DL11-C,-D,-E

DL11-C,D,E OFLINE TST
CZDLCC0

AH-8525C-MC

FICHE 1 OF 1

JAN 1980
digital
MADE IN USA

```
.REM@
```

IDENTIFICATION

PRODUCT CODE: AC-8524C-MC

PRODUCT NAME: CZDLCC0 DL11-C,D,E OFLNE TST

PRODUCT DATE: JULY 1979

MAINTAINER: DIAGNOSTIC 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:

210 - SELECTS PROGRAM #2
220 - SELECTS PROGRAM #3

```
                                        230 - SELECTS PROGRAM #4
                                        240 - 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.
                6.2     FOR USE WITH PROCESSOR THAT DOES NOT HAVE A KEYBOARD.

                        IF A HARDWARE SWITCH REGISTER DOES NOT EXIST, THE PROGRAM WILL
                        USE THE CONTENTS OF LOCATION 176 AS THE VALUE OF THE SWITCHES.
                        THE PROGRAM WILL PRINT OUT THE PRESENT CONTENTS OF THE
                        SOFTWARE SWITCH REGISTER WHEN THE PROGRAM IS STARTED.  IT
                        WILL THEN ASK FOR THE NEW CONTENTS TO BE INPUT TO THE SOFTWARE
                        SWITCH REGISTER.  TYPE CARRIAGE RETURN TO FINISH INPUT.



                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

PAGE 6

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

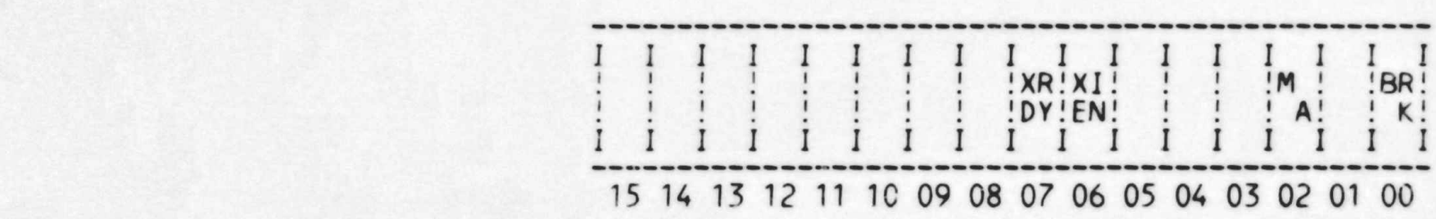
176170

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

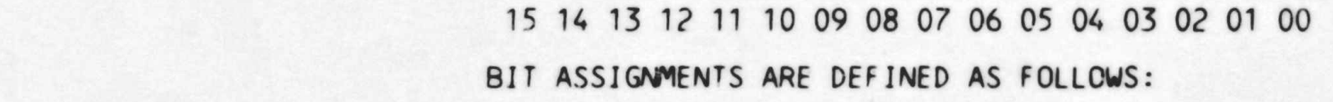
1. WHEN SET, DISABLES SERIAL LINE INPUT TO RECEIVER &

