DL11C/D/E

DL11C/D/E OFFLINE TEST

MD-11-DZDLC-B

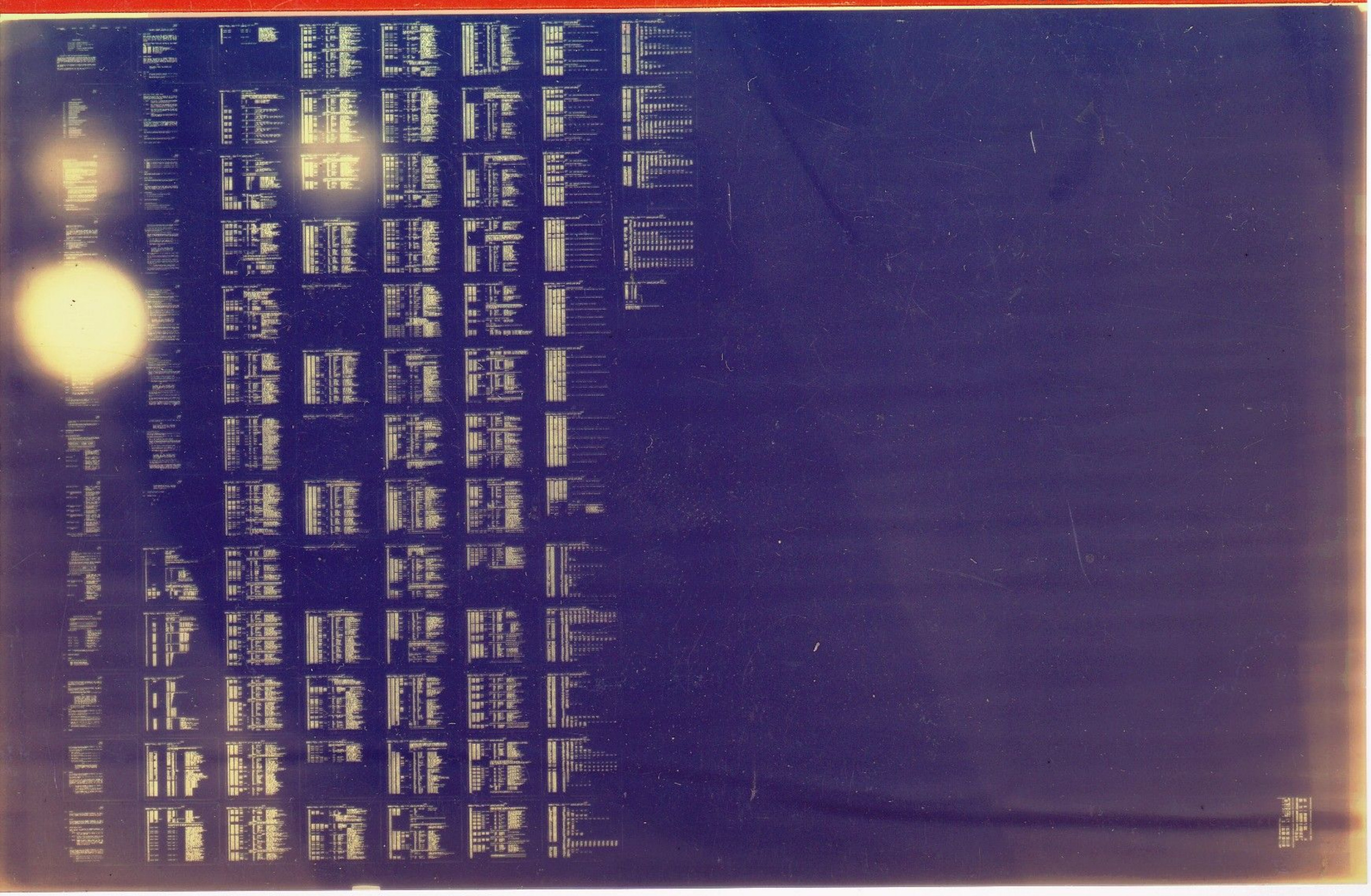
EP-DZDLC-B-DL-A

FICHE 1 OF 1

AUG 1977

digital

MADE IN USA



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  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  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

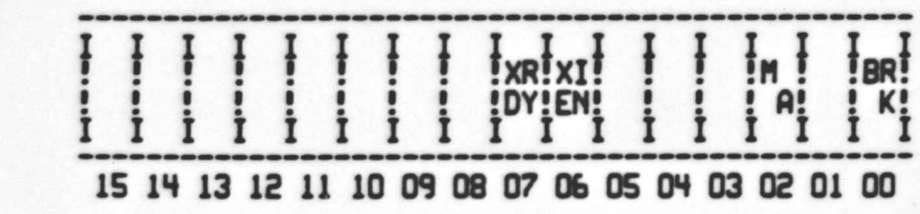
176170

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:



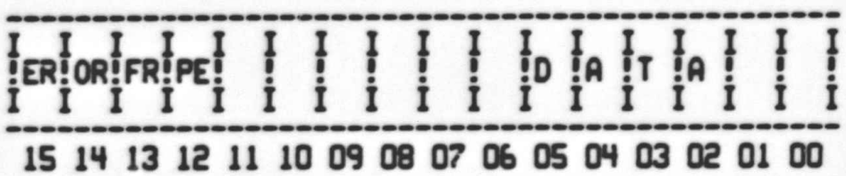
Bit assignments are defined as follows:

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.



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 'SERRTYP' routine shown below (presumes 'HALT' on error SW<15> not set).

12.5 SERRTYP

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' (SITEMB) to determine which error is to be reported. It then obtains, from the 'error Table' (SERRTB) 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 'SITEMB' is supplied by .SCMTAG, a 'SYSMAC' utility package routine.

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

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

12.6 STYPOC

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 STYPDS

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 SRDCHR, SRDLIN, SRDOCT

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

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

b. SRDLIN - 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. SRDOCT - Handles an octal number coming in from the SRDDEC 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 .SCMTAG, a 'SYSMAC' package utility routine.

12.9 STYPE

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 STRAP, STRPAD

These routines are supplied by MAINDEC-11-DZQAC-C3, the PDP-11 Maindec 'SYSMAC' utility package. The 'STRAP' routine will strip off the lower byte of a trap instruction and use it to index thru the trap table (STRPAD) 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 -

STYPE	104400	TTY typeout routine
STYPOC	104402	Type octal # (with leading zeros)
STYPOS	104404	Type octal # (no leading zeros)
STYPON	104406	Type octal # (per last character method)
STYPDS	104410	Type decimal # (with sign)
SRDCHR	104412	TTY character input
SRDLIN	104414	TTY string input
SRDOCT	104416	TTY octal # input

12.11 SPWRDN, SPWRUP

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

```
                                .NLIST  CND,MD,MC
                                .LIST   ME,SEQ,BIN
            167400              $SWR=167400
                                .ENABLE ABS

                                .TITLE  MAINDEC-11-DZDLC-B
                                ;*COPYRIGHT (C) 1977
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY E. CROWLEY/B. BURGESS
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                                ;*
            000001              $TN=1


                                .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        14            LOOP ON TEST
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        12            /C OR /D MODEL
                                ;*        11            INHIBIT ITERATIONS
                                ;*        10            BELL ON ERROR
                                ;*         9            LOOP ON ERROR
                                ;*         8            LOOP ON TEST IN SWR<7:0>
                                ;*
                                ;*         0            CREATION OF DEVICE/S TABLE
                                ;*                      OR CHANGE CHARACTER LENGTH


                                .SBTTL  TRAP CATCHER

            000000                      .=0
                                ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
            000174              .=174
000174  000000          DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000          SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                                .SBTTL  STARTING ADDRESS(ES)
000200  000137  001446          JMP     @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000137  006344          JMP     @#PRG2          ;JUMP TO USER PROGRAM NO. 2
000210  000137  006604          JMP     @#PRG3          ;JUMP TO USER PROGRAM NO. 3
000214  000137  007054          JMP     @#PRG4          ;JUMP TO USER PROGRAM NO. 4
000220  000137  007446          JMP     @#PRG5          ;JUMP TO USER PROGRAM NO. 5

            000052              .=52                ;INFORMATION LOCATION FOR ACT11
000052  000000                  .WORD   0           ;NO POWER FAIL REQUIRED <BIT15=0>
                                                    ;IS NOT MEMORY SIZE DEPENDENT <BIT14=0>
                                                    ;IS SUITABLE FOR AUTOMATIC OPERATION <BIT13=0>
```

```
                        .SBTTL  BASIC DEFINITIONS

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

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

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

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

                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
```

```
                        SW01=   2
                        SW00=   1
                        .EQUIV  SW09,SW9
                        .EQUIV  SW08,SW8
                        .EQUIV  SW07,SW7
                        .EQUIV  SW06,SW6
                        .EQUIV  SW05,SW5
                        .EQUIV  SW04,SW4
                        .EQUIV  SW03,SW3
                        .EQUIV  SW02,SW2
                        .EQUIV  SW01,SW1
                        .EQUIV  SW00,SW0

                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
  100000                BIT15=  100000
  040000                BIT14=  40000
  020000                BIT13=  20000
  010000                BIT12=  10000
  004000                BIT11=  4000
  002000                BIT10=  2000
  001000                BIT09=  1000
  000400                BIT08=  400
  000200                BIT07=  200
  000100                BIT06=  100
  000040                BIT05=  40
  000020                BIT04=  20
  000010                BIT03=  10
  000004                BIT02=  4
  000002                BIT01=  2
  000001                BIT00=  1
                        .EQUIV  BIT09,BIT9
                        .EQUIV  BIT08,BIT8
                        .EQUIV  BIT07,BIT7
                        .EQUIV  BIT06,BIT6
                        .EQUIV  BIT05,BIT5
                        .EQUIV  BIT04,BIT4
                        .EQUIV  BIT03,BIT3
                        .EQUIV  BIT02,BIT2
                        .EQUIV  BIT01,BIT1
                        .EQUIV  BIT00,BIT0

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

```
                                .SBTTL  COMMON TAGS

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

                        001100          .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000          $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
001102  000             $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103  000             $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114  000             $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115  001             $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000          $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000          $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                  .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                  .WORD   0
001134  000             $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135  000             $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                  .WORD   0
001140  177570          $SWR:   .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570          DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560          $TKS:   177560                  ;;TTY KBD STATUS
001146  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154  000             $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155  002             $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156  012             $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157  000             $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000          $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
                                                        ;;WHICH  ($REG0) WAS OBTAINED
001162  000000          $REG0:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001164  000000          $REG1:  .WORD   0               ;;CONTAINS (($REGAD)+2)
001166  000000          $REG2:  .WORD   0               ;;CONTAINS (($REGAD)+4)
001170  000000          $REG3:  .WORD   0               ;;CONTAINS (($REGAD)+6)
001172  000000          $REG4:  .WORD   0               ;;CONTAINS (($REGAD)+10)
001174  000000          $REG5:  .WORD   0               ;;CONTAINS (($REGAD)+12)
001176  000000          $REG6:  .WORD   0               ;;CONTAINS (($REGAD)+14)
001200  000000          $REG7:  .WORD   0               ;;CONTAINS (($REGAD)+16)
001202  000000          $TMP0:  .WORD   0               ;;USER DEFINED
001204  000000          $TMP1:  .WORD   0               ;;USER DEFINED
001206  000000          $TMP2:  .WORD   0               ;;USER DEFINED
001210  000000          $TMP3:  .WORD   0               ;;USER DEFINED
001212  000000          $TMP4:  .WORD   0               ;;USER DEFINED
001214  000000          $TMP5:  .WORD   0               ;;USER DEFINED
001216  000000          $TMP6:  .WORD   0               ;;USER DEFINED
001220  000000          $TMP7:  .WORD   0               ;;USER DEFINED
001222  000000          $TMP10: .WORD   0               ;;USER DEFINED
```

```
224  001224  000000                  STMP11: .WORD   0               ;;USER DEFINED
225  001226  000000                  STMP12: .WORD   0               ;;USER DEFINED
226  001230  000000                  STMP13: .WORD   0               ;;USER DEFINED
227  001232  000000                  STMP14: .WORD   0               ;;USER DEFINED
228  001234  000000                  STMP15: .WORD   0               ;;USER DEFINED
229  001236  000000                  STMP16: .WORD   0               ;;USER DEFINED
230  001240  000000                  STMP17: .WORD   0               ;;USER DEFINED
231  001242  000000                  STIMES: 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                  $LF:    .ASCIZ  <12>            ;;LINE FEED
237                                   ;;**************************************************************
238
239                                   ;THE FOLLOWING TAG(S) ARE USER SUPPLIED BY CALLING THE MACRO
240                                   ;'MORETAGS' AS ONE OF THE ARGUMENTS TO THE SYSMAC ROUTINE .$CMTAG
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
```

```
                                                ;TESTED (IF MULTIPLE DEVICE
                                                ;TESTING WAS SELECTED BY USER)
001302  000000          DLPRI:  .WORD   0       ;STORAGE LOCATION FOR THE DEVICE
                                                ;INTERRUPT PRIORITY LEVEL
001304  000000          LESS1:  .WORD   0       ;THE PRIORITY LEVEL THE CPU
                                                ;MUST BE AT TO ALLOW DEVICE INTERRUPTS.
                                                ;THIS WILL BE 1 LEVEL LESS THAN
                                                ;THE DEVICE LEVEL (BASED ON &
                                                ;CALCULATED FROM USER RESPONSE TO
                                                ;DEVICE PRIORITY LEVEL QUESTION)
001306  177740          STLMSK: 177740          ;THIS MASK IS USED BY THE 'STALL'
                                                ;ROUTINE WHICH WAITS A RANDOM NO.
                                                ;OF MILLISECONDS.  ITS' USE PREVENTS
                                                ;A STALL > 37 MSEC.  THIS LOCATION
                                                ;HOWEVER, CAN BE PATCHED BY THE
                                                ;USER TO ALLOW LARGER 'STALLS'.

                        ;END OF USER SUPPLIED TAG(S)
```

```
                        .SBTTL  ERROR POINTER TABLE

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

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

001310                  $ERRTB:

                        ;ERROR TABLE ITEM FOR ERROR MESSAGE 1

001310  015146                 EM1             ;"DL11 REGISTER REFERENCE CAUSED TIMEOUT"
001312  015215                 DH1             ;"   (PC)   (PS)   (SP)   TEST   DEVADR   REGADR   "
001314  015274                 DT1             ;  (R7)   (PSW)   (R6)   (R0)   (R1)   (R2)
001316  000000                 0               ;PRINT ALL OCTAL

                        ;ERROR TABLE ITEM FOR ERROR MESSAGE 2

001320  015312                 EM2             ;" DL11 REGISTER ERROR "
001322  015336                 DH2             ;"   (PC)   (PS)   (SP)   TEST   DEVADR   REGADR   WAS   S/B   "
001324  015434                 DT2             ;" (R7)   (PSW)   (R6)   (R0)   (R1)   (R2)   (R3)   (R4)   "
001326  000000                 0               ;PRINT ALL OCTAL

                        ;ERROR TABLE ITEM FOR ERROR MESSAGE 3

001330  015456                 EM3             ;" DL11 DATA COMPARE ERROR "
001332  015506                 DH3             ;" (PC)   (PS)   (SP)   TEST   WASADR   SHBADR   WAS   S/B   "
001334  015604                 DT3             ;"   (R7)   (PSW)   (R6)   (R0)   (R1)   (R2)   (R3)   (R4)   "
001336  000000                 0               ;PRINT ALL OCTAL

                        ;ERROR TABLE ITEM FOR ERROR MESSAGE 4

001340  015626                 EM4             ;" UNEXPECTED TRAP TO VECTOR AT LOCATION XXX "
001342  015700                 DH4             ;"   (PC)   (PS)   (SP)   TEST   "
001344  015736                 DT4             ;" (R7)   (PSW)   (R6)   (R0)   "
001346  000000                 0               ;PRINT ALL OCTAL

                        ;ERROR TABLE ITEM FOR ERROR MESSAGE 5

001350  015750                 EM5             ;" DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN)   "
001352  016025                 DH5             ;" (PC)   (PS)   (SP)   TEST   DEVADR   REGADR   (REG)   "
001354  016114                 DT5             ;"   (R7)   (PSW)   (R6)   (R0)   (R1)   (R2)   (R3)   "
001356  000000                 0

                        ;ERROR TABLE ITEM FOR ERROR MESSAGE 6

001360  015146                 EM1             ;"DL11 REGISTER REFERENCE CAUSED TIMEOUT"
001362  016134                 DH6             ;"   (PC)     (PS)     (SP)     REGADR"
001364  016174                 DT6             ;$ERRPC,$TMP0,$REG6,$REG2
```

```
                        000000                  0               ;PRINT ALL OCTAL

                                        ;ERROR TABLE ITEM FOR ERROR MESSAGE 7

                        015750          EM5             ;" DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN) "
                        016206          DH7             ;" (PC)  DEVADR  REGADR  (REG)"
                        016246          DT7             ;$ERRPC,$REG1,$REG2,$REG3
                        000000          0               ;PRINT ALL OCTAL

                                        ;ERROR TABLE ITEM FOR ERROR MESSAGE 10

                        015456          EM3             ;" DL11 DATA COMPARE ERROR "
                        016260          DH10            ;" (PC)  DEVADR  REGADR  (REG)  S/B"
                        016326          DT10            ;$ERRPC,$REG1,$REG2,$REG3,$REG4
                        000000          0               ;PRINT ALL OCTAL

                                        ;;*****************************************************************
                                        ;DL11 DEFINITIONS
                                        ;;*****************************************************************

001410  175610                  DLRCSR: 175610          ;CONTAINS ADDRESS OF RCVR CSR
001412  175612                  DLRDBR: 175612          ;CONTAINS ADDRESS OF RCVR DBR
001414  175614                  DLXCSR: 175614          ;CONTAINS ADDRESS OF XMIT CSR
001416  175616                  DLXDBR: 175616          ;CONTAINS ADDRESS OF XMIT DBR
001420  000300                  DLVECT: 300             ;CONTAINS VECTOR ADDRESS OF CURRENT DL11
001422  000000                  XFLGO:  0               ;FLAG FOR HARD XMIT ERRORS
001424  000000                  RFLGO:  0               ;FLAG FOR HARD RCVR ERRORS
001426  000000                  RFLG1:  0               ;FLAG FOR SOFT RCVR ERRORS
001430  000000                  RTRY:   0               ;COUNTS NO. OF RETRIES ON SOFT ERRORS
001432  000000                  OPTR:   0               ;CONTAINS POINTER TO OUTPUT BUFFER
001434  000000                  IPTR:   0               ;CONTAINS POINTER TO INPUT BUFFER
001436  000000                  LDOUT:  0               ;CONTAINS POINTER TO LOAD BUFFER ROUTINE
001440  000000                  TIMR1:  0               ;TIMERS FOR 256. BYTE BLOCK TRANSFERS
001442  000000                  TIMR2:  0
001444  000000                  INTFLG: 0               ;SOFTWARE INTR. FLAG


001446  000240          BEGIN:  NOP                     ;PROGRAM WILL START HERE
                        .SBTTL  INITIALIZE THE COMMON TAGS
                        ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
001450  012706  001100          MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001454  005026                  CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001456  022706  001140          CMP     #SWR,R6 ;;DONE?
001462  001374                  BNE     .-6             ;;LOOP BACK IF NO
001464  012706  001100          MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                        ;;INITIALIZE A FEW VECTORS
001470  012737  010476  000020  MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001476  012737  000340  000022  MOV     #340,@#IOTVEC+2 ;;LEVEL 7
001504  012737  010746  000030  MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001512  012737  000340  000032  MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001520  012737  013066  000034  MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001526  012737  000340  000036  MOV     #340,@#TRAPVEC+2;LEVEL 7
001534  012737  013146  000024  MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
001542  012737  000340  000026  MOV     #340,@#PWRVEC+2 ;;LEVEL 7
```

```
                001550  005067  177466                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
                001554  005067  177464                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
                001560  112767  000001  177327          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
                001566  012767  001566  177312          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
                001574  012767  001574  177306          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                                ;;EQUAL TO A "-1"  SETUP FOR A SOFTWARE SWITCH REGISTER.
                001602  013746  000004          MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
                001606  012737  001642  000004  MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
                001614  012767  177570  177316  MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
                001622  012767  177570  177312  MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
                001630  022777  177777  177302  CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
                001636  001012                  BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                        ;;AND  THE HARDWARE SWR IS NOT = -1
                001640  000403                  BR      65$             ;;BRANCH IF NO TIMEOUT
                001642  012716  001650          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
                001646  000002                  RTI
                001650  012767  000176  177262  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
                001656  012767  000174  177256          MOV     #DISPREG,DISPLAY
                001664  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

                001670  005067  177376                  CLR     MULTD           ;CLEAR MULTIPLE DEVICE
                                                                                ;TESTING FLAG
                001674  005067  177356                  CLR     TABFLG          ;CLEAR TABLE CREATION FLAG
                001700  012767  000010  177326          MOV     #8.,$TMP15      ;SET CHARACTER LENGTH DESIGNATOR
                                                                                ;FOR 8 BITS --- THIS IS THE DEFAULT
                                                                                ;LENGTH ASSUMED BY THE PROGRAM
                                                                                ;UNLESS THE USER CHANGES IT THRU
                                                                                ;THE QUESTION AND ANSWER CYCLE
                                                                                ;INITIATED BY SETTING SW<0> TO A 1
                001706  012767  000200  177366          MOV     #200,DLPRI      ;SET STANDARD PRIORITY LEVEL
                                                                                ;FOR DEVICE
                001714  032777  000400  177216          BIT #SW8,@SWR           ;IS THE 'LOOP ON TEST' SWITCH SET?
                001722  001411                          BEQ     1$              ;BRANCH IF NOT

                                        ;IF THE 'LOOP ON TEST' SWITCH WAS SET WE WILL TAKE THE NEXT BRANCH
                                        ;INSTRUCTION THUS BYPASSING TABLE CREATION
                                        ;
                                        ;IF THE USER DESIRED TO LOOP ON A TEST OF OTHER THAN THE DEFAULT DEVICE
                                        ;THEN HE SHOULD HAVE PREVIOUSLY FILLED THE FOLLOWING PROGRAM LOCATIONS
                                        ;WITH THE DESIRED DEVICE REGISTER VALUES:
                                        ;
                                        ;                       ********************************
                                        ;               UNDER   ;DL11 DEFINITIONS               ABOVE
                                        ;                       ********************************
                                        ;
                                        ;       DLRCSR:         PATCH THE ADDRESS OF THE RCVR CSR
                                        ;       DLRDBR:         PATCH THE ADDRESS OF THE RCVR DBR
                                        ;       DLXCSR:         PATCH THE ADDRESS OF THE XMIT CSR
                                        ;       DLXDBR:         PATCH THE ADDRESS OF THE XMIT DBR
                                        ;       DLVECT:         PATCH THE VECTOR ADDRESS OF THIS DL11
                                        ;

                001724  104401  016651          1$:     TYPE,   STMES           ;PRINT OUT 'MAINDEC' NAME
                001730  104401  020673                  TYPE,   FAILSA          ;TYPE FAILSAFE MESSAGE
                001734  104000                  ERROR   +0                      ;TYPE OUT THE PC VALUE
                001736  104401  021251          TYPE,   PCMSG                   ;FOLLOWED BY =PC
```

```
000466  001742  000000                          HALT                    ;WAIT FOR USER TO RESPOND
000467  001744  000443                          BR      ONCE            ;GO TO TEST DEVICE PATCHED IN BY USER
000468  001746                          1$:
000469                                  ;ENSURE THAT IF MULTIPLE DEVICE TESTING WAS BEING DONE
000470                                  ;AND THE USER 'HALTED' THE PROGRAM BEFORE ALL DEVICES
000471                                  ;WERE COMPLETED AND WENT BACK TO 'LOAD ADDRESS 200'
000472                                  ;TO RESTART THE PROGRAM THAT AS A BARE MINIMUM
000473                                  ;HE CAN RUN THE DEFAULT DEVICE (1ST RECEIVER
000474                                  ;STATUS REGISTER ADDRESS 175610)
000475                                  ;NOTE:  IF THIS IS NOT SUITABLE THE USER WILL
000476                                  ;       HAVE TO SET SW0=1 (OR UP) IN ORDER TO
000477                                  ;       RECREATE THE TABLE HE DESTROYED FROM
000478                                  ;       ABOVE
000479  001746  012767  175610  177304          MOV     #175610,DLBASE  ;1ST POSSIBLE RECEIVER CSR
000480  001754  004767  006110                  JSR     PC,DLADDR       ;FORM DL ADDRESSES FOR
000481                                                                  ;1ST POSSIBLE DEVICE
000482  001760  012767  000300  177432          MOV     #300,DLVECT     ;1ST POSSIBLE INTERRUPT VECTOR
000483  001766  005067  177264                  CLR     TABFLG          ;CLEAR TABLE CREATION FLAG
000484
000485  001772  012706  001100          RESTRT: MOV     #STACK,SP               ;SET UP STACK POINTER
000486  001776  012737  015034  000004          MOV     #BUSERR,@#ERRVEC        ;SET UP BUS ERROR VECTOR
000487  002004  012737  000340  000006          MOV     #340,@#ERRVEC+2
000488  002012  012737  015060  000010          MOV     #RSVERR,@#RESVEC        ;SET UP RSVD INSTR. VECTOR
000489  002020  012737  000340  000012          MOV     #340,@#RESVEC+2
000490
000491                                  ;THIS  NEXT SECTION WILL CHECK TO SEE IF MULTIPLE DEVICE TESTING
000492                                  ;WILL TAKE PLACE I.E.-
000493                                  ;       A)  HAS FREE RUNNING DEVICE TABLE ALREADY BEEN CREATED, AND/OR
000494                                  ;       B)  IF IT HAS, DOES USER WISH TO CHANGE IT, OR DO WE TEST DEFAULT DEVICE?
000495  002026  105767  177224                  TSTB    TABFLG          ;HAS TABLE CREATION BEEN PERFORMED?
000496  002032  001010                          BNE     ONCE            ;BRANCH IF YES TO SKIP 'MAINDEC
000497                                                                  ;TITLE' MESSAGE
000498  002034  104401  016651                  TYPE    ,STMES          ;OTHERWISE, PRINT OUT 'MAINDEC'
000499                                                                  ;NAME
000500  002040  105167  177212                  COMB    TABFLG          ;IF TABLE CREATION HAS NOT BEEN
000501                                                                  ;PERFORMED, THEN SET FLAG, AND DO SO
000502  002044  032777  000001  177066          BIT     #SW0,@SWR       ;THE PROGRAM HAS OBVIOUSLY BEEN
000503                                                                  ;RESTARTED - DOES USER WISH TO
000504                                                                  ;RESELECT VECTOR AND CONTROL REGISTER
000505                                                                  ;ADDRESSES I.E. - CREATE A NEW TABLE?
000506  002052  001012                          BNE     GO              ;BRANCH IF YES
000507  002054  005077  177334          ONCE:   CLR     @DLXCSR         ;CLEAR OUT BOTH CSR'S
000508  002060  005077  177324                  CLR     @DLRCSR
000509  002064  005777  177322                  TST     @DLRDBR         ;FLUSH RCVR "DONE" BIT
000510  002070  005777  177316                  TST     @DLRDBR
000511  002074  000167  000670                  JMP     TST1            ;OTHERWISE, GO WITH EXISTING
000512                                                                  ;TABLE OR NOT USE ANY TABLE AT
000513                                                                  ;ALL WHICHEVER THE CASE MAY BE
000514                                                                  ;(DEFAULT CASE IS 1ST POSSIBLE
000515                                                                  ;DEVICE)
000516                                  ;IF WE COME THIS PATH THE USER HAS DECIDED 1 OF 2 ALTERNATIVES:
000517                                  ;       A)  TO RUN MULTIPLE DEVICES
000518                                  ;       B)  TO CREATE A NEW TABLE TO RUN FROM, OR
000519                                  ;       C)  TO CHANGE THE CHARACTER LENGTH
000520  002100  104401  017233          GO:     TYPE,   LENGTH          ;ASK USER FOR THE CHARACTER LENGTH
000521                                                                  ;FOR WHICH HIS DEVICE IS SET
```

```
                                        RDDEC                   ;ACCEPT THE ANSWER TYPED BY USER
                                ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
                                        MOV     (SP)+,R0        ;GET THE ANSWER TYPED
                                        CMP     R0,#8.          ;IS THE NUMBER TOO HIGH?
                                        BHI     RETRY           ;IF YES - GO TO RETRY SITUATION
                                        CMP     R0,#5.          ;IS THE NUMBER TOO LOW?
                                        BLO     RETRY           ;IF YES - GO TO RETRY SITUATION
                                        MOV     R0,$TMP15       ;THE VALUE TYPED IS OK
                                                                ;STORE FOR FUTURE USE
                                        TYPE,   DEFAULT         ;ASK USER IF HE WISHES TO TEST OTHER
                                                                ;THAN THE DEFAULT DEVICE
                                        RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
                                        TST     (SP)+           ;LOOK AT THE ANSWER
                                        BNE     1$              ;BRANCH IF REPLY WAS YES
                                        JMP     @#FLUSH         ;OTHERWISE, SKIP REST OF INTERROGATION
                                1$:     MOV     #300,R0         ;START RESTORATION OF TRAPCATCHER
                                        MOV     #302,R1         ;AREA FROM LOCATIONS 300 TO 776
                                        MOV     #4,R2           ;SO THAT WE CREATE THE MULTIPLE
                                2$:     MOV     R1,(R0)         ;DEVICES TABLE WITH A CLEAN SLATE
                                        CLR     (R1)            ;       .
                                        ADD     R2,R0           ;       .
                                        ADD     R2,R1           ;       .
                                        CMP     #1000,R1        ;       .
                                        BLT     2$              ;       .
                                ;THE TRAPCATCHER VECTOR AREA FROM 300 - 776 SHOULD NOW BE RESTORED.
                                ;PROCEED TO FIND OUT THE 1ST DEVICE RECEIVER CONTROL REGISTER
                                ;ADDRESS
                                FIRSTD: TYPE    ,MFIRSTD        ;ASK USER FOR THE RECEIVER CONTROL
                                                                ;REGISTER ADDRESS OF HIS FIRST
                                                                ;DEVICE
                                        RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
                                                                ;AND STORE ON TOP OF STACK
                                ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
                                        MOV     (SP)+,R0        ;GET THE ANSWER TYPED
                                        CMP     R0,#176170      ;IS THE NUMBER TOO HIGH?
                                        BHI     RETRYO          ;IF YES-GO TO RETRY SITUATION
                                        CMP     R0,#175610      ;IS THE NUMBER TOO LOW?
                                        BLO     RETRYO          ;IF YES - GO TO RETRY SITUATION
                                        BITB    #BIT0,R0        ;NUMBER IS IN RANGE BUT IS IT
                                                                ;ON AN EVEN BOUNDARY?
                                        BNE     RETRYO          ;IF NO - GO TO RETRY SITUATION
                                ;CHECK TO SEE IF USER RESPONSE WAS TRULY A RCVR STATUS REGISTER
                                        BIT #7,R0               ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
                                                                ;USER RESPONSE EQUAL TO A ZERO?
                                        BNE     RETRYO          ;BRANCH IF NOT
                                        MOV     R0,DLBASE       ;THE 1ST ADDRESS VALUE TYPED IS OK
                                                                ;STORE FOR FUTURE USE
                                ;NOW WE ARE READY TO FIND OUT THE DEVICE INTERRUPT VECTOR
                                        MOV     DLBASE,KEEPADD  ;GET 1ST ADDRESS VALUE
                                        JSR     PC,DLADDR       ;GO FORM DL ADDRESSES FOR
                                                                ;1ST DEVICE SELECTED
                                        MOV     KEEPADD,BASEADD ;RESTORE 1ST DEVICE ADDRESS
                                VECT:   TYPE    ,MVECT          ;ASK USER FOR A VECTOR ADDRESS
                                        RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
                                                                ;AND STORE ON TOP OF STACK
                                ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
```

```
                                                    MOV     (SP)+,R0        ;GET THE ANSWER TYPED
                                                    CMP     R0,#776         ;IS THE NUMBER TOO HIGH?
                                                    BHI     RETRY1          ;IF YES - GO TO RETRY SITUATION
                                                    CMP     R0,#300         ;IS THE NUMBER TOO LOW?
                                                    BLO     RETRY1          ;IF YES - GO TO RETRY SITUATION
                                                    BITB    #BIT0,R0        ;NUMBER IS IN RANGE BUT IS IT
                                                                            ;ON AN EVEN BOUNDARY?
                                                    BNE     RETRY1          ;IF NO - GO TO RETRY SITUATION
                                            ;CHECK TO SEE IF THE USER RESPONSE WAS TRULY A RCVR VECTOR ADDRESS
                                                    BIT     #7,R0           ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
                                                                            ;USER RESPONSE EQUAL TO A ZERO?
                                                    BNE     RETRY1          ;BRANCH IF NOT
                                                    MOV     R0,DLVECT       ;THE VECTOR VALUE TYPED IS OK
                                                                            ;STORE FOR FUTURE USE
                                                    MOV     DLVECT,KEEPIV   ;GET THE FIRST VECTOR VALUE
                                                    MOV     DLVECT,BASEIV   ;SAVE FIRST VECTOR VALUE
                                                    BR      HOWMANY         ;GO TO SEE IF USER WANTS MORE
                                                                            ;THAN 1 DEVICE
                                            RETRY:  TYPE,   $QUES           ;TYPE '?' INDICATING USER TYPED
                                                                            ;SOMETHING WRONG FOR CHARACTER LENGTH
                                                    JMP     GO              ;GO BACK TO REISSUE QUESTION
                                            RETRY0: TYPE    ,$QUES          ;TYPE '?' INDICATING USER TYPE
                                                                            ;SOMETHING WRONG FOR 1ST ADDRESS
                                                    JMP     FIRSTD          ;GO BACK TO REISSUE QUESTION
                                            RETRY1: TYPE    ,$QUES          ;TYPE '?' INDICATING USER TYPED
                                                                            ;SOMETHING WRONG FOR VECTOR
                                                    JMP     VECT            ;GO BACK TO REISSUE QUESTION
                                            HOWMANY:TYPE    ,MULDEV         ;ASK USER IF HE WISHES TO RUN
                                                                            ;MULTIPLE DEVICES
                                                    RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
                                                                            ;AND STORE ON TOP OF STACK
                                                    MOV     (SP)+,R0        ;GET THE ANSWER TYPED
                                                    TST     R0              ;WAS THE ANSWER YES?
                                                    BNE     1$              ;BRANCH IF IT WAS
                                                    CLR     MULTD           ;OTHERWISE, INITIALIZE FLAG TO
                                                                            ;INDICATE NON-MULTIPLE DEVICES
                                                    BR      2$              ;SKIP NEXT INSTRUCTION
                                            1$:     COMB    MULTD           ;INITIALIZE FLAG TO INDICATE
                                                                            ;RUNNING OF MULTIPLE DEVICES
                                            2$:
                                                    TSTB    MULTD           ;ARE THERE MULTIPLE DEVICES ON
                                                                            ;THE SYSTEM?
                                                    BMI     LASTD   ;IF SO, GO TO ASK NEXT QUESTION
                                                    CLR     ACTREG          ;CLEAR DEVICE ACTIVE FLAG TO
                                                                            ;INDICATE NO RUNNING OF MULTIPLE
                                                                            ;DEVICES
                                                    CLR     ROTADD          ;CLEAR DEVICE ADDRESS POINTER IN
                                                                            ;USE WHEN RUNNING MULTIPLE DEVICES
                                                    JMP     CONQUES         ;SKIP ASKING NEXT QUESTION
                                            LASTD:
                                            ;WE WILL NOW BEGIN TO SET UP THE DEVICE ACTIVE REGISTER FOR RUNNING
                                            ;MULTIPLE DL11 DEVICES
                                                    TYPE    ,MLASTD         ;ASK USER FOR THE RECEIVER
                                                                            ;CONTROL REGISTER ADDRESS OF
                                                                            ;HIS LAST DEVICE
                                                    RDOCT                   ;ACCEPT THE ANSWER TYPED BY
```

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

```
690
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
```

```
                                        MOV     #1$,ERRVEC      ;GO TO 1$ IF TIMEOUT
                                        MOV     DLXCSR,R2       ;REGADR = XCSR ADR
                                        TST     (R2)            ;USE REGADR ON BUS
                                        BR      3$              ;<GO TO NEXT TEST IF NO TIMEOUT>
                                1$:     JSR     PC,SUERT1       ;GO SET UP ERROR INFO
                                        MOV     #2$,$ESCAPE     ;RETURN TO 2$ AFTER ERROR PRINT
                                        ERROR+1                 ;DL REFERENCE CAUSED BUS TIMEOUT
                                2$:     CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
                                3$:     MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR

                                ;;**************************************************************
                                ;*TEST 3         TEST THAT REFERENCE TO RDBR DOES NOT CAUSE TIMEOUT
                                ;;**************************************************************
                                TST3:   SCOPE
                                        MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
                                        MOV     #1$,ERRVEC      ;GO TO 1$ IF TIMEOUT
                                        MOV     DLRDBR,R2       ;REGADR = RDBR ADR
                                        TST     (R2)            ;USE REGADR ON BUS
                                        BR      3$              ;<GO TO NEXT TEST IF NO TIMEOUT>
                                1$:     JSR     PC,SUERT1       ;GO SET UP ERROR INFO
                                        MOV     #2$,$ESCAPE     ;RETURN TO 2$ AFTER ERROR PRINT
                                        ERROR+1                 ;DL REFERENCE CAUSED BUS TIMEOUT
                                2$:     CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
                                3$:     MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR

                                ;;**************************************************************
                                ;*TEST 4         TEST THAT REFERENCE TO XDBR DOES NOT CAUSE TIMEOUT
                                ;;**************************************************************
                                TST4:   SCOPE
                                        MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
                                        MOV     #1$,ERRVEC      ;GO TO 1$ IF TIMEOUT
                                        MOV     DLXDBR,R2       ;REGADR = XDBR ADR
                                        TST     (R2)            ;USE REGADR ON BUS
                                        BR      3$              ;<GO TO NEXT TEST IF NO TIMEOUT>
                                1$:     JSR     PC,SUERT1       ;GO SET UP ERROR INFO
                                        MOV     #2$,$ESCAPE     ;RETURN TO 2$ AFTER ERROR PRINT
                                        ERROR+1                 ;DL REFERENCE CAUSED BUS TIMEOUT
                                2$:     CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
                                3$:     MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR

                                ;;**************************************************************
                                ;*TEST 5         TEST THAT RCSR IS ALL ZEROES ON ENTRY
                                ;;**************************************************************
                                TST5:   SCOPE
                                        CLR     R4              ;RESULT IN RCSR S/B = 0
                                        MOV     DLRCSR,R2       ;REGADR = RCSR ADR
                                        CMP     R4,(R2)         ;[RCSR]=000000 ??
                                        BEQ     TST6            ;<BR IF YES>
                                        JSR     PC,SUER2        ;GO SET UP ERROR INFO
                                        ERROR+2                 ;RCSR NOT CLEAR ON START UP
                                ;;**************************************************************
                                ;*TEST 6         TEST THAT "READY" BIT IS ONLY BIT SET IN XCSR
                                ;;**************************************************************
                                TST6:   SCOPE
                                        MOV     #200,R4         ;RESULT IN XCSR S/B = 000200
                                        MOV     DLXCSR,R2       ;REGADR = XCSR ADR
```

```
                        CMP     R4,(R2)         ;[XCSR]=000200 ??
                        BEQ     1$T7            ;;<BR IF YES>
                        JSR     PC,$UER2        ;GO SETUP ERROR INFO
                        ERROR+2                 ;[XCSR] INCORRECT ON START UP
                ;;**************************************************************
                ;*TEST 7         TEST THAT "MAINT" BIT CAN BE SET AND CLEARED
                ;;**************************************************************
                T$T7:   SCOPE
                        MOV     #204,R4         ;RESULT IN XCSR S/B = 000204
                        MOV     DLXCSR,R2       ;REGADR = XCSR ADR
                        BIS     #BIT2,(R2)      ;SET THE "MAINT" BIT
                        CMP     R4,(R2) ;RESULT IN XCSR OK ??
                        BEQ     1$              ;;<BR IF YES>
                        JSR     PC,$UER2        ;GO SET UP ERROR INFO
                        ERROR+2                 ;MAINT. BIT FAILED TO SET PROPERLY
                1$:     MOV     #200,R4         ;RESULT IN XCSR S/B = 000200
                        BIC     #BIT2,(R2)      ;NOW CLEAR THE "MAINT" BIT
                        CMP     R4,(R2)         ;RESULT IN XCSR OK ??
                        BEQ     T$T10           ;;<BR IF YES>
                        JSR     PC,$UER2        ;GO SET UP ERROR INFO
                        ERROR+2                 ;MAINT BIT FAILED TO CLEAR PROPERLY
                ;;**************************************************************
                ;*TEST 10        TEST THAT XMIT I.E. CAN CAUSE AN INTR
                ;;**************************************************************
                T$T10:  SCOPE
                        CLR     INTFLG          ;INIT SOFTWARE INTR FLAG
                        MOV     DLVECT,R5       ;GET VECTOR ADDRESS
                        MOV     #2$,4(R5)       ;GO TO 4$ ON INTR
                        MOV     DLPRI,6(R5)     ;PRIORITY LEVEL 4
                        CLR     R5              ;INIT INTR. TIMER
                        MOV     #200,R4         ;RESULT IN XCSR S/B = 000200
                        MOV     DLXCSR,R2       ;REGADR = XCSR ADR
                        BIS     #100,(R2)       ;SET INTR. ENABLE BIT 06
                1$:     TST     INTFLG          ;DID INTR OCCUR YET ??
                        BNE     3$              ;BR IF IT DID
                        DEC     R5              ;COUNT THE TIMER
                        BNE     1$              ;BR IF NO TIMEOUT
                        MOV     #300,R4         ;RESULT IN XCSR S/B = 000300
                        JSR     PC,$UER2        ;GO SETUP ERROR INFO
                        MOV     #4$,$ESCAPE     ;RETURN TO 4$ AFTER ERROR PRINT
                        ERROR+2                 ;INTR.  FAILED
                4$:
                        BR      T$T11           ;;<GO TO NEXT TEST>
                2$:     COM     INTFLG          ;SET THE SOFTWARE FLAG
                        BIC     #100,(R2)       ;TURN OFF I.E. BIT
                        RTI                     ;RETURN CONTROL TO INTR. ROUTINE
                3$:     CMP     R4,(R2)         ;RESULT IN XCSR OK ??
                        BEQ     T$T11           ;;<BR IF YES>
                        JSR     PC,$UER2        ;GO SET UP ERROR INFO.
                        ERROR+2                 ;XMIT INTR. NOT SERVICED PROPERLY
                ;;**************************************************************
                ;*TEST 11        TEST THAT RCVR I.E. BIT CAN BE SET AND CLEARED
                ;;**************************************************************
                T$T11:  SCOPE
                        MOV     #100,R4         ;RESULT IN RCSR S/B = 000100
                        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  003556  005005                      CLR     R5              ;INIT INTR. TIMER
878  003560  105067  175420              CLRB    $TMP1           ;INIT WHERE DATA WILL BE STORED
879  003564  016702  175620              MOV     DLRCSR,R2       ;REGADR = RCSR ADR
880  003570  005012                      CLR     (R2)            ;INIT THE RCSR TO 000000
881  003572  052712  000100              BIS     #BIT6,(R2)      ;ENABLE RCVR INTERRUPTS
882  003576  052762  000004  000004      BIS     #BIT2,4(R2)     ;NOW TURN ON MAINT MODE
883  003604  112767  000252  175372      MOVB    #252,$TMP1      ;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    $TMP14,R0       ;SAVE DATA PATTERN FOR FURTHER USE
887  003622  116762  175404  000006      MOVB    $TMP14,6(R2)    ;LOAD XMIT BUFFER REG.
888  003630  005767  175610      1$:     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,$TMP0     ;SAVE ERROR PSW
893  003650  042762  000004  000004      BIC     #BIT2,4(R2)     ;DISABLE MAINT MODE
894  003656  042712  000100              BIC     #100,(R2)       ;DISABLE RCVR INTR.
895  003662  010667  175310              MOV     SP,$REG6        ;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$,$ESCAPE     ;RETURN TO 2$ AFTER ERROR ALWAYS
901  003710  104002                      ERROR+2                 ;RCVR INTERRUPT 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),$TMP1     ;GET THE RECEIVED DATA
908  003734  042712  000100              BIC     #BIT6,(R2)      ;TURN OFF RCVR INTR. ENAB
909  003740  005167  175500              COM     INTFLG          ;SET THE SOFTWARE FLAG
910  003744  000002                      RTI                     ;RETURN TO MAINLINE
911  003746  005004              4$:     CLR     R4              ;[RCSR] S/B=0
912  003750  005712                      TST     (R2)            ;IS IT ALL ZEROES ??
913  003752  001403                      BEQ     5$              ;;<BR IF YES>
```

```
                                        JSR     PC,SUER2        ;GO SET UP ERROR INFO
                                        ERROR+2                 ;RCVR INTR NOT SERVICED PROPERLY
                                5$:     MOV     DLRDBR,R1       ;SAVE WAS ADDRESS
                                        MOV     DLXDBR,R2       ;SAVE THE S/B ADDRESS
                                        JSR     PC,UPMASK       ;GET THE WAS DATA AND
                                                                ;GO MASK OFF BITS AS A FUNCTION OF
                                                                ;CHARACTER LENGTH ( 5, 6, 7, OR 8 BITS)
                                        MOVB    $TMP14,R3       ;SET UP FOR ERROR CHECKING
                                        MOVB    R0,R4           ;GET THE S/B DATA
                                        CMP     R4,R3           ;WAS = S/B ??
                                        BEQ     TST13           ;;<BR IF YES>
                                        JSR     PC,SUERR1       ;GO SET UP THE ERROR INFO
                                        ERROR+3                 ;DATA COMPARE ERROR
                                ;;**************************************************************
                                ;;*TEST 13       TEST THAT "REQ TO SEND" ASSERTS "RING"
                                ;;**************************************************************
                                TST13:  SCOPE
                                        BIT     #SW12,@SWR      ;ARE WE TESTING /C OR /D MODEL?
                                        BNE     TST14           ;;<BRANCH IF YES>
                                        MOV     #140004,R4      ;RESULT IN RCSR S/B = 140004
                                        MOV     DLRCSR,R2       ;REGADR = RCSR ADR
                                        CLR     (R2)            ;INIT THE RCSR TO 000000
                                        BIS     #BIT2,(R2)      ;SET "REQ TO SEND"
                                        BIT     #BIT15,@DLRCSR  ;DID "RING" SET "DATA SET INT" ?
                                        BNE     1$              ;;<BR IF YES>
                                        JSR     PC,SUER2        ;GO SET UP ERROR INFO.
                                        ERROR+2                 ;"RING" TRANSITION FAILED TO SET "DATA SET INT"
                                                                ;NOTE: "BIT  #BIT15,(R2)" RESETS BIT15
                                1$:     MOV     #40004,R4       ;RESULT IN RCSR S/B = 40004
                                        CMP     R4,(R2)         ;BOTH "RING" AND "REQ TO SEND" ASSERTED ?
                                        BEQ     2$              ;;<BR IF YES>
                                        JSR     PC,SUER2        ;GO SET UP ERROR INFO.
                                        ERROR+2                 ;"RING" OR "REQ TO SEND" FAILED TO SET
                                2$:     CLR     R4              ;RESULT IN RCSR S/B = 000000
                                        BIC     #BIT2,(R2)      ;CLEAR "REQ TO SEND"
                                        BIT     #BIT15,@DLRCSR  ;DID "DATA SET INT" GET SET ??
                                        BEQ     3$              ;;<BR IF NOT>
                                        JSR     PC,SUER2        ;GO SET UP ERROR INFO
                                        ERROR+2                 ;CLEARING "RING" SET "DATA SET INT"
                                3$:     CMP     R4,(R2)         ;RCSR CONTAIN ALL ZEROES ??
                                        BEQ     TST14           ;;<BR IF YES>
                                        JSR     PC,SUER2        ;GO SET UP ERROR INFO.
                                        MOV     .+6,$REG7       ;SAVE THE ERROR PC
                                        ERROR+2                 ;CLEARING  "REQ TO SEND" FAILED TO CLEAR "RING"
                                ;;**************************************************************
                                ;;*TEST 14       TEST THAT "SEC XMIT" ASSERTS "SEC REC" AND "DATA SET INT"
                                ;;**************************************************************
                                TST14:  SCOPE
                                        BIT     #SW12,@SWR      ;ARE WE TESTING /C OR /D MODEL?
                                        BNE     TST15           ;;<BRANCH IF YES>
                                        MOV     DLRCSR,R2       ;REGADR = RCSR ADR
                                        CLR     (R2)            ;INIT RCSR TO 000000
                                        MOV     #102010,R4      ;CONTENTS OF RCSR S/B = 102010
                                        BIS     #BIT3,(R2)      ;SET "SEC XMIT" BIT
                                        BIT     #BIT15,@DLRCSR  ;DID "DATA SET INT" SET ??
                                        BNE     1$              ;;<BR IF YES>
```

```
004206  004767  007640                  JSR     PC,SUER2        ;GO SET UP ERROR INFO
004212  104002                          ERROR+2                 ;"DATA SET INT" FAILED TO SET-NOTE THAT
                                                                ;"BIT  #BIT15,(R2)" RESETS BIT15
004214  012704  002010          1$:     MOV     #2010,R4        ;RESULT IN RCSR S/B = 2010
004220  020412                          CMP     R4,(R2)         ;ARE "SEC XMIT" AND "SEC REC" BOTH SET ?
004222  001403                          BEQ     2$              ;;(BR IF YES)
004224  004767  007622                  JSR     PC,SUER2        ;GO SET UP ERROR INFO
004230  104002                          ERROR+2                 ;"SEC XMIT" OR "SEC REC" FAILED TO SET
                                                                ;OR "DATA SET INT" FAILED TO BE CLEARED
                                                                ;WHEN REFERENCING RCSR
004232  012704  100000          2$:     MOV     #BIT15,R4       ;RESULT IN RCSR S/B = 100000
004236  042712  000010                  BIC     #BIT3,(R2)      ;CLEAR "SEC XMIT" BIT
004242  032777  100000  175140          BIT     #BIT15,@DLRCSR  ;DID CLEARING IT SET "DATA SET INT"??
004250  001003                          BNE     3$              ;;(BR IF YES)
004252  004767  007574                  JSR     PC,SUER2        ;GO SET UP ERROR INFO.
004256  104002                          ERROR+2                 ;CLEARING "SEC XMIT" FAILED TO SET "DATA
                                                                ;SET INT. (NOTE THAT REFERENCING RCSR
                                                                ; CLEARS "DATA SET INT"
004260  005004                  3$:     CLR     R4              ;RESULT IN RCSR S/B = 000000
004262  020412                          CMP     R4,(R2)         ;"SEC XMIT" AND "SEC REC" CLEAR ?
004264  001403                          BEQ     TST15           ;;(BR IF YES)
004266  004767  007560                  JSR     PC,SUER2        ;GO SETUP ERROR INFO
004272  104002                          ERROR+2                 ;"SEC XMIT" OR "SEC REC" FAILED TO CLEAR
                                                                ;OR REFERENCING RCSR  FAILED TO CLEAR "DATA SET INT"
                                ;;**************************************************************
                                ;*TEST 15        TEST THAT "DTR" CAN ASSERT "CLR TO SEND" AND "CAR DET"
                                ;;**************************************************************
004274  000004                  TST15:  SCOPE
004276  032777  010000  174634          BIT     #SW12,@SWR      ;ARE WE TESTING /C OR /D MODEL?
004304  001046                          BNE     TST16           ;;(BRANCH IF YES)
004306  016702  175076                  MOV     DLRCSR,R2       ;REGADR = RCSR ADR
004312  005012                          CLR     (R2)            ;INIT RCSR TO 000000
004314  012704  130002                  MOV     #130002,R4      ;RESULT IN RCSR S/B = 130002
004320  052712  000002                  BIS     #BIT1,(R2)      ;SET "DTR" BIT
004324  032777  100000  175056          BIT     #BIT15,@DLRCSR  ;DID "DATA SET INT" SET ??
004332  001003                          BNE     1$              ;;(BR IF YES)
004334  004767  007512                  JSR     PC,SUER2        ;GO SET UP ERROR INFO.
004340  104002                          ERROR+2                 ;"DATA SET INT" FAILED TO SET -
                                                                ;NOTE: THE REFERENCE TO RCSR  ABOVE WILL
                                                                ;WILL UNCONDITIONALLY CLEAR RCSR BIT 15.
004342  012704  030002          1$:     MOV     #30002,R4       ;RESULT IN RCSR S/B = 30002
004346  020412                          CMP     R4,(R2)         ;"DTR", "CLR TO SEND", AND "CAR DET" ALLSET
004350  001403                          BEQ     2$              ;;(BR IF ALL SET)
004352  004767  007474                  JSR     PC,SUER2        ;GO SET UP ERROR INFO
004356  104002                          ERROR+2                 ;"DTR", "CLR TO SEND" OR "CAR DET" FAILED
                                                                ;TO SET OR "DATA SET INT" FAILED TO CLEAR
004360  012704  100000          2$:     MOV     #BIT15,R4       ;RESULT IN RCSR S/B = 100000
004364  042712  000002                  BIC     #BIT1,(R2)      ;NOW CLEAR "DTR"
004370  032777  100000  175012          BIT     #BIT15,@DLRCSR  ;DID "DATA SET INT" SET ??
004376  001003                          BNE     3$              ;;(BR IF YES)
004400  004767  007446                  JSR     PC,SUER2        ;GO SETUP ERROR INFO
004404  104002                          ERROR+2                 ;"DATA SET INT" FAILED TO SET WHEN "DTR"
                                                                ;WENT TO A ZERO.
004406  005004                  3$:     CLR     R4              ;RESULT IN RCSR S/B = 000000
004410  020412                          CMP     R4,(R2)         ;DID ALL BITS CLEAR??
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     DLPRI,(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     #BIT0,(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,$TIMES       ;;DO 1 ITERATION
1123  004770  004767  007212                JSR     PC,SUVEC        ;GO SET UP VECTORS
1124  004774  005067  174430                CLR     RTRY            ;INITIALIZE RETRY FLAG
1125  005000  012767  014362  174430  1$:   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$:   TST     XFLGO           ;ANY HARD XMIT ERRORS ??
1128  005016  001040                        BNE     5$              ;BR IF YES
1129  005020  005767  174400                TST     RFLGO           ;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$,$ESCAPE     ;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$,$ESCAPE     ;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$,$ESCAPE     ;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     $TMP1,R3        ;GET CONTENTS OF ERROR RDBR
1169  005222  004767  006652                JSR     PC,SUERR1       ;GO SETUP ERROR INFO
1170  005226  012767  005236  174010        MOV     #10$,$ESCAPE    ;GO TO 10$ 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$:   TST     XFLGO           ;ANY HARD XMIT ERRORS ??
1185  005310  001040                      BNE     5$              ;BR IF YES
1186  005312  005767  174106              TST     RFLGO           ;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,$TIMES       ;;DO 1 ITERATION
1237  005554  004767  006426                    JSR     PC,SUVEC        ;GO SET UP VECTORS
1238  005560  005067  173644                    CLR     RTRY            ;INITIALIZE RETRY FLAG
1239  005564  012767  014424  173644    1$:     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$:     TST     XFLGO           ;ANY HARD XMIT ERRORS ??
1242  005602  001040                            BNE     5$              ;BR IF YES
1243  005604  005767  173614                    TST     RFLGO           ;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,SUERR1       ;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,SUERR1       ;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     $TMP1,R3        ;GET CONTENTS OF ERROR RDBR
1283  006006  004767  006066                    JSR     PC,SUERR1       ;GO SETUP ERROR INFO
1284  006012  012767  006022  173224            MOV     #10$,$ESCAPE    ;GO TO 10$ 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                    T25:    SCOPE
1293  006040  012767  000001  173174            MOV     #1,$TIMES       ;;DO 1 ITERATION
1294  006046  004767  006134                    JSR     PC,SUVEC        ;GO SET UP VECTORS
1295  006052  005067  173352                    CLR     RTRY            ;INITIALIZE RETRY FLAG
1296  006056  012767  014460  173352    1$:     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$:     TST     XFLGO           ;ANY HARD XMIT ERRORS ??
1299  006074  001042                            BNE     5$              ;BR IF YES
1300  006076  005767  173322                    TST     RFLGO           ;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  104401  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
```

```
                                ;THIS IS PROGRAM #2
                                ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
                                ;       A)  SELECTION OF A 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
                                ;
006344  012706  001100        PRG2:   MOV     #STACK,SP       ;INITIALIZE THE STACK POINTER
006350  012737  013066  000034        MOV     #$TRAP,@#TRAPVEC        ;TRAP VECTOR FOR TRAP CALLS
006356  012737  000340  000036        MOV     #340,@#TRAPVEC+2        ;LEVEL 7
006364  012737  010746  000030        MOV     #$ERROR,@#EMTVEC        ;EMT VECTOR FOR ERROR ROUTINE
006372  012737  000340  000032        MOV     #340,@#EMTVEC+2         ;LEVEL 7
006400  104401  016700                TYPE    ,PRG2M                  ;INDICATE THAT USER SELECTED
                                                                        ;PROGRAM #2
006404  104401  020322        PRG2A:  TYPE    ,LINTAD                 ;ASK USER FOR THE TRANSMITTER
                                                                        ;DATA BUFFER ADDRESS OF THE DEVICE
                                                                        ;HE WISHES TO TEST
006410  104410                        RDOCT                           ;ACCEPT THE ANSWER TYPED BY USER
                                                                        ;AND STORE ON TOP OF STACK
                                ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
006412  012602                        MOV     (SP)+,R2                ;GET THE ANSWER TYPED
006414  020227  176176                CMP     R2,#176176              ;IS THE NUMBER TOO HIGH?
006420  101065                        BHI     REDO1                   ;IF YES - GO TO RETRY SITUATION
006422  020227  175616                CMP     R2,#175616              ;IS THE NUMBER TOO LOW?
006426  103462                        BLO     REDO1                   ;IF YES - GO TO RETRY SITUATION
006430  132702  000001                BITB    #BIT0,R2                ;NUMBER IS IN RANGE BUT IS IT
                                                                        ;ON AN EVEN BOUNDARY?
006434  001057                        BNE     REDO1                   ;IF NOT GO TO RETRY SITUATION
                                ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
006436  010203                        MOV     R2,R3                   ;GET THE USER RESPONSE
006440  142703  000370                BICB    #370,R3                 ;MASK OFF LOWER BYTE EXCEPT FOR
                                                                        ;LEAST SIGNIFICANT DIGIT
006444  122703  000006                CMPB    #6,R3                   ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
                                                                        ;USER RESPONSE EQUAL TO A SIX?
006450  001051                        BNE     REDO1                   ;BRANCH IF NOT
006452  010267  172524                MOV     R2,$TMP0                ;THE TRANSMITTER ADDRESS
                                                                        ;TYPED IS OK - STORE FOR
                                                                        ;FUTURE USE
                                ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
006456  016746  171322                MOV     ERRVEC,-(SP)            ;SAVE THE TIMEOUT VECTOR
006462  012767  006474  171314        MOV     #2$,ERRVEC              ;SET UP TIMEOUT SERVICE ADDRESS
006470  005712                        TST     (R2)                    ;IF PRESENT WE WILL EXECUTE THE
                                                                        ;NEXT INSTRUCTION - IF NOT
                                                                        ;WE GO TO 2$:
006472  000412                        BR      4$                      ;BRANCH IF PRESENT
006474  004767  005460        2$:     JSR     PC,SUERT2               ;GO SET UP FOR ERROR INFORMATION
006500  012767  006510  172536        MOV     #3$,$ESCAPE             ;POINT OF RETURN AFTER ERROR REPORT
006506  104006                        ERROR   +6                      ;XDBR REFERENCE CAUSED TIMEOUT
006510  022626                3$:     CMP     (SP)+,(SP)+             ;CLEAN STACK FROM TIMEOUT
006512  012667  171266                MOV     (SP)+,ERRVEC            ;RESTORE TIMEOUT VECTOR
006516  000426                        BR      REDO1                   ;GO TO RETRY SITUATION
006520  012667  171260        4$:     MOV     (SP)+,ERRVEC            ;DEVICE REGISTER IS PRESENT!
                                                                        ;RESTORE TIMEOUT VECTOR
                                ;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)+,$TMP1     ;GET THE ANSWER TYPED
1408                                ;NOTE:  THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE
1409                                ;       OCTAL ASCII EQUIVALENT OF THE CHARACTER E.G. A=101
1410  006536  104401  020511                TYPE    ,SELDLY         ;ASK THE USER FOR THE DELAY
1411                                                                ;IN MSEC (OCTAL NO.) BETWEEN
1412                                                                ;CHARACTER TRANSFERS
1413  006542  104410                        RDOCT                   ;ACCEPT THE ANSWER TYPED BY
1414                                                                ;USER AND STORE ON TOP OF STACK
1415  006544  012667  172436                MOV     (SP)+,$TMP2     ;GET THE ANSWER TYPED
1416  006550  116767  172432  000012  1$:   MOVB    $TMP2,2$        ;SET THE DELAY COUNT ARGUMENT
1417                                                                ;FOR TIMER ROUTINE
1418  006556  116777  172422  172416        MOVB    $TMP1,@$TMP0    ;LOAD THE TRANSMITTER DATA
1419                                                                ;BUFFER WITH THE CHARACTER
1420  006564  004767  004750                JSR     PC,DELAY        ;GO OFF TO WAIT THE SPECIFIED
```

```
                                                                ;NO. OF MSEC. BEFORE ISSUING
                                                                ;ANOTHER CHARACTER
006570  000000                  2$:     .WORD   0               ;THIS IS WHERE THE DELAY COUNT RESIDES
006572  000766                  BR      1$                      ;GO BACK TO ISSUE ANOTHER CHARACTER
006574  104401  001252          REDO1:  TYPE    ,$QUES          ;TYPE A QUESTION MARK(?)
006600  000167  177600          JMP     PRG2A                   ;REITERATE THE XDBR QUESTION TO USER
                                ;THIS IS PROGRAM #3
                                ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
                                ;       A)  SELECTION OF A TRANSMITTER DATA BUFFER
                                ;       B)  SELECTION OF A CHARACTER FOR CONTINUOUS TRANSFER
                                ;           IN MAINTENANCE MODE
                                ;       C)  SELECTION OF AN EXPIRATION TIME IN MILLISECONDS
                                ;           BETWEEN EACH TRANSMITTER DATA BUFFER CHARACTER TRANSFER
                                ;       D)  A TIGHT SCOPE LOOP LOCK ON A SPECIFIC CHARACTER
                                ;
006604  012706  001100          PRG3:   MOV     #STACK,SP       ;INITIALIZE THE STACK POINTER
006610  012737  013066  000034          MOV     #$TRAP,@#TRAPVEC        ;TRAP VECTOR FOR TRAP CALLS
006616  012737  000340  000036          MOV     #340,@#TRAPVEC+2        ;LEVEL 7
006624  012737  010746  000030          MOV     #$ERROR,@#EMTVEC        ;EMT VECTOR FOR ERROR ROUTINE
006632  012737  000340  000032          MOV     #340,@#EMTVEC+2         ;LEVEL 7
006640  104401  016744                  TYPE    ,PROG3M         ;INDICATE THAT USER SELECTED
                                                                ;PROGRAM #3
006644  104401  020322          PRG3A:  TYPE    ,LINTAD         ;ASK USER FOR THE TRANSMITTER DATA
                                                                ;BUFFER ADDRESS OF THE DEVICE
                                                                ;HE WISHES TO TEST
006650  104410                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY
                                                                ;USER AND STORE ON TOP OF STACK
                                ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
006652  012602                          MOV     (SP)+,R2        ;GET THE ANSWER TYPED
006654  020227  176176                  CMP     R2,#176176      ;IS THE NUMBER TOO HIGH?
006660  101071                          BHI     REDO2           ;IF YES - GO TO RETRY SITUATION
006662  020227  175616                  CMP     R2,#175616      ;IS THE NUMBER TOO LOW?
006666  103466                          BLO     REDO2           ;IF YES - GO TO RETRY SITUATION
006670  132702  000001                  BITB    #BIT0,R2        ;NUMBER IS IN RANGE BUT IS IT
                                                                ;ON AN EVEN BOUNDARY?
006674  001063                          BNE     REDO2           ;IF NOT - GO TO RETRY SITUATION
                                ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XDBR DBR ADDRESS
006676  010203                          MOV     R2,R3           ;GET THE USER RESPONSE
006700  142703  000370                  BICB    #370,R3         ;MASK OFF LOWER BYTE EXCEPT FOR
                                                                ;LEAST SIGNIFICANT DIGIT
006704  122703  000006                  CMPB    #6,R3           ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
                                                                ;USER RESPONSE EQUAL TO A TWO?
006710  001055                          BNE     REDO2           ;BRANCH IF NOT
006712  010267  172264                  MOV     R2,$TMP0        ;THE TRANSMITTER ADDRESS TYPED IS
                                                                ;OK - STORE FOR FUTURE USE
                                ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
006716  016746  171062                  MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
006722  012767  006734  171054          MOV     #2$,ERRVEC      ;SET UP TIMEOUT SERVICE ADDRESS
006730  005712                          TST     (R2)            ;IF PRESENT WE WILL EXECUTE THE
                                                                ;NEXT INSTRUCTION - IF NOT WE
                                                                ;GO TO 2$:
006732  000412                          BR      4$              ;BRANCH IF PRESENT
006734  004767  005220          2$:     JSR     PC,SUERT2       ;GO SET UP FOR ERROR INFORMATION
006740  012767  006750  172276          MOV     #3$,$ESCAPE     ;POINT OF RETURN AFTER ERROR REPORT
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)+,$TMP1     ;GET THE ANSWER TYPED
1490                                  ;NOTE:  THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE
1491                                  ;       OCTAL ASCII EQUIVALENT OF THE CHARACTER E.G. B=102
1492  006776  104401  020511                  TYPE    ,SELDLY         ;ASK THE USER FOR THE DELAY
1493                                                                    ;IN MSEC (OCTAL NO.) BETWEEN
1494                                                                    ;CHARACTER TRANSFERS
1495  007002  104410                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY
1496                                                                    ;USER AND STORE ON TOP OF STACK
1497  007004  012667  172176                  MOV     (SP)+,$TMP2     ;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    $TMP2,2$        ;SET THE DELAY COUNT ARGUMENT
1503                                                                    ;FOR TIMER ROUTINE
1504  007026  116777  172152  172146          MOVB    $TMP1,@$TMP0    ;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     RED03           ;IF YES - GO TO RETRY SITUATION
1538  007132  020227  175616                            CMP     R2,#175616      ;IS THE NUMBER TOO LOW?
1539  007136  103533                                    BLO     RED03           ;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     RED03           ;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     RED03           ;BRANCH IF NOT
1550  007162  010267  172014                            MOV     R2,$TMP0        ;THE TRANSMITTER ADDRESS TYPED
1551                                                    ;IS OK - STORE FOR FUTURE USE
1552                                            ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
1553  007166  016746  170612                            MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
1554  007172  012767  007204  170604                    MOV     #2$,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$:             JSR     PC,SUERT2       ;GO SET UP FOR ERROR INFORMATION
1560  007210  012767  007220  172026                    MOV     #3$,$ESCAPE     ;POINT OF RETURN AFTER ERROR REPORT
1561  007216  104006                                    ERROR   +6              ;XDBR REFERENCE CAUSED TIMEOUT
1562  007220  022626                          3$:       CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
1563  007222  012667  170556                            MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
1564  007226  000477                                    BR      RED03           ;GO TO RETRY SITUATION
1565  007230  012667  170550          4$:             MOV     (SP)+,ERRVEC    ;DEVICE REGISTER IS PRESENT!
1566                                                    ;RESTORE TIMEOUT VECTOR
1567  007234  104401  020604                            TYPE    ,RSTALL         ;ASK THE USER IF HE DESIRES SOME
1568                                                    ;RANDOM NO. OF MSEC. WAIT TIME
1569                                                    ;BEFORE CHECKING FOR XCSR DONE
1570                                                    ;FLAG
1571  007240  104410                                    RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
1572                                                    ;AND STORE ON TOP OF STACK
1573  007242  012667  171740                            MOV     (SP)+,$TMP2     ;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)+,$TMP1     ;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,$TMP15       ;THE VALUE TYPED IS OK
1593                                                                  ;STORE FOR FUTURE USE
1594  007310  016767  171666  171672          MOV     $TMP0,$TMP3     ;GET THE XDBR ADDRESS
1595  007316  162767  000002  171664          SUB     #2,$TMP3        ;FORM THE XCSR ADDRESS
1596  007324  005767  171656          1$:     TST     $TMP2           ;DO WE RANDOM STALL?
1597  007330  001402                          BEQ     2$              ;BRANCH IF IT WASN'T DESIRED
1598  007332  004767  004246                  JSR     PC,STALL        ;GO STALL RANDOM VALUE OF MSEC.
1599  007336  004767  004352          2$:     JSR     PC,TIMETX       ;GO WAIT FOR TRANSMITTER DONE
1600                                                                  ;BIT TO SET
1601  007342  104401  017120                  TYPE    ,XDB            ;TYPE TRANSMITTER DONE BIT MESSAGE
1602  007346  104000                          ERROR   +0              ;XCSR DONE BIT NEVER SET
1603  007350  052777  000004  171632          BIS     #BIT2,@$TMP3    ;SET THE MAINTENANCE BIT IN THE
1604                                                                  ;TRANSMITTER CONTROL STATUS REGISTER
1605  007356  016777  171622  171616          MOV     $TMP1,@$TMP0    ;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     $TMP3,$TMP4     ;GET THE TRANSMITTER CONTROL
1612                                                                  ;STATUS REGISTER ADDRESS
1613  007404  162767  000002  171600          SUB     #2,$TMP4        ;FORM THE RECEIVER DATA BUFFER
1614                                                                  ;ADDRESS
1615  007412  017767  171574  171574          MOV     @$TMP4,$TMP5    ;STORE THE CHARACTER FROM THE
1616                                                                  ;RECEIVER BUFFER + REST OF CONTENTS
1617  007420  004767  004326                  JSR     PC,DATCHK       ;GO TO COMPARE EXPECTED & RECEIVED
1618                                                                  ;DATA
1619  007424  000737                          BR      1$              ;GO BACK TO ISSUE ANOTHER CHARACTER
1620  007426  104401  001252          REDO3:  TYPE    ,$QUES          ;TYPE A QUESTION MARK(?)
1621  007432  000167  177456                  JMP     PRG4A           ;REITERATE THE XDBR QUESTION TO USER
1622  007436  104401  001252          REDO3A: TYPE,   $QUES           ;TYPE '?' INDICATING USER TYPED
1623                                                                  ;SOMETHING WRONG FOR CHARACTER LENGTH
1624  007442  000167  177612                  JMP     PRG4C           ;GO BACK TO REISSUE QUESTION
1625
1626                                  ;THIS IS PROGRAM #5
1627                                  ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW USER PARAMETERS
1628                                  ;FOR RUNNING A BINARY COUNT IN MAINTENANCE MODE
1629                                  ;
1630
1631  007446  012706  001100          PRG5:   MOV     #STACK,SP       ;INITIALIZE THE STACK POINTER
1632  007452  012737  013066  000034          MOV     #$TRAP,@#TRAPVEC        ;TRAP VECTOR FOR TRAP CALLS
1633  007460  012737  000340  000036          MOV     #340,@#TRAPVEC+2        ;LEVEL 7
1634  007466  012737  010746  000030          MOV     #$ERROR,@#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,$TMP0        ;THE TRANMITTER ADDRESS TYPED
1660                                                          ;IS OK - STORE FOR FUTURE USE
1661                          ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
1662  007560  016746  170220          MOV     ERRVEC,-(SP)    ;SAVE THE TIMOUT VECTOR
1663  007564  012767  007576  170212  MOV     #2$,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$:     MOV     (SP)+,ERRVEC    ;DEVICE REGISTER IS PRESENT!
1675                                                          ;ASK THE USER IF HE DESIRES SOME
1676                                                          ;RANDOM NO. OF MSEC. WAIT TIME
1677                                                          ;BEFORE CHECKING XCSR DONE FLAG
1678  007626  104401  017233  PRG5C:  TYPE,   LENGTH  ;ASK USER FOR THE CHARACTER LENGTH
1679                                                          ;FOR WHICH HIS DEVICE IS SET
1680  007632  104411                  RDDEC                   ;ACCEPT THE ANSWER TYPED BY USER
1681                          ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1682  007634  012600                  MOV     (SP)+,R0        ;GET THE ANSWER TYPED
1683  007636  020027  000010          CMP     R0,#8.          ;IS THE NUMBER TOO HIGH?
1684  007642  101106                  BHI     REDO4A          ;IF YES - GO TO RETRY SITUATION
1685  007644  020027  000005          CMP     R0,#5.          ;IS THE NUMBER TOO LOW?
1686  007650  103503                  BLO     REDO4A          ;IF YES - GO TO RETRY SITUATION
1687  007652  010067  171356          MOV     R0,$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     $TMP0,$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     $TMP1,@$TMP0    ;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 - RECEVER 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 ($PASS)
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
```

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

                                        .SBTTL  ERROR HANDLER ROUTINE

                                        ;;**********************************************************************
```

```
                                        ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                        ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                        ;*AND GO TO $ERRTYP ON ERROR
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW15=1          HALT ON ERROR
                                        ;*SW13=1          INHIBIT ERROR TYPEOUTS
                                        ;*SW10=1          BELL ON ERROR
                                        ;*SW09=1          LOOP ON ERROR
                                        ;*CALL
                                        ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER

 010746                                 $ERROR:
 010746  105267  170131                 7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
 010752  001775                         BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
 010754  016777  170122  170160         MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
 010762  032777  002000  170150         BIT     #BIT10,@SWR     ;;BELL ON ERROR?
 010770  001402                         BEQ     1$              ;;NO - SKIP
 010772  104401  001246                 TYPE    ,$BELL          ;;RING BELL
 010776  005267  170110                 1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
 011002  011667  170110                 MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
 011006  162767  000002  170102         SUB     #2,$ERRPC
 011014  117767  170076  170072         MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
 011022  032777  020000  170110         BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
 011030  001004                         BNE     20$             ;;SKIP TYPEOUTS
 011032  004767  000044                 JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
 011036  104401  001253                 TYPE    ,$CRLF
 011042                                 20$:
 011042  005777  170072                 2$:     TST     @SWR            ;;HALT ON ERROR
 011046  100001                         BPL     3$              ;;SKIP IF CONTINUE
 011050  000000                         HALT                    ;;HALT ON ERROR!
 011052  032777  001000  170060         3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
 011060  001402                         BEQ     4$              ;;BR IF NO
 011062  016716  170022                 MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
 011066  005767  170152                 4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
 011072  001402                         BEQ     5$              ;;BR IF NONE
 011074  016716  170144                 MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
 011100                                 5$:
 011100  000002                         RTI                     ;;RETURN

                                        .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

                                        ;;**************************************************************
                                        ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
                                        ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
                                        ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

 011102                                 $ERRTYP:
 011102  104401  001253                 TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
 011106  010046                         MOV     R0,-(SP)        ;;SAVE R0
 011110  005000                         CLR     R0              ;;PICKUP THE ITEM INDEX
 011112  153700  001114                 BISB    @#$ITEMB,R0
 011116  001004                         BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                                 ;;TYPE THE PC OF THE ERROR
 011120  016746  167772                 MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                 ;;ERROR ADDRESS
 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  000040                  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                          ;;**************************************************************
```

```
                                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
                                        ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
                                        ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
                                        ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
                                        ;*REPLACED WITH SPACES.
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
                                        ;*      TYPDS                   ;;GO TO THE ROUTINE

011464                                  $TYPDS:
011464  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
011466  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
011470  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
011472  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
011474  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
011476  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
011502  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
011506  100004                          BPL     1$              ;;BR IF INPUT IS POS.
011510  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
011512  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
011520  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
011522  012703  011700                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
011526  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
011532  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
011534  016001  011670                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
011540  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
011542  002402                          BLT     4$              ;;BR IF DONE
011544  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
011546  000774                          BR      3$
011550  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
011552  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
011554  001002                          BNE     5$              ;;FALL THROUGH IF 0
011556  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
011560  100407                          BMI     7$              ;;BR IF YES
011562  106316                  5$:     ASLB    (SP)            ;;MSD?
011564  103003                          BCC     6$              ;;BR IF NO
011566  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
011574  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
011600  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
011604  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
011606  005720                          TST     (R0)+           ;;JUST INCREMENTING
011610  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
011614  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
011616  003002                          BGT     8$              ;;GO TO EXIT
011620  010502                          MOV     R5,R2           ;;GET THE LSD
011622  000764                          BR      6$              ;;GO CHANGE TO ASCII
011624  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
011626  100003                          BPL     9$              ;;BR IF NO
011630  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
011636  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
011640  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
011642  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
011644  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
011646  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
011650  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
011652  104401  011700                  TYPE    ,$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          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
2206  012044  101456                          BLOS    4$              ;;BR IF YES
2207  012046  104406                          ROCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
2208  012050  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
2209  012052  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
2210  012056  001022                          BNE     5$              ;;BR IF NO
2211  012060  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
2212  012062  001007                          BNE     6$              ;;BR IF NO
2213  012064  112767  000134  000170        MOVB    #'\,9$          ;;TYPE A BACK SLASH
2214  012072  104401  012262                  TYPE    9$
2215  012076  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
2216  012102  005303                  6$:     DEC     R3              ;;BACKUP BY ONE
2217  012104  020327  012264                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
2218  012110  103434                          BLO     4$              ;;BR IF YES
2219  012112  111367  000144                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
2220  012116  104401  012262                  TYPE    9$              ;;GO TYPE
2221  012122  000746                          BR      2$              ;;GO READ ANOTHER CHAR.
2222  012124  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
2223  012126  001406                          BEQ     7$              ;;BR IF NO
2224  012130  112767  000134  000124        MOVB    #'\,9$          ;;TYPE A BACK SLASH
2225  012136  104401  012262                  TYPE    9$
2226  012142  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
2227  012144  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
2228  012150  001003                          BNE     8$              ;;BR IF NO
2229  012152  104401  012274                  TYPE    $CNTLU          ;;TYPE A CONTROL "U"
2230  012156  000726                          BR      1$              ;;GO START OVER
2231  012160  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A "↑R"?
2232  012164  001011                          BNE     3$              ;;BRANCH IF NO
2233  012166  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
2234  012170  104401  001253                  TYPE    ,$CRLF          ;;TYPE A "CR" & "LF"
2235  012174  104401  012264                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
2236  012200  000717                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
2237  012202  104401  001252          4$:     TYPE    ,$QUES          ;;TYPE A '?'
2238  012206  000712                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
2239  012210  111367  000046          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
2240  012214  104401  012262                  TYPE    9$
2241  012220  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
2242  012224  001305                          BNE     2$              ;;LOOP IF NOT RETURN
2243  012226  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
2244  012232  104401  001254                  TYPE    ,$LF            ;;TYPE A LINE FEED
2245  012236  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
2246  012240  012603                          MOV     (SP)+,R3        ;;RESTORE R3
2247  012242  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
2248  012244  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
2249  012252  012766  012264  000004          MOV     #$TTYIN,4(SP)
2250  012260  000002                          RTI                     ;;RETURN
2251  012262     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
2252  012263     000                          .BYTE   0               ;;TERMINATOR
2253  012264  000010          $TTYIN: .BLKB   8.                      ;;RESERVE 8 BYTES FOR TTY INPUT
2254  012274  052536  005015  000   $CNTLU: .ASCIZ  /↑U/<15><12>    ;;CONTROL "U"
2255  012301     136  006507  000012$CNTLG: .ASCIZ  /↑G/<15><12>    ;;CONTROL "G"
2256  012306  005015  053523  020122$MSWR:  .ASCIZ  <15><12>/SWR = /
2257  012314  020075     000
2258  012317     040  047040  053505$MNEW:  .ASCIZ  /  NEW = /
2259  012324  036440  000040
2260
```

```
                                .SBTTL  READ AN OCTAL NUMBER FROM THE TTY

                                ;;**********************************************************************
                                ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
                                ;*CHANGE IT TO BINARY.
                                ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
                                ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
                                ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED, THE COMPLETE NUMBER MUST
                                ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
                                ;*CALL:
                                ;*      RDOCT                   ;;READ AN OCTAL NUMBER
                                ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
                                ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
                                ;*

012330  011646          $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
012332  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
012340  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
012342  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
012344  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
012346  104407                1$:     RDLIN                   ;;READ AN ASCIZ LINE
012350  012600                        MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
012352  010067  000100                MOV     R0,5$           ;;AND SAVE IT
012356  005001                        CLR     R1              ;;CLEAR DATA WORD
012360  005002                        CLR     R2
012362  112046                2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
012364  001420                        BEQ     3$              ;;IF ZERO GET OUT
012366  122716  000060                CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
012372  003026                        BGT     4$              ;;IS AN OCTAL DIGIT
012374  122716  000067                CMPB    #'7,(SP)
012400  002423                        BLT     4$
012402  006301                        ASL     R1              ;;*2
012404  006102                        ROL     R2
012406  006301                        ASL     R1              ;;*4
012410  006102                        ROL     R2
012412  006301                        ASL     R1              ;;*8
012414  006102                        ROL     R2
012416  042716  177770                BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
012422  062601                        ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
012424  000756                        BR      2$              ;;LOOP
012426  005726                3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
012430  010166  000012                MOV     R1,12(SP)       ;;SAVE THE RESULT
012434  010267  000026                MOV     R2,$HIOCT
012440  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
012442  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
012444  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
012446  000002                        RTI                     ;;RETURN
012450  005726                4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
012452  105010                        CLRB    (R0)            ;;SET A TERMINATOR
012454  104401                        TYPE                    ;;TYPE UP THRU THE BAD CHAR.
012456  000000                5$:     .WORD   0
012460  104401  001252                TYPE    $QUES           ;;"?" "CR" & "LF"
012464  000730                        BR      1$              ;;TRY AGAIN
012466  000000                $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE

                                .SBTTL  TYPE ROUTINE
```

```
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
```

```
                                        $RDCHR  ;;CALL=RDCHR    TRAP+6(104406)  TTY TYPEIN CHARACTER ROUTINE
                                        $RDLIN  ;;CALL=RDLIN    TRAP+7(104407)  TTY TYPEIN STRING ROUTINE
                                        $RDOCT  ;;CALL=RDOCT    TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY
                                        $RDDEC  ;;CALL=RDDEC    TRAP+11(104411) READ A DECIMAL NUMBER FROM TTY

                                .SBTTL  POWER DOWN AND UP ROUTINES

                                ;;**********************************************************************
                                ;POWER DOWN ROUTINE
013146  012737  013312  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
013154  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
013162  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
013164  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
013166  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
013170  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
013172  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
013174  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
013176  017746  165736                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
013202  010667  000110                  MOV     SP,$SAVR6       ;;SAVE SP
013206  012737  013220  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
013214  000000                          HALT
013216  000776                          BR      .-2             ;;HANG UP

                                ;;**********************************************************************
                                ;POWER UP ROUTINE
013220  012737  013312  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
013226  016706  000064                  MOV     $SAVR6,SP       ;;GET SP
013232  005067  000060                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
013236  005267  000054          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
013242  001375                          BNE     1$              ;;OF  WORD
013244  012677  165670                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
013250  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
013252  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
013254  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
013256  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
013260  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
013262  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
013264  012737  013146  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
013272  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
013300  104401                          TYPE                    ;REPORT THE POWER FAILURE
013302  013320          $PWRMG: .WORD   $POWER                  ;;POWER FAIL MESSAGE POINTER
013304  012716                  MOV     (PC)+,(SP)              ;;RESTART AT RESTRT
013306  001772          $PWRAD: .WORD   RESTRT                  ;;RESTART ADDRESS
013310  000002                  RTI
013312  000000          $ILLUP: HALT                            ;;THE POWER UP SEQUENCE WAS STARTED
013314  000776                  BR      .-2                     ;; BEFORE THE POWER DOWN WAS COMPLETE
013316  000000          $SAVR6: 0                               ;;PUT THE SP HERE
013320  005015  047520  042527  $POWER: .ASCIZ  <15><12>"POWER"
013326  000122
                                .EVEN

                                ;;**********************************************************************
                                ;TRANSMIT INTERRUPT SERVICE ROUTINE FOR 256. BYTE BLOCK TRANSFERS
                                ;;**********************************************************************

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: 1233                    ;LOW LIMIT FOR RANDOM NO.
2623  013674  007622                  NUMTWO: 7622                    ;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     #DLBUF0,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     #DLBUF0,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     #DLBUF0,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$:     MOVB    R5,DLBUF0(R5)   ;LOAD ONE BYTE
2754  014412  005205                          INC     R5              ;INCREMENT BYTE
2755  014414  022705  000400                  CMP     #400,R5         ;DONE 000 THRU 377 ??
2756  014420  001372                          BNE     1$              ;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,$TMP7      ;START WITH A 377 BYTE
2762  014432  012705  021260                  MOV     #DLBUF0,R5      ;R5 POINTS TO OUTPUT BUFFER
2763  014436  116725  164556          1$:     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     #DLBUF0,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     #DLBUF0,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
2791  014552  012701  021660            MOV     #DLBUFI,R1      ;CHARACTER LENGTH(5, 6, 7, OR 8 BITS)
2792  014556  122221                1$:  CMPB    (R2)+,(R1)+     ;R1 POINTS TO WAS DATA IN RCVR. BUFFER
2793  014560  001004                     BNE     3$              ;DID S/B = WAS ??
2794  014562  022701  022260        2$:  CMP     #BUFEND,R1      ;BR IF NOT
2795  014566  001373                     BNE     1$              ;CHECKED ALL BYTES ??
2796  014570  000207                     RTS     PC              ;BR IF NOT
2797  014572  013767  177776  164402 3$:  MOV     @#PSW,$TMP0     ;RETURN TO CALLER
2798  014600  010667  164372            MOV     SP,$REG6        ;SAVE THE [PSW]
2799  014604  114204                     MOVB    -(R2),R4        ;SAVE THE [SP]
2800  014606  042704  177400            BIC     #177400,R4      ;GET THE S/B DATA
2801  014612  114103                     MOVB    -(R1),R3        ;CLEAR JUNK FROM HI BYTE
2802  014614  042703  177400            BIC     #177400,R3      ;GET THE WAS DATA
2803  014620  004767  177254            JSR     PC,SUERR1       ;CLEAR JUNK FROM HI BYTE
2804  014624  012767  014634  164412    MOV     #4$,$ESCAPE     ;GO SET UP ERROR INFO.
2805  014632  104003                     ERROR+3                 ;RETURN TO 4$ AFTER ERROR PRINT
2806  014634  005202                4$:  INC     R2              ;DATA COMPARE ERROR
2807  014636  005201                     INC     R1              ;REPOSITION BUFFER POINTERS
2808  014640  000750                     BR      2$              ;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
                                        ;TRANSMITTED IS DONE.  THE MASKING IS DONE AS A FUNCTION OF CHARACTER
                                        ;LENGTH WHICH CAN BE EITHER 5, 6, 7, OR 8 BITS.

014744  016767  164234  164260  UPMASK: MOV     $TMP1,$TMP14    ;PICK UP THE CHARACTER THAT WAS
                                                                ;SENT OUT FROM THE XMITTER
014752  005005                          CLR     R5              ;INITIIALIZE TABLE OFFSET
                                                                ;FOR PICKING UP MASK WORD
014754  022767  000010  164252          CMP     #8.,$TMP15      ;IS THE CHARACTER LENGTH 8 BITS?
014762  001423                          BEQ     2$              ;BRANCH IF IT IS
014764  062705  000002                  ADD     #2,R5           ;SET UP FOR NEXT MASK WORD
                                                                ;IT COULD BE THIS ONE
014770  022767  000007  164236          CMP     #7.,$TMP15      ;IS THE CHARACTER LENGTH 7 BITS?
014776  001410                          BEQ     1$              ;BRANCH IF IT IS
015000  062705  000002                  ADD     #2,R5           ;SET UP FOR NEXT MASK WORD
                                                                ;IT COULD BE THIS ONE
015004  022767  000006  164222          CMP     #6.,$TMP15      ;IS THE CHARACTER LENGTH 6 BITS?
015012  001402                          BEQ     1$              ;BRANCH IF IT IS
015014  062705  000002                  ADD     #2,R5           ;SET UP FOR NEXT MASK WORD
                                                                ;IT MUST BE THIS ONE!!!!
015020  016505  014734          1$:     MOV     CHARL(R5),R5    ;PICK UP THE MASK WORD
015024  005105                          COM     R5              ;FORM THE BITS THAT ARE TO BE MASKED
015026  140567  164200                  BICB    R5,$TMP14       ;MASK THE LOW BYTE
015032  000207                  2$:     RTS     PC              ;RETURN TO MAINLINE CODE

                                        ;ROUTINE TO SERVICE BUS ERROR TRAPS

015034  112767  000060  000632  BUSERR: MOVB    #60,EM4+46      ;SET UP ERROR MESSAGE
015042  112767  000060  000625          MOVB    #60,EM4+47
015050  112767  000064  000620          MOVB    #64,EM4+50
015056  000412                          BR      TRPCOM          ;GO SET UP AND REPORT BUS ERROR

                                        ;ROUTINE TO SERVICE RSVD INSTRUCTION TRAPS

015060  112767  000060  000606  RSVERR: MOVB    #60,EM4+46      ;SET UP ERROR MESSAGE
015066  112767  000061  000601          MOVB    #61,EM4+47
015074  112767  000060  000574          MOVB    #60,EM4+50
015102  000400                          BR      TRPCOM          ;GO SET UP AND REPORT RSVD INSTR. ERROR

                                        ;ROUTINE TO SET UP AND REPORT BUS ERROR AND RSVD INSTR ERRORS

015104  010667  164066          TRPCOM: MOV     SP,$REG6        ;SAVE THE TRAP SP
015110  116700  163766                  MOVB    $TSTNM,R0       ;PUT TEST NO. IN R0
015114  010067  164042                  MOV     R0,$REG0        ;SAVE TEST #
015120  016667  000002  164054          MOV     2(SP),$TMP0     ;SAVE THE ERROR PSW
015126  012767  015142  164110          MOV     #1$,$ESCAPE     ;GO TO 1$ AFTER ERROR PRINT
015134  011667  164040                  MOV     (SP),$REG7      ;SAVE THE ERROR PC
015140  104004                          ERROR+4                 ;REPORTED TRAP ERROR
015142  000137  001772          1$:     JMP     @#RESTRT        ;ATTEMPT TO RESTART THE PROGRAM
                                                                ;AND TRY AGAIN


                                        ;;*****************************************************************
                                        ;ERROR MESSAGE INFORMATION
                                        ;;*****************************************************************
```

```
                                        ;INFORMATION FOR ERROR MESSAGE 1
                                        
015146  046104  030461  051040  EM1:    .ASCIZ  'DL11 REGISTER REFERENCE CAUSED TIMEOUT'
015154  043505  051511  042524
015162  020122  042522  042506
015170  042522  041516  020105
015176  040503  051525  042105
015204  052040  046511  047505
015212  052123      000
015215      040 050050  024503  DH1:    .ASCIZ  ' (PC)    (PS)    (SP)    TEST   DEVADR  REGADR'
015222  020040  020040  050050
015230  024523  020040  020040
015236  051450  024520  020040
015244  020040  042524  052123
015252  020040  042040  053105
015260  042101  020122  051040
015266  043505  042101  000122
                        .EVEN
015274  001116  001202  001176  DT1:    .WORD   $ERRPC,$TMPO,$REG6,$REG0,$REG1,$REG2,0
015302  001162  001164  001166
015310  000000
                                        
                                        ;INFORMATION FOR ERROR MESSAGE 2
                                        
015312  046104  030461  051040  EM2:    .ASCIZ  'DL11 REGISTER ERROR'
015320  043505  051511  042524
015326  020122  051105  047522
015334  000122
015336  024040  041520  020051  DH2:    .ASCIZ  ' (PC)    (PS)    (SP)    TEST   DEVADR  REGADR   WAS     S/B'
015344  020040  024040  051520
015352  020051  020040  024040
015360  050123  020051  020040
015366  052040  051505  020124
015374  020040  042504  040526
015402  051104  020040  042522
015410  040507  051104  020040
015416  053440  051501  020040
015424  020040  051440  041057
015432      000
        015434          .EVEN
015434  001116  001202  001176  DT2:    .WORD   $ERRPC,$TMPO,$REG6,$REG0,$REG1,$REG2,$REG3,$REG4,0
015442  001162  001164  001166
015450  001170  001172  000000
                                        
                                        ;INFORMATION FOR MESSAGE 3
                                        
015456  046104  030461  042040  EM3:    .ASCIZ  'DL11 DATA COMPARE ERROR'
015464  052101  020101  047503
015472  050115  051101  020105
015500  051105  047522  000122
015506  024040  041520  020051  DH3:    .ASCIZ  ' (PC)    (PS)    (SP)    TEST   WASADR  SHBADR    WAS     S/B'
015514  020040  024040  051520
015522  020051  020040  024040
015530  050123  020051  020040
015536  052040  051505  020124
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   $ERRPC,$TMPO,$REG6,$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  DM4:    .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   $REG7,$TMPO,$REG6,$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  DM5:    .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   $ERRPC,$TMPO,$REG6,$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  DM6:    .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
```

```
                                                ;MESSAGES SEEKING USER RESPONSE
                                                ;
017233  000015  053412  040510  LENGTH: .ASCIZ  <15><12>'WHAT IS THE CHARACTER LENGTH (5,6,7 OR 8 BITS)?'
017240  020124  051511  052040
017246  042510  041440  040510
017254  040522  052103  051105
017262  046040  047105  052107
017270  020110  032450  033054
017276  033454  047440  020122
017304  020070  044502  051524
017312  037451  000000
017315  000015  042012  020117  DEFAULT:        .ASCII  <15><12>'DO YOU WISH TO TEST OTHER THAN THE'
017322  047531  020125  044527
017330  044123  052040  020117
017336  042524  052123  047440
017344  044124  051105  052040
017352  040510  020116  044124
017360  000105
017361  000015  042012  043105          .ASCIZ  <15><12>'DEFAULT DEVICE (1/0 = YES/NO)?'
017366  052501  052114  042040
017374  053105  041511  020105
017402  030450  030057  036440
017410  054440  051505  047057
017416  024517  000077
017422  005015  044127  052101  MFIRSTD:.ASCIZ  <15><12>'WHAT IS THE 1ST RECEIVER STATUS REGISTER ADDRESS?  '
017430  044440  020123  044124
017436  020105  051461  020124
017444  042522  042503  053111
017452  051105  051440  040524
017460  052524  020123  042522
017466  044507  052123  051105
017474  040440  042104  042522
017502  051523  020077  000040
017510  005015  044127  052101  MVECT:  .ASCIZ  <15><12>'WHAT IS THE 1ST RECEIVER''S VECTOR ADDRESS?  '
017516  044440  020123  044124
017524  020105  051461  020124
017532  042522  042503  053111
017540  051105  020123  042526
017546  052103  051117  040440
017554  042104  042522  051523
017562  020077  000040
017566  005015  047504  054440  MULDEV: .ASCIZ  <15><12>'DO YOU WANT TO TEST MULTIPLE DEVICES 1/0=YES/NO?  '
017574  052517  053440  047101
017602  020124  047524  052040
017610  051505  020124  052515
017616  052114  050111  042514
017624  042040  053105  041511
017632  051505  030440  030057
017640  054475  051505  047057
017646  037517  020040  000000
017653  000015  053412  040510  MLASTD: .ASCIZ  <15><12>'WHAT IS THE STATUS REGISTER ADDRESS OF THE LAST RECEIVER?  '
017660  020124  051511  052040
017666  042510  051440  040524
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          .ASCIZ  <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: .ASCIZ  <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: .ASCIZ  <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
BASEIV  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  1223
                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*  713*  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*  2875*  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#
RESTBI  001772           485#   1785    1854    2527    2894
RESVEC= 000010           156#    488*    489*
RETRY   002346           526     528     596#
RETRY0  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#
SELOLY  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
SUERT1  014132           735     750     765     780    2691#
SUERT2  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#
TYPDS  = 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    1593    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
                           2229    2234    2235    2237    2240    2244    2309    2311    2349    2442    2444    2478#   2524
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#
XFLGO   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#
```

```
SAUTOB  001134            192#
SBDADR  001122            187#
SBDDAT  001126            189#
SBELL   001246            233#  1942  1963
SCHARC  012704           2351#  2361#  2368  2377#  2382#
SCKSWR= ****** U         2485
SCNTAG  001100            175#   395   396   404   410   411
SCN1  = 000010            207#   208#  209#  210#  211#  212#  213#  214#  215#
SCN2  = 000020            207#   208#  209#  210#  211#  212#  213#  214#  215#
SCN3  = 000010            205#   207
SCN4  = 000020            215#   216#  217#  218#  219#  220#  221#  222#  223#  224#  225#  226#  227#
                          228#   229#  230#  231#
SCNTLG  012301           2255#
SCNTLU  012274           2229   2254#
SCRLF   001253            235#  1950  1963  1972  1991  1996  2000  2234  2254  2314  2350  2385  2446
SDBLK   011700           2114   2148  2156#
SDOAGN  010452           1837   1846  1852#
SDTBL   011670           2117   2152#
SENDAD  010442           1848#
SENDCT  010410           1839#
SENDMG  010461           1841   1856#
SENULL  010456           1844   1855#
SEOP    010144           1307   1317  1325  1333  1343  1767#
SEOPCT  010402           1836#  1840
SERFLG  001103            178#  1864  1892  1894  1900*  1921  1937*  1963
SERMAX  001115            184#   412*  1894  1916*  1921
SERROR  010746            404   1357  1440  1525  1634  1936#
SERRPC  001116            185#  1944*  1945*  1946  1963  1978  2920  2942  2964  3007  3020  3032  3045
SERRTB  001310            312#  1986
SERRTY  011102           1949   1971#
SERTTL  001112            182#  1943*  1963
SESCAP  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*
SFILLC  001156            203#  2354  2385
SFILLS  001155            202#  2385
SGDADR  001120            186#
SGDDAT  001124            188#
SGET42  010432           1845#
SGTSWR= ****** U         2484
SHD   = 000001             16    17
SHIOCT  012466           2302*  2313#
SICNT   001104            179#  1907*  1908  1910*  1920
SILLUP  013312           2494   2510  2529#
SINTAG  001135            193#
SITEMB  001114            183#  1946*  1963  1975
SLF     001254            236#  1963  2244  2254  2314  2385  2446
SLPADR  001106            180#   413*  1898*  1913*  1918  1920
SLPERR  001110            181#   414*  1898  1914*  1920  1957
SMAIL = ****** U          431   1913  1952  2338
SMNEW   012317           2258#
SMSWR   012306           2256#
SMXCNT  010744           1911   1920#
SNULL   001154            201#  2356  2385
SNWTST= 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#
$PWRON  013146           408  2494#  2522
$PWRNG  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#
$$GET4= 000000         1847#
$OFILL  011461         2038*  2042*  2052   2087#
$$OCAT= ****** 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#
SSCMRE   168#   207   208   209   210   211   212   213   214
SSCMTM   168#   215   216   217   218   219   220   221   222   223   224   225   226   227   228
          229   230
SSESCA   167#
SSNEWT   167#   726   741   756   771   786   796   806   823   852   869   927   958   994  1029
         1048  1084  1105  1118  1175  1232  1289
SSSET   2470#  2479  2480  2481  2482  2485  2486  2487  2488
SSSKIP   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
.$CATC      4#     36
.$CMTA      4#    168
.$EOP       4#   1759
.$ERRO      4#   1922
.$ERRT      4#   1964
.$POWE      4#   2490
.$RDDE      4#   2386
.$RDOC      4#   2261
.$READ      4#   2158
.$SCOP      4#   1859
.$TRAP      4#   2447
.$TYPD      4#   2090
.$TYPE      4#   2315
.$TYPO      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)
```