL  020035      688     3198#
PRG2    006344      47      1354#
PRG2A   006404      1361#   1426
PRG2B   006524      1403#
PRG3    006604      48      1437#
PRG3A   006644      1444*   1512
PRG3B   006764      1485#
PRG4    007054      49      1522#
PRG4A   007114      1529#   1621
PRG4B   007246      1575#
PRG4C   007260      1583#   1624
PRG5    007446      50      1631#
PRG5A   007506      1638#   1730
PRG5B   007676      1697#   1727
PRG5C   007626      1678#   1733
PRIME   014234      1126    1183    1240    1297    2713#
PROG2M  016700      1359    3090#
PROG3M  016744      1442    3096#
PROG4M  017010      1527    3102#
PROG5M  017054      1636    3108#
PR0  =  000000      89#
PR1  =  000040      90#
PR2  =  000100      91#
PR3  =  000140      92#
PR4  =  000200      93#
PR5  =  000240      94#
PR6  =  000300      95#
PR7  =  000340      96#
PS   =  177776      69#     70
PSW  =  177776      70#     892     2542    2560    2566    2678    2691    2696    2797
PWRVEC= 000024      161#    408*    409*    2494*   2495*   2504*   2510*   2522*   2523*
RDB     017167      1609    1717    3121#
RDCHR = 104406      2207    2485#
RDDEC = 104411      522     1585    1680    2488#
RDLIN = 104407      2280    2405    2486#
RDOCT = 104410      533     552     575     607     633     679     689     1364    1405    1413    1447    1487    1495
                    1532    1571    1577    1641    1690    2487#
REDO1   006574      1369    1371    1374    1381    1397    1425#
REDO2   007044      1452    1454    1457    1464    1479    1511#
REDO3   007426      1537    1539    1542    1549    1564    1620#
REDO3A  007436      1589    1591    1622#
```

```
REDO4    010050        1646    1648    1651    1658    1673    1729#
REDO4A   010060        1684    1686    1731#
RESTRT   001772         485    1785    1854    2527    2894
RESVEC=  000010         156     488*    489*
RETRY    002346         526     528     596#
RETRYO   002356         557     559     562     566     599#
RETRY1   002366         580     582     585     589     602#
RETRY2   002750         638     640     643     647     718#
RETRY3   002760         693     695     721#
RFLG0    001424         380#   1129    1186    1243    1300    2562*   2716*
RFLG1    001426         381#   1131    1188    1245    1302    2569*   2717*
RINT     013416        2558#   2705
ROTADD   001276         273#    624*    652*    654     657*    664     665*   1795*   1800    1821*
RSTALL   020604        1567    1689    3262#
RSVERR   015060         488    2880#
RTRY     001430         382#   1124#   1172#   1173    1181*   1229*   1230    1238*   1286*   1287    1295*   1340*   1341
SELCAR   020403        1403    1485    1575    3240#
SELDLY   020511        1410    1492    3252#
STACK =  001100          60#    400     485    1354    1437    1522    1631
STALL    013604        1598    1707    2602#
STKLMT=  177774          71#
STLMSK   001306         290#   2614
STMES    016651         462     498    3085#
SUERR1   014100         899     925    1152    1161    1169    1209    1218    1226    1266    1275    1283    1322    1330
                       1337    2683#   2803
SUEPT1   014132         735     750     765     780    2691#
SUEPT2   014160        1392    1474    1559    1668    2696#
SUER2    014052         794     804     815     821     840     850     861     867     914     939     945     951     955
                        970     976     984     991    1006    1013    1020    1026    1040    1046    1067    1078    1082
                       1095    1101    1115    2678#
SUVEC    014206        1123    1180    1237    1294    2704#
SWR      001140         195#    398     419*    421     427*    442     502     931     962     998    1033    1052    1088
                       1874    1888    1890    1896    1903    1940    1947    1952    1955    2502    2515*
SWREG    000176          44#    427
SW0   =  000001         124#    502
SW00  =  000001         114#    124
SW01  =  000002         113#    123
SW02  =  000004         112#    122
SW03  =  000010         111#    121
SW04  =  000020         110#    120
SW05  =  000040         109#    119
SW06  =  000100         108#    118
SW07  =  000200         107#    117
SW08  =  000400         106#    116
SW09  =  001000         105#    115
SW1   =  000002         123#
SW10  =  002000         104#
SW11  =  004000         103#
SW12  =  010000         102#    931     962     998    1033    1052    1088
SW13  =  020000         101#
SW14  =  040000         100#
SW15  =  100000          99#
SW2   =  000004         122#
SW3   =  000010         121#
SW4   =  000020         120#
SW5   =  000040         119#
```

```
SW6   =  000100         118#
SW7   =  000200         117#
SW8   =  000400         116#    442
SW9   =  001000         115#
TABFLG   001256         242#    433*    483*    495     500*   1784*
TBITVE=  000014         157#
TCONT    013722        2631    2635#
TIMERX   013676        1607    1716    2627#
TIMETX   013714        1599    1708    2633#
TIMR1    001440         386#   1137*   1194*   1251*   1308*   2722*
TIMR2    001442         387#   1139*   1196*   1253*   1310*   2723*
TKVEC =  000060         164#
TPVEC =  000064         165#
TRAPVE=  000034         163#    406*    407*   1355*   1356*   1438*   1439*   1523*   1524*   1632*   1633*
TRPCOM   015104        2876    2883    2887#
TRTVEC=  000014         158#
TST1     002770         511     717     729#   1816
TST10    003340         820     826#
TST11    003464         844     849     855#
TST12    003534         866     872#
TST13    004016         905     924     930#
TST14    004146         932     954     961#
TST15    004274         963     990     997#
TST16    004422         999    1025    1032#
TST17    004502        1034    1045    1051#
TST2     003036         744#
TST20    004644        1053    1072    1081    1087#
TST21    004726        1089    1100    1108#
TST22    004760        1114    1121#
TST23    005252        1136    1147    1156    1165    1178#
TST24    005544        1193    1204    1213    1222    1235#
TST25    006036        1250    1261    1270    1279    1292#
TST3     003104         759#
TST4     003152         774#
TST5     003220         789#
TST6     003242         793     799#
TST7     003266         803     809#
TYPOS    104405        1843    2482#
TYPE  =  104401         462     463     465     498     520     531     549     574     596     599     602     605     630
                        677     688     718     721    1143    1200    1257    1314    1359    1361    1403    1410    1425
                       1442    1444    1485    1492    1511    1527    1529    1567    1575    1583    1601    1609    1620
                       1622    1636    1638    1678    1689    1710    1717    1729    1731    1778    1841    1844    1942
                       1950    1972    1989    1991    1994    1996    2000    2007    2072    2148    2214    2220    2225
                       1980    2004    2479#
TYPOC =  104402        1980    2004    2479#
TYPON =  104404        2481#
TYPOS =  104403        2480#
UPMASK   014744         884     918    2662    2850#
VECT     002264         574#    604
XDB      017120        1601    1710    3114#
XFLG0    001422         379#   1127    1184    1241    1298    2544*   2715*
XINT     013330        2540#   2707
XMSG1    016342        1143    3049#
XMSG2    016421        1200    3057#
XMSG3    016502        1257    3066#
XMSG4    016565        1314    3075#
```

```
$AUTOB  001134          192#
$BDADR  001122          187#
$BDDAT  001126          189#
$BELL   001246          233#   1942   1963
$CHARC  012704          2351*  2361*  2368   2377*  2382#
$CKSWR= ****** U        2485
$CMTAG  001100          175#   395    396    404    410    411
$CM1  = 000010          207#   208#   209#   210#   211#   212#   213#   214#   215#
$CM2  = 000020          207#   208#   209#   210#   211#   212#   213#   214#   215#
$CM3  = 000010          205#   207
$CM4  = 000020          215#   216#   217#   218#   219#   220#   221#   222#   223#   224#   225#   226#   227#
                        228#   229#   230#   231#
$CNTLG  012301          2255#
$CNTLU  012274          2229   2254#
$CRLF   001253          235#   1950   1963   1972   1991   1996   2000   2234   2254   2314   2350  ·2385   2446
$DBLK   011700          2114   2148   2156#
$DOAGN  010452          1837   1846   1852#
$DTBL   011670          2117   2152#
$ENDAD  010442          1848#
$ENDCT  010410          1839#
$ENDMG  010461          1841   1856#
$ENULL  010456          1844   1855#
$EOP    010144          1307   1317   1325   1333   1343   1767#
$EOPCT  010402          1836#  1840
$ERFLG  001103          178#   1864   1892   1894   1900*  1921   1937#  1963
$ERMAX  001115          184#   412*   1894   1916*  1921
$ERROR  010746          404    1357   1440   1525   1634   1936#
$ERRPC  001116          185#   1944#  1945*  1946   1963   1978   2920   2942   2964   3007   3020   3032   3045
$ERRTB  001310          312#   1986
$ERRTY  011102          1949   1971#
$ERTTL  001112          182#   1943#  1963
$ESCAP  001244          232#   411*   736*   751*   766*   781*   841*   900*   1069*  1144*  1153*  1162*  1170*
                        1201*  1210*  1219*  1227*  1258*  1267*  1276*  1284*  1315*  1323*  1331*  1338*  1393*
                        1475*  1560*  1669*  1915*  1958   1960   1963   2804*  2891*
$FILLC  001156          203#   2354   2385
$FILLS  001155          202#   2385
$GDADR  001120          186#
$GDDAT  001124          188#
$GET42  010432          1845#
$GTSWR= ****** U        2484
$HD   = 000001          16     17
$HIOCT  012466          2302*  2313#
$ICNT   001104          179#   1907*  1908   1910*  1920
$ILLUP  013312          2494   2510   2529#
$INTAG  001135          193#
$ITEMB  001114          183#   1946*  1963   1975
$LF     001254          236#   1963   2244   2254   2314   2385   2446
$LPADR  001106          180#   413*   1898*  1913#  1918   1920
$LPERR  001110          181#   414*   1898   1914*  1920   1957
$MAIL = ****** U        431    1913   1952   2338
$MNEW   012317          2258#
$MSWR   012306          2256#
$MXCNT  010744          1911   1920#
$NULL   001154          201#   2356   2385
$NWTST= 000001          726#   741#   756#   771#   786#   796#   806#   823#   852#   869#   927#   958#   994#
                        1029#  1048#  1084#  1105#  1118#  1175#  1232#  1289#
```

```
$OCNT   011460          2044*  2073*  2086#
$OMODE  011462          2039*  2043*  2048   2051*  2062*  2088#
$OVER   010730          1875   1891   1899   1909   1917#
$PASS   001100          176#   1833*  1834*  1842   1855   1905   1921
$POWER  013320          2525   2532#
$PWRAD  013306          2527#
$PWRDN  013146          408    2494#  2522
$PWRMG  013302          2525#
$PWRUP  013220          2504   2510#
$QUES   001252          234#   596    599    602    718    721    1425   1511   1620   1622   1729   1731   1963
                        2237   2254   2311   2314   2385   2444   2446
$RDCHR  011710          2174#  2485
$RDDEC  012710          2400#  2488
$RDLIN  012030          2202#  2486
$RDOCT  012330          2275#  2487
$RDSZ = 000010          2195#
$REGAD  001160          205#
$REG0   001162          207#   2684*  2694*  2889*  2920   2942   2964   2983   3007
$REG1   001164          208#   2654*  2656*  2685*  2695*  2920   2942   2964   3007   3032   3045
$REG2   001166          209#   2652*  2654   2686*  2699*  2920   2942   2964   3007   3020   3032   3045
$REG3   001170          210#   2658*  2687*  2942   2964   3007   3032   3045
$REG4   001172          211#   2667*  2688*  2942   2964   3045
$REG5   001174          212#
$REG6   001176          213#   895*   2543*  2561*  2567*  2681*  2682*  2697*  2698*  2798*  2887*  2920   2942
                        2964   2983   3007   3020
$REG7   001200          214#   956*   2892*  2983
$RTNAD  010454          1854#
$R2A  = ****** U        2489
$SAVRE= ****** U        2489
$SAVR6  013316          2503*  2511   2512*  2513*  2531#
$SCOPE  010476          402    1873#
$SETUP= 000017          392#   401    402    404    406    408    410    411    413    1831   1874   1937   1955
                        1962   2163   2260
$STUP = 177777          392#
$SVLAD  010702          1883   1912#
$SWR  = 167400          3#     16     24     25     26     27     28     29     30     31     231    232    233
                        410    411    413    414    730    745    760    775    790    800    810    827    856
                        873    931    962    998    1033   1052   1088   1109   1122   1179   1236   1293   1764
                        1832   1847   1853   1855   1865   1866   1867   1868   1869   1874   1886   1888   1889
                        1892   1893   1894   1901   1902   1903   1914   1917   1920   1928   1929   1930   1931
                        1932   1940   1947   1952   '1955   1963   2528
$SWRMK= 000000          31     32     1869   1870   1890
$TIMES  001242          231#   410*   1122#  1179*  1236*  1293*  1832*  1901*  1908   1911*  1920
$TKB    001146          198#   2161   2178   2184
$TKS    001144          197#   2161   2176   2182
$TMP0   001202          215#   892*   1382*  1418*  1465*  1504*  1550*  1594   1605*  1659*  1703   1714*  2542*
                        2560#  2566*  2678*  2691*  2797*  2890*  2920   2942   2964   2983   3007
$TMP1   001204          216#   878*   883*   907*   1168   1225   1282   1336   1407*  1418   1489*  1504   1579*
                        1605   1701*  1714   2568*  2696*  2850   3020
$TMP10  001222          223#
$TMP11  001224          224#   1696*  1697   1700*
$TMP12  001226          225#   1697*  1698*  1699*  1700   1701
$TMP13  001230          226#
$TMP14  001232          227#   886    887    921    2664   2667   2850*  2868*
$TMP15  001234          228#   434*   529*   1592*  1687*  2818   2822   2826   2854   2858   2862
$TMP16  001236          229#
```

```
$TMP17  001240           230#
$TMP2   001206           217#   1415*   1416    1497*   1502    1573*   1596    1692*   1705
$TMP3   001210           218#   1594*   1595*   1603*   1611    1703*   1704*   1712*   1719    2627    2633
$TMP4   001212           219#   1611*   1613*   1615    1719*   1721*   1723    2652
$TMP5   001214           220#   1615*   1723*   2650    2658    2664
$TMP6   001216           221#   2635*   2636*   2650*   2660
$TMP7   001220           222#   2761*   2763    2766*   2774*   2775    2776*   2777    2778*   2779*
$TN   = 000026           16#    726     730#    741     745#    756     760#    771     775#    786     790#    793     796
                         800#   803     806     810#    820     823     827#    844     849     852     856#    866     869
                         873#   905     924     927     931#    932     954     958     962#    963     990     994     998#
                         999    1025    1029    1033#   1034    1045    1048    1052#   1053    1072    1081    1084    1088#
                         1089   1100    1105    1109#   1114    1118    1122#   1136    1147    1156    1165    1175    1179#
                         1193   1204    1213    1222    1232    1236#   1250    1261    1270    1279    1289    1293#
$TPB    001152           200#   2374*   2385
$TPFLG  001157           204#   2332    2385
$TPS    001150           199#   2372    2385
$TRAP   013066           406    1355    1438    1523    1632    2455#
$TRAP2  013110           2466#  2477
$TRP  = 000012           2470#  2479#   2480#   2481#   2482#   2483#   2485    2486#   2487#   2488#   2489#
$TRPAD  013122           2460   2477#
$TSTNM  001102           177#   1810*   1831*   1864    1890    1912*   1917    1921    1939    1963    2683    2692    2888
$TTYIN  012264           2204   2205    2217    2235    2249    2253#
$TYPBN= ***#*** U        2483
$TYPDS  011464           2102#  2482
$TYPE   012470           2332#  2470    2478
$TYPEC  012640           2353   2360    2367    2372#   2373
$TYPEX  012706           2378   2380    2383#
$TYPOC  011262           2042#  2479
$TYPON  011276           2041   2044#   2481
$TYPOS  011236           2037#  2480
$XTSTR  010506           1877#
$$GET4= 000000           1847#
$OFILL  011461           2038*  2042*   2052    2087#
$40CAT= ***#** U         1874   1949
.     = 022262           38#    42#     52#     174#    237     399     413     414     956     1855    1859    1920    1921
                         1963   2010#   2156#   2161    2253#   2254    2260    2314    2385    2446    2506    2530    2941#
                         3019#  3031#   3322#   3325#
```

```
COMMEN  167#
ENDCOM  167#
EOPBEG  19#    1768
ERROR   61#    464     737     752     767     782     795     805     816     822     842     851     862     868     901
        915    926     940     946     952     957     971     977     985     992     1007    1014    1021    1027    1041
        1047   1070    1079    1083    1096    1102    1116    1145    1154    1163    1171    1202    1211    1220    1228
        1259   1268    1277    1285    1316    1324    1332    1339    1394    1476    1561    1602    1610    1670    1711
        1718   2669    2671    2805    2893
ESCAPE  167#
GETPRI  167#
GETSWR  167#
MORETA  18#    238
MULT    167#
NEWTST  167#   726     741     756     771     786     796     806     823     852     869     927     958     994     1029
        1048   1084    1105    1118    1175    1232    1289
POP     4#     167#    2143    2303    2435    2515    2516
PUSH    4#     167#    2102    2277    2402    2496    2502
REPORT  167#
SCOPE   62#    729     744     759     774     789     799     809     826     855     872     930     961     997     1032
        1051   1087    1108    1121    1178    1235    1292    1772
SETPRI  167#
SETTRA  2470#  2479    2480    2481    2482    2485    2486    2487    2488
SETUP   4#     167#    394
SKIP    167#   734     749     764     779     793     803     814     820     843     849     860     866     905     913
        924    932     938     944     950     954     963     969     975     983     990     999     1005    1012    1019
        1025   1034    1039    1045    1053    1071    1077    1081    1089    1094    1100    1114    1136    1146    1155
        1164   1193    1203    1212    1221    1250    1260    1269    1278
SLASH   167#
SPACE   167#
STARS   4#     167#    170     237     370     372     726     728     741     743     756     758     771     773     786
        788    796     798     806     808     823     825     852     854     869     871     927     929     958     960
        994    996     1029    1031    1048    1050    1084    1086    1105    1107    1118    1120    1175    1177    1232
        1234   1289    1291    1761    1830    1861    1924    1966    2014    2092    2160    2166    2195    2263    2317
        2388   2449    2492    2508    2536    2538    2554    2556    2674    2676    2898    2900
SWRSU   167#   415#
TRMTRP  2470#
TYPBIN  167#
TYPDEC  167#   1842
TYPNAM  167#
TYPNUM  167#
TYPOCS  167#
TYPOCT  167#   1978    2002
TYPTXT  167#
$$CMRE  168#   207     208     209     210     211     212     213     214
$$CMTM  168#   215     216     217     218     219     220     221     222     223     224     225     226     227     228
        229    230
$$ESCA  167#
$$NEWT  167#   726     741     756     771     786     796     806     823     852     869     927     958     994     1029
        1048   1084    1105    1118    1175    1232    1289
$$SET   2470#  2479    2480    2481    2482    2485    2486    2487    2488
$$SKIP  167#   793     803     820     844     849     866     905     924     932     954     963     990     999     1025
        1034   1045    1053    1072    1081    1089    1100    1114    1136    1147    1156    1165    1193    1204    1213
        1222   1250    1261    1270    1279
.EQUAT  4#     57
.HEADE  4#     6
.SETUP  4#     392
```

```
,SWRHI      4#     20
,SWRLO      4#     32#     33
,SCATC      4#     36
,SCMTA      4#    168
,SEOP       4#   1759
,SERRO      4#   1922
,SERRT      4#   1964
,SPOWE      4#   2490
,SRDDE      4#   2386
,SRDOC      4#   2261
,SREAD      4#   2158
,SSCOP      4#   1859
,STRAP      4#   2447
,STYPD      4#   2090
,STYPE      4#   2315
,STYPO      4#   2012


, ABS,  022262     000


ERRORS DETECTED:  0

DSKZ:DZDLCB.BIN,DSKZ:DZDLCB.LST/CRF/SOL/NL:TOC=DSKZ:DZDLCB.P11
RUN=TIME: 21 10 1 SECONDS
RUN=TIME RATIO: 314/33=9,4
CORE USED:  25K  (49 PAGES)
```