```
                                                    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 !!
```

PAGE 10

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 '$ERRTYP' ROUTINE SHOWN BELOW (PRESUMES 'HALT' ON ERROR SW<15> NOT SET).

12.5 $ERRTYP

THIS ROUTINE IS SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE.

THIS ROUTINE HANDLES THE INFORMATION FOR ERROR MESSAGE TYPEOUTS AS FOLLOWS:

THIS ROUTINE USES THE 'ITEM CONTROL BYTE' ($ITEMB) TO DETERMINE WHICH ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE' ($ERRTB) THE ADDRESSES OF WHERE THE INFORMATION, FOR PRINTOUT, IS STORED; AND CAUSES THE APPROPRIATE INFORMATION CONCERNING THE ERROR TO BE PRINTED OUT.

NOTE: 1. THE VARIABLE '$ITEMB' IS SUPPLIED BY .$CMTAG, A 'SYSMAC' UTILITY PACKAGE ROUTINE.

2. THE 1ST ADDRESS '$ERRTB' FOR LOCATION OF 'ERROR TABLE' INFORMATION IS ALSO SUPPLIED BY .$CMTAG.

3. IF THE '$ITEMB' VALUE IS ZERO(0), THEN ONLY A PROGRAM COUNTER (PC) IS PRINTED OUT. IT HAS NO LABEL, IT IS A PURE NUMBER.

12.6 $TYPOC

THIS ROUTINE IS SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE. THIS ROUTINE IS USED FOR ALL OCTAL TYPEOUTS (16 BIT VALUES) THROUGHOUT THE PROGRAM.

12.7 $TYPDS

THIS ROUTINE IS SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE. THIS ROUTINE IS USED TO TYPE A DECIMAL VALUE AT THE END OF A PASS OF THE PROGRAM OF THE FORM 'END PASS # XXX' WHERE 'XXX' IS THE DECIMAL VALUE.

12.8 $RDCHR, $RDLIN, $RDOCT

THESE ROUTINES ARE SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE. THEIR USES ARE AS FOLLOWS:

A. $RDCHR - HANDLES A SINGLE CHARACTER COMING IN FROM THE TTY. THE CHARACTER IS PLACED ON TOP OF THE STACK FOR FUTURE USE.

B. $RDLIN - HANDLES A STRING OF CHARACTERS COMING IN FROM THE TTY. THE ADDRESS OF THE 1ST CHARACTER IS PLACED ON TOP OF THE STACK FOR FUTURE USE.

C. $RDOCT - HANDLES AN OCTAL NUMBER COMING IN FROM THE

$RDDEC 104420 TTY DECIMAL # INPUT TTY. LOW ORDER BITS ARE STORED ON TOP OF THE STACK; HIGH ORDER BITS ARE STORED IN LOCATION $HIOCT. $HIOCT IS SUPPLIED BY .$CMTAG, A 'SYSMAC' PACKAGE UTILITY ROUTINE.

12.9 $TYPE

THIS ROUTINE IS SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE. THIS ROUTINE IS USED TO TYPE ASCII MESSAGES (WHICH MUST TERMINATE WITH A 0 BYTE) AS WELL AS ALL OTHER FORMS OF TYPED INFORMATION. THE ROUTINE IS ALSO RESPONSIBLE FOR INSERTING A NUMBER OF FILL CHARACTERS AFTER A LINE FEED.

NOTE: 1. $NULL CONTAINS THE CHARACTER TO BE USED AS FILL.

2. $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQ'D.

3. $FILLC CONTAINS THE CHARACTER TO FILL AFTER.

4. THE ABOVE THREE(3) VARIABLES ARE SUPPLIED BY .$CMTAG, A 'SYSMAC' PACKAGE UTILITY ROUTINE.

12.10 $TRAP, $TRPAD

THESE ROUTINES ARE SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE. THE '$TRAP' ROUTINE WILL STRIP OFF THE LOWER BYTE OF A TRAP INSTRUCTION AND USE IT TO INDEX THRU THE TRAP TABLE ($TRPAD) FOR THE STARTING ADDRESS OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL THEN TRANSFER PROGRAM CONTROL TO THAT ROUTINE.

THE FOLLOWING TABLE DEFINES ALL ROUTINES IN THE PROGRAM CALLED BY A 'TRAP' INSTRUCTION BY SHOWING THEIR 'TRAP' EQUIVALENCES -

$TYPE	104400	TTY TYPEOUT ROUTINE
$TYPOC	104402	TYPE OCTAL # (WITH LEADING ZEROS)
$TYPOS	104404	TYPE OCTAL # (NO LEADING ZEROS)
$TYPON	104406	TYPE OCTAL # (PER LAST CHARACTER METHOD)
$TYPDS	104410	TYPE DECIMAL # (WITH SIGN)
$RDCHR	104412	TTY CHARACTER INPUT
$RDLIN	104414	TTY STRING INPUT
$RDOCT	104416	TTY OCTAL # INPUT

12.11 $PWRDN, $PWRUP

THESE ROUTINES ARE SUPPLIED BY MAINDEC-11-DZQAC-C3, THE PDP-11 MAINDEC 'SYSMAC' UTILITY PACKAGE. THESE ROUTINES HANDLE THE 'POWER DOWN AND UP' SEQUENCE. THE PROGRAM MAY BE POWER FAILED WHEN RUNNING; HOWEVER, USE CAUTION IN TURNING POWER OFF/ON WHILE THE POWER FAIL MESSAGE IS BEING TYPED - IT MAY CAUSE STACK OVERFLOW.

NOTE

WHEN POWER RETURNS THE PROGRAM WILL AUTOMATICALLY START ITSELF OVER AT THE BEGINNING.

12.12 XINT, RINT

XINT - THIS IS THE TRANSMITTER INTERRUPT SERVICE ROUTINE FOR 256(10) BYTE BLOCK TRANSFERS.

RINT - THIS IS THE RECEIVER INTERRUPT SERVICE ROUTINE FOR 256(10) BYTE BLOCK TRANSFERS.

12.13 DELAY, STALL, DATCHK, TIMERX, TIMETX

THESE ROUTINES ARE ALL USED BY PROGRAMS 2, 3, 4 AND 5. PROGRAMS 2 THROUGH 5 ARE THE 'SPECIAL' USER INTERACTION ROUTINES WHICH WILL BE DEFINED LATER IN THIS DOCUMENT. THE ABOVE ROUTINE USES ARE AS FOLLOWS:

A. DELAY - THIS ROUTINE IS USED BY ALL THE UTILITY PROGRAMS TO WAIT A NO. OF MILLISECONDS BETWEEN CHARACTER TRANSFERS AS SPECIFIED BY THE USER.

B. STALL - THIS ROUTINE IS USED BY PROGRAM #4 AND WILL ALLOW A RANDOM NO. OF MILLISECONDS TO TRANSPIRE BEFORE A TRANSMISSION OF A CHARACTER. THIS ROUTINE IS ACTIVATED BASED ON USER RESPONSE.

C. DATCHK - THIS ROUTINE IS USED BY PROGRAM #4 AND WILL CHECK FOR CORRECT EXPECTED AND RECEIVED DATA AFTER CHARACTER TRANSMISSION AS WELL AS ANY ERROR BIT CONDITIONS.

D. TIMERX + TIMETX -
THESE TWO(2) ROUTINES ARE USED BY PROGRAM #4 TO VERIFY THE 'DONE' BIT AFTER BOTH TRANSMITTER AND RECEIVER OPERATIONS.

12.14 SUERR1, SUER2

THESE TWO(2) ROUTINES ARE USED THROUGHOUT THE PROGRAM TO SET UP THE ERROR INFORMATION FOR 'ERROR REPORTING' BEFORE THE 'ERROR REPORT' CALL IS MADE. 'ERROR REPORT' CALLS APPEAR THROUGHOUT THE PROGRAM IN THE FORM 'ERROR + XX'' WHERE 'XX' INDICATES THE PARTICULAR ERROR TABLE (ERRTB:) ENTRY USED BY THE ERROR SERVICE ROUTINE.

12.15 PRIME

THIS ROUTINE IS USED TO SET UP THE DATA BUFFERS AN THE DEVICE UNDER TEST FOR EACH 256(10) BYTE BLOCK TRANSFER.

12.16 CLDLBF

THIS ROUTINE IS USED IN CONJUNCTION WITH ROUTINE 'PRIME' TO CLEAR INPUT AND OUTPUT BUFFERS BEFORE DATA TRANSFERS.

12.17 LDOUT1, LDOUT2, LDOUT3, LDOUT4

THE ROUTINES ARE ALL USED FOR SET UP AND LOADING PURPOSES AS FOLLOWS:

A. LDOUT1 - IS CALLED TO SET UP THE 'NULL-DEL-NULL' PATTERN
B. LDOUT2 - IS CALLED TO LOAD AN ASCENDING BINARY COUNT PATTERN
C. LDOUT3 - IS CALLED TO LOAD A DESCENDING BINARY COUNT PATTERN
D. LDOUT4 - IS CALLED TO LOAD A COMPLEMENTING WORSE CASE PATTERN

12.18 CHKDAT

THIS ROUTINE IS USED TO CHECK FOR DATA COMPARE ERRORS IN 256(10) BYTE BLOCK TRANSFERS.

12.19 BUSERR, RSVERR

THESE TWO(2) ROUTINES ARE USED TO SERVICE 'UNEXPECTED' BUS ERROR AND RESERVED INSTRUCTION TRAPS, RESPECTIVELY.

12.20 TRPCOM

THIS ROUTINE IS USED TO SET UP AND REPORT THE INFORMATION CONCERNING 'UNEXPECTED' BUS ERROR AND RESERVED INSTRUCTION TRAPS. THIS ROUTINE IS USED IN CONJUNCTION WITH ROUTINES 'BUSERR' AND 'RSVERR' DESCRIBED ABOVE.

13.0 MISCELLANEOUS

A. THE STACK POINTER IS INITIALLY SET TO 1100.

B. THE PARITY BIT IS NOT COVERED.

14.0 USER SELECTION PROGRAMS

14.1 PROGRAM #2 DESCRIPTION

THIS UTILITY PROGRAM WILL ALLOW THE FOLLOWING:

A. SELECTION OF TRANSMITTER DATA BUFFER

B. SELECTION OF A CHARACTER FOR CONTINUOUS TRANSFER

C. SELECTION OF AN EXPIRATION TIME IN MILLISECONDS BETWEEN EACH TRANSMITTER DATA BUFFER CHARACTER TRANSFER

D. A TIGHT SCOPE LOOP LOCK ON A SPECIFIC CHARACTER'

THE PROGRAM RELIES ON USER RESPONSE (VIA TTY) TO SPECIFIC QUESTIONS AS DESCRIBED BELOW:

A. WHAT IS THE TRANSMITTER DATA BUFFER ADDRESS?

THE USER SHOULD RESPOND BY TYPING AN ADDRESS IN THE RANGE 175616 TO 176176 AND FOLLOW IT WITH A 'CARRIAGE RETURN' AT WHICH TIME THE PROGRAM WILL VALIDATE WHAT WAS TYPED TO SEE IF -

1. THE VALUE TYPED IS WITHIN THE CORRECT RANGE

2. THE VALUE TYPED IS AN 'EVEN' ADDRESS, SO AS NOT TO CAUSE A 'BUS TIMEOUT' WHEN REFERENCED, AND

3. THEN CHECKS TO SEE IF THE DEVICE ASSOCIATED WITH THE VALUE TYPED IS INDEED PRESENT

NOTE

IF EITHER OF THE THREE(3) ABOVE CONDITIONS ARE NOT MET THE PROGRAM WILL TYPE A QUESTION MARK (?), REITERATE THE INITIAL QUESTION, AND WAIT FOR A 'NEW' USER RESPONSE.

B. WHAT IS THE CHARACTER TO BE TRANSMITTED (OCTAL ASCII E.G., A=101)?

THE USER SHOULD RESPOND BY TYPING AN OCTAL ASCII VALUE, FOR THE CHARACTER DESIRED AND FOLLOW IT WITH A 'CARRIAGE RETURN'.

C. WHAT IS THE DESIRED MSEC. DELAY (OCTAL E.G., 10=8(10))?

THE USER SHOULD RESPOND BY TYPING AN OCTAL VALUE FOR THE DESIRED NO. OF MSEC. DELAY AND FOLLOW IT WITH A 'CARRIAGE RETURN'.

E.G. - IF USER DESIRED 16 MSEC. DELAY BETWEEN EACH CHARACTER TRANSFER HE SHOULD TYPE '20'.

D. AT THIS POINT THE PROGRAM WILL LOOP CONTINUOUSLY SENDING THE CHARACTER SPECIFIED, WITH THE DESIRED MSEC. DELAY BETWEEN EACH CHARACTER TRANSMISSION.

14.2 PROGRAM #3 DESCRIPTION

THIS UTILITY PROGRAM WILL ALLOW THE FOLLOWING:

A. SELECTION OF TRANSMITTER DATA BUFFER

B. SELECTION OF A CHARACTER FOR CONTINUOUS TRANSFER IN MAINTENANCE MODE.

C. SELECTION OF AN EXPIRATION TIME IN MILLISECONDS BETWEEN EACH TRANSMITTER DATA BUFFER CHARACTER TRANSFER

D. A TIGHT SCOPE LOOP LOCK ON A SPECIFIC CHARACTER

THE PROGRAM RELIES ON USER RESPONSE (VIA TTY) TO SPECIFIC QUESTIONS AS DESCRIBED BELOW:

A. WHAT IS THE TRANSMITTER DATA BUFFER ADDRESS?

THE USER SHOULD RESPOND BY TYPING AN ADDRESS IN THE RANGE 175616 TO 176176 AND FOLLOW IT WITH A 'CARRIAGE RETURN' AT WHICH TIME THE PROGRAM WILL VALIDATE WHAT WAS TYPED TO SEE IF -

1. THE VALUE TYPED IS WITHIN THE CORRECT RANGE

2. THE VALUE TYPED IS AN 'EVEN' ADDRESS, SO AS NOT TO CAUSE A 'BUS TIMEOUT' WHEN REFERENCED, AND

3. THEN CHECKS TO SEE IF THE DEVICE ASSOCIATED WITH THE VALUE TYPED IS INDEED PRESENT

NOTE

IF EITHER OF THE THREE(3) ABOVE CONDITIONS ARE NOT MET THE PROGRAM WILL TYPE A QUESTION MARK (?), REITERATE THE INITIAL QUESTION, AND WAIT FOR A 'NEW' USER RESPONSE.

B. WHAT IS THE CHARACTER TO BE TRANSMITTED (OCTAL ASCII E.G., A=101)?

THE USER SHOULD RESPOND BY TYPING AN OCTAL ASCII VALUE, FOR THE CHARACTER DESIRED AND FOLLOW IT WITH A 'CARRIAGE RETURN'.

C. WHAT IS THE DESIRED MSEC. DELAY (OCTAL E.G., 10=8(10))?

THE USER SHOULD RESPOND BY TYPING AN OCTAL VALUE FOR THE DESIRED NO. OF MSEC. DELAY AND FOLLOW IT WITH A 'CARRIAGE RETURN'.

E.G. - IF USER DESIRED 16 MSEC. DELAY BETWEEN EACH CHARACTER TRANSFER HE SHOULD TYPE '20'.

D. AT THIS POINT THE PROGRAM WILL LOOP CONTINUOUSLY SENDING THE CHARACTER SPECIFIED, WITH THE DESIRED MSEC. DELAY

BETWEEN EACH CHARACTER TRANSMISSION.

14.3 PROGRAM #4 DESCRIPTION

THIS UTILITY PROGRAM WILL ALLOW THE FOLLOWING:

A. SELECTION OF A TRANSMITTER DATA BUFFER

B. SELECTION OF A SINGLE CHARACTER TO BE SENT, RECEIVED AND CHECKED WITH MAINTENANCE BIT SET.

THE PROGRAM RELIES ON USER RESPONSE (VIA TTY) TO SPECIFIC QUESTIONS AS DESCRIBED BELOW:

A. WHAT IS THE TRANSMITTER DATA BUFFER ADDRESS?

THE USER SHOULD RESPOND BY TYPING AN ADDRESS IN THE RANGE 175616 TO 176176 AND FOLLOW IT WITH A 'CARRIAGE RETURN' AT WHICH TIME THE PROGRAM WILL VALIDATE WHAT WAS TYPED TO SEE IF -

1. THE VALUE TYPED IS WITHIN THE CORRECT RANGE

2. THE VALUE TYPED IS AN 'EVEN' ADDRESS, SO AS NOT TO CAUSE A 'BUS TIMEOUT' WHEN REFERENCED, AND

3. THEN CHECKS TO SEE IF THE DEVICE ASSOCIATED WITH THE VALUE TYPED IS INDEED PRESENT.

NOTE

IF EITHER OF THE THREE(3) ABOVE CONDITIONS ARE NOT MET THE PROGRAM WILL TYPE A QUESTION MARK (?), REITERATE THE INITIAL QUESTION, AND WAIT FOR A 'NEW' USER RESPONSE.

B. IS A RANDOM WAIT TIME (MSEC.) DESIRED 1/0=YES/NO?

THE USER SHOULD RESPOND AS ASKED AND FOLLOW IT WITH A 'CARRIAGE RETURN'.

C. WHAT IS THE CHARACTER TO BE TRANSMITTED (OCTAL ASCII E.G. A=101)?

THE USER SHOULD RESPOND BY TYPING AN OCTAL ASCII VALUE, FOR THE CHARACTER DESIRED AND FOLLOW IT WITH A 'CARRIAGE RETURN'.

D. AT THIS POINT THE PROGRAM WILL LOOP CONTINUOUSLY SENDING THE CHARACTER SPECIFIED, WITH A RANDOM MSEC. DELAY BETWEEN EACH CHARACTER TRANSMISSION. BETWEEN EACH TRANSMISSION, 'RCVR' & 'XMITTER' DONE BITS WILL BE

VERIFIED, AS WELL AS CHECKS FOR CORRECT DATA AND ANY ERROR BIT CONDITIONS.

NOTE

IF USER RESPONSE TO ITEM B. (DIRECTLY ABOVE) WAS A '0' OR A PLAIN 'CARRIAGE RETURN' THEN THERE IS NO DELAY BETWEEN CHARACTER TRANSMISSIONS.

14.4 PROGRAM #5 DESCRIPTION

THIS UTILITY PROGRAM WILL ALLOW USER PARAMETERS FOR RUNNING A BINARY COUNT IN MAINTENANCE MODE.

THE PROGRAM RELIES ON USER RESPONSE (VIA TTY) TO SPECIFIC QUESTIONS AS DESCRIBED BELOW:

A. WHAT IS THE TRANSMITTER DATA BUFFER ADDRESS?

THE USER SHOULD RESPOND BY TYPING AN ADDRESS IN THE RANGE 175616 TO 176176 AND FOLLOW IT WITH A 'CARRIAGE RETURN' AT WHICH TIME THE PROGRAM WILL VALIDATE WHAT WAS TYPED TO SEE IF -

1. THE VALUE TYPED IS WITHIN THE CORRECT RANGE

2. THE VALUE TYPED IS AN 'EVEN' ADDRESS, SO AS NOT TO CAUSE A 'BUS TIMEOUT' WHEN REFERENCED, AND

3. THEN CHECKS TO SEE IF THE DEVICE ASSOCIATED WITH THE VALUE TYPED IS INDEED PRESENT.

NOTE

IF EITHER OF THE THREE(3) ABOVE CONDITIONS ARE NOT MET THE PROGRAM WILL TYPE A QUESTION MARK (?), REITERATE THE INITIAL QUESTION, AND WAIT FOR A 'NEW' USER RESPONSE.

B. IS A RANDOM WAIT TIME (MSEC.) DESIRED 1/0=YES/NO?

THE USER SHOULD RESPOND AS ASKED AND FOLLOW IT WITH A 'CARRIAGE RETURN'.

C. AT THIS POINT THE PROGRAM WILL LOOP CONTINUOUSLY SENDING BINARY CHARACTERS, WITH A RANDOM MSEC. DELAY BETWEEN EACH CHARACTER TRANSMISSION. BETWEEN EACH TRANSMISSION, 'RCVR' & 'XMITTER' DONE BITS WILL BE VERIFIED, AS WELL AS CHECKS FOR CORRECT DATA AND ANY ERROR BIT CONDITIONS.

NOTE

IF USER RESPONSE TO ITEM B. (DIRECTLY

```
                                        ABOVE)  WAS  A  '0' OR A PLAIN 'CARRIAGE
                                        RETURN' THEN THERE IS NO  DELAY  BETWEEN
                                        CHARACTER TRANSMISSIONS.



                        15.0    PROGRAM FUNCTIONAL FLOW CHARTS


                        16.0    PROGRAM LISTING

                        17.0    ECO TABLE

                                CHGC1 - .SETUP MACRO EXPANDED TO INCLUDE ''SOFTSWR''.
                                CHGC2 - GETSWR MACRO ADDED AFTER SETUP.
                                CHGC3 - JSR PC,$TKINT FOLLOWS GETSWR.
                                CHGC4 - .$READ ARGUMENTS EXPANDED TO READ ''.$READ ,X,8.,200''
                                CHGC5 - MODIFIED PROGRAM START SO THAT ALL STARTS RUN
                                        THROUGH A COMMON SOFTSWR ROUTINE AND VECTOR INIT.
                                CHGC6 - ADDED DELAY TO TEST 10.

                        .ENDR @


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

                        .TITLE  CZDLCCO DL11-C,D,E OFLNE TST
                        ;*COPYRIGHT (C) 1979
                        ;*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

                 000174                 .=174
000174  000000                 DISPREG:  .WORD 0          ;SOFTWARE DISPLAY REGISTER
000176  000000                 SWREG:    .WORD            ;SOFTWARE SWITCH REGISTER
000200  012767  001734  001242          MOV     #PRG1,STAD     ;ADDRESS OF USER PROGRAM NO. 1
000206  000417                         BR      STCONT
000210  012767  006424  001232          MOV     #PRG2,STAD     ;ADDRESS OF USER PROGRAM NO.2
000216  000413                         BR      STCONT
000220  012767  006632  001222          MOV     #PRG3,STAD     ;ADDRESS OF USER PROGRAM NO. 3
000226  000407                         BR      STCONT
000230  012767  007050  001212          MOV     #PRG4,STAD     ;ADDRESS OF USER PROGRAM NO. 4
000236  000403                         BR      STCONT
000240  012767  007410  001202          MOV     #PRG5,STAD     ;ADDRESS OF USER PROGRAM NO. 5
000246  000137  001452          STCONT: JMP     @#BEGIN        ;JUMP TO COMMON START

                 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  IOT,SCOPE          ;;BASIC DEFINITION OF SCOPE CALL

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

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

```
1280
1281                                ;*PRIORITY LEVEL DEFINITIONS
1282            000000              PR0=    0                   ;;PRIORITY LEVEL 0
1283            000040              PR1=    40                  ;;PRIORITY LEVEL 1
1284            000100              PR2=    100                 ;;PRIORITY LEVEL 2
1285            000140              PR3=    140                 ;;PRIORITY LEVEL 3
1286            000200              PR4=    200                 ;;PRIORITY LEVEL 4
1287            000240              PR5=    240                 ;;PRIORITY LEVEL 5
1288            000300              PR6=    300                 ;;PRIORITY LEVEL 6
1289            000340              PR7=    340                 ;;PRIORITY LEVEL 7
1290
1291                                ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
1292            100000              SW15=   100000
1293            040000              SW14=   40000
1294            020000              SW13=   20000
1295            010000              SW12=   10000
1296            004000              SW11=   4000
1297            002000              SW10=   2000
1298            001000              SW09=   1000
1299            000400              SW08=   400
1300            000200              SW07=   200
1301            000100              SW06=   100
1302            000040              SW05=   40
1303            000020              SW04=   20
1304            000010              SW03=   10
1305            000004              SW02=   4
1306            000002              SW01=   2
1307            000001              SW00=   1
1308                                .EQUIV  SW09,SW9
1309                                .EQUIV  SW08,SW8
1310                                .EQUIV  SW07,SW7
1311                                .EQUIV  SW06,SW6
1312                                .EQUIV  SW05,SW5
1313                                .EQUIV  SW04,SW4
1314                                .EQUIV  SW03,SW3
1315                                .EQUIV  SW02,SW2
1316                                .EQUIV  SW01,SW1
1317                                .EQUIV  SW00,SW0
1318
1319                                ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
1320            100000              BIT15=  100000
1321            040000              BIT14=  40000
1322            020000              BIT13=  20000
1323            010000              BIT12=  10000
1324            004000              BIT11=  4000
1325            002000              BIT10=  2000
1326            001000              BIT09=  1000
1327            000400              BIT08=  400
1328            000200              BIT07=  200
1329            000100              BIT06=  100
1330            000040              BIT05=  40
1331            000020              BIT04=  20
1332            000010              BIT03=  10
1333            000004              BIT02=  4
1334            000002              BIT01=  2
1335            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
```

```
1417  001224  000000                  $TMP11: .WORD   0               ;;USER DEFINED
1418  001226  000000                  $TMP12: .WORD   0               ;;USER DEFINED
1419  001230  000000                  $TMP13: .WORD   0               ;;USER DEFINED
1420  001232  000000                  $TMP14: .WORD   0               ;;USER DEFINED
1421  001234  000000                  $TMP15: .WORD   0               ;;USER DEFINED
1422  001236  000000                  $TMP16: .WORD   0               ;;USER DEFINED
1423  001240  000000                  $TMP17: .WORD   0               ;;USER DEFINED
1424  001242  000000                  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
1425  001244  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
1426  001246  177607  000377          $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
1427  001252    077                   $QUES:  .ASCII  /?/             ;;QUESTION MARK
1428  001253    015                   $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
1429  001254  000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
1430                                  ;;**************************************************************
1431
1432                                  ;THE FOLLOWING TAG(S) ARE USER SUPPLIED BY CALLING THE MACRO
1433                                  ;'MORETAGS' AS ONE OF THE ARGUMENTS TO THE SYSMAC ROUTINE .$CMTAG
1434
1435  001256  000000                  TABFLG: .WORD   0               ;AN INDICATOR TO SHOW THAT THE
1436                                                                  ;INFORMATION FOR MULTIPLE DEVICE
1437                                                                  ;TESTING HAS ALREADY TRANSPIRED
1438                                                                  ;& 'MAINDEC' NAME HAS BEEN PRINTED
1439  001260  000000                  DLBASE: .WORD   0               ;STORAGE & WORKING LOCATION FOR A DEVICE
1440                                                                  ;RECEIVER STATUS REGISTER ADDRESS
1441  001262  000000                  KEEPAD: .WORD   0               ;STORAGE LOCATION FOR THE 1ST
1442                                                                  ;DEVICE RCSR FROM WHICH
1443                                                                  ;'BASEADD' IS RESTORED AT THE
1444                                                                  ;END OF A COMPLETE PROGRAM PASS.
1445  001264  000000                  BASEADD:.WORD   0               ;STORAGE LOCATION WHICH HOLDS
1446                                                                  ;THE RCSR ADDRESS OF THE 'NEXT'
1447                                                                  ;DEVICE DURING MULTIPLE TESTING
1448  001266  000000                  KEEPIV: .WORD   0               ;STORAGE LOCATION FOR THE 1ST
1449                                                                  ;DEVICE RECEIVER VECTOR FROM
1450                                                                  ;WHICH 'BASEIV' IS RESTORED AT THE
1451                                                                  ;END OF A COMPLETE PROGRAM PASS
1452  001270  000000                  BASEIV: .WORD   0               ;STORAGE LOCATION WHICH HOLDS
1453                                                                  ;THE VECTOR ADDRESS OF THE 'NEXT'
1454                                                                  ;DEVICE DURING MULTIPLE TESTING
1455  001272  000000                  MULTD:  .WORD   0               ;FLAG TO INDICATE TO 'END OF PASS'
1456                                                                  ;ROUTINE THAT MULTIPLE DEVICE
1457                                                                  ;TESTING IS BEING CONDUCTED
1458                                                                  ;0=NO, 1=YES
1459  001274  000000                  ACTREG: .WORD   0               ;THIS IS THE DEVICE ACTIVE REGISTER
1460                                                                  ;A BIT IS SET (STARTING AT
1461                                                                  ;BIT0)FOR EACH CONTIGUOUS DEVICE
1462                                                                  ;(A MAX. OF 16) THAT IS TO UNDERGO
1463                                                                  ;TESTING.  THIS LOCATION IS
1464                                                                  ;AUTOMATICALLY FILLED BASED ON
1465                                                                  ;USER RESPONSE TO PROGRAM QUESTIONS
1466  001276  000000                  ROTADD: .WORD   0               ;A ROTATING POINTER TO SIGNAL
1467                                                                  ;THE LAST DEVICE TESTED (IF
1468                                                                  ;MULTIPLE DEVICE TESTING WAS BEING
1469                                                                  ;DONE) IF LESS THAN A FULL COMPLE-
1470                                                                  ;MENT OF DEVICES (16)WAS SELECTED
1471  001300  000000                  LASTADD:.WORD   0               ;STORAGE LOCATION FOR THE
1472                                                                  ;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 $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

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


001310                          $ERRTB:

                                ;ERROR TABLE ITEM FOR ERROR MESSAGE 1

001310  015714                          EM1             ;'DL11 REGISTER REFERENCE CAUSED TIMEOUT''
001312  015763                          DH1             ;''  (PC)  (PS)  (SP)  TEST  DEVADR  REGADR  ''
001314  016042                          DT1             ; (R7)  (PSW)  (R6)  (R0)  (R1)  (R2)
001316  000000                          0               ;PRINT ALL OCTAL

                                ;ERROR TABLE ITEM FOR ERROR MESSAGE 2

001320  016060                          EM2             ;'' DL11 REGISTER ERROR ''
001322  016104                          DH2             ;''  (PC)  (PS)  (SP)  TEST  DEVADR  REGADR  WAS  S/B  ''
001324  016202                          DT2             ;'' (R7)  (PSW)  (R6)  (R0)  (R1)  (R2)  (R3)  (R4)  ''
001326  000000                          0               ;PRINT ALL OCTAL

                                ;ERROR TABLE ITEM FOR ERROR MESSAGE 3

001330  016224                          EM3             ;'' DL11 DATA COMPARE ERROR ''
001332  016254                          DH3             ;'' (PC)  (PS)  (SP)  TEST  WASADR  SHBADR  WAS  S/B  ''
001334  016352                          DT3             ;''  (R7)  (PSW)  (R6)  (R0)  (R1)  (R2)  (R3)  (R4)  ''
001336  000000                          0               ;PRINT ALL OCTAL

                                ;ERROR TABLE ITEM FOR ERROR MESSAGE 4

001340  016374                          EM4             ;'' UNEXPECTED TRAP TO VECTOR AT LOCATION XXX ''
001342  016446                          DH4             ;''  (PC)  (PS)  (SP)  TEST  ''
001344  016504                          DT4             ;'' (R7)  (PSW)  (R6)  (R0)  ''
001346  000000                          0               ;PRINT ALL OCTAL

                                ;ERROR TABLE ITEM FOR ERROR MESSAGE 5

001350  016516                          EM5             ;'' DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN)  ''
001352  016573                          DH5             ;'' (PC)  (PS)  (SP)  TEST  DEVADR  REGADR  (REG)  ''
001354  016662                          DT5             ;''  (R7)  (PSW)  (R6)  (R0)  (R1)  (R2)  (R3)  ''
001356  000000                          0

                                ;ERROR TABLE ITEM FOR ERROR MESSAGE 6

001360  015714                          EM1             ;'DL11 REGISTER REFERENCE CAUSED TIMEOUT''
001362  016702                          DH6             ;''  (PC)    (PS)    (SP)    REGADR''
001364  016742                          DT6             ;$ERRPC,$TMP0,$REG6,$REG2
```

```
1547  001366  000000                    0                ;PRINT ALL OCTAL
1548
1549                                ;ERROR TABLE ITEM FOR ERROR MESSAGE 7
1550
1551  001370  016516                    EM5              ;'' DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN) ''
1552  001372  016754                    DH7              ;'' (PC)  DEVADR  REGADR  (REG)''
1553  001374  017014                    DT7              ;$ERRPC,$REG1,$REG2,$REG3
1554  001376  000000                    0                ;PRINT ALL OCTAL
1555
1556                                ;ERROR TABLE ITEM FOR ERROR MESSAGE 10
1557
1558  001400  016224                    EM3              ;'' DL11 DATA COMPARE ERROR ''
1559  001402  017026                    DH10             ;'' (PC)  DEVADR  REGADR  (REG)  S/B''
1560  001404  017074                    DT10             ;$ERRPC,$REG1,$REG2,$REG3,$REG4
1561  001406  000000                    0                ;PRINT ALL OCTAL
1562
1563                                ;;*****************************************************************
1564                                    ;DL11 DEFINITIONS
1565                                ;;*****************************************************************
1566
1567  001410  175610                    DLRCSR: 175610          ;CONTAINS ADDRESS OF RCVR CSR
1568  001412  175612                    DLRDBR: 175612          ;CONTAINS ADDRESS OF RCVR DBR
1569  001414  175614                    DLXCSR: 175614          ;CONTAINS ADDRESS OF XMIT CSR
1570  001416  175616                    DLXDBR: 175616          ;CONTAINS ADDRESS OF XMIT DBR
1571  001420  000300                    DLVECT: 300             ;CONTAINS VECTOR ADDRESS OF CURRENT DL11
1572  001422  000000                    XFLG0:  0               ;FLAG FOR HARD XMIT ERRORS
1573  001424  000000                    RFLG0:  0               ;FLAG FOR HARD RCVR ERRORS
1574  001426  000000                    RFLG1:  0               ;FLAG FOR SOFT RCVR ERRORS
1575  001430  000000                    RTRY:   0               ;COUNTS NO. OF RETRIES ON SOFT ERRORS
1576  001432  000000                    OPTR:   0               ;CONTAINS POINTER TO OUTPUT BUFFER
1577  001434  000000                    IPTR:   0               ;CONTAINS POINTER TO INPUT BUFFER
1578  001436  000000                    LDOUT:  0               ;CONTAINS POINTER TO LOAD BUFFER ROUTINE
1579  001440  000000                    TIMR1:  0               ;TIMERS FOR 256. BYTE BLOCK TRANSFERS
1580  001442  000000                    TIMR2:  0
1581  001444  000000                    TIMR3:  0               ;DELAY TIMER FOR TEST 10
1582  001446  000000                    INTFLG: 0               ;SOFTWARE INTR. FLAG
1583  001450  000000                    STAD:   0               ;TEMPORARY ADDRESS STORAGE LOC.
1584
1585
1586
1587
1588  001452  000240              BEGIN: NOP                     ;PROGRAM WILL START HERE
1589                                .SBTTL  INITIALIZE THE COMMON TAGS
1590                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1591  001454  012706  001100            MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
1592  001460  005026                    CLR     (R6)+           ;;CLEAR MEMORY LOCATION
1593  001462  022706  001140            CMP     #SWR,R6 ;;DONE?
1594  001466  001374                    BNE     .-6             ;;LOOP BACK IF NO
1595  001470  012706  001100            MOV     #STACK,SP       ;;SETUP THE STACK POINTER
1596                                ;;INITIALIZE A FEW VECTORS
1597  001474  012737  010406  000020    MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
1598  001502  012737  000340  000022    MOV     #340,@#IOTVEC+2 ;;LEVEL 7
1599  001510  012737  010660  000030    MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
1600  001516  012737  000340  000032    MOV     #340,@#EMTVEC+2 ;;LEVEL 7
1601  001524  012737  013630  000034    MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
1602  001532  012737  000340  000036    MOV     #340,@#TRAPVEC+2;LEVEL 7
```

```
1603  001540  012737  013714  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
1604  001546  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
1605  001554  005067  177462                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
1606  001560  005067  177460                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1607  001564  112767  000001  177323          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
1608  001572  012767  001572  177306          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1609  001600  012767  001600  177302          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
1610                                  ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1611                                  ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1612  001606  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
1613  001612  012737  001646  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
1614  001620  012767  177570  177312          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
1615  001626  012767  177570  177306          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
1616  001634  022777  177777  177276          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
1617  001642  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1618                                                                  ;;AND  THE HARDWARE SWR IS NOT = -1
1619  001644  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
1620  001646  012716  001654          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
1621  001652  000002                          RTI
1622  001654  012767  000176  177256  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
1623  001662  012767  000174  177252          MOV     #DISPREG,DISPLAY
1624  001670  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
1625
1626                                  .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1627  001674  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
1628  001700  001006                          BNE     67$             ;;BRANCH IF YES
1629  001702  026727  177232  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
1630  001710  001005                          BNE     68$             ;;BRANCH IF NO
1631  001712  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
1632  001714  000403                          BR      68$
1633  001716  112767  000001  177210  67$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
1634  001724                          68$:
1635  001724  004767  007714                  JSR     PC,$TKINT       ;SET TTY INTERRUPT
1636  001730  000177  177514                  JMP     @STAD           ;JUMP TO SELECTED TEST
1637  001734  005067  177332          PRG1:   CLR     MULTD           ;CLEAR MULTIPLE DEVICE
1638                                                                  ;TESTING FLAG
1639  001740  005067  177312                  CLR     TABFLG          ;CLEAR TABLE CREATION FLAG
1640  001744  012767  000010  177262          MOV     #8.,$TMP15      ;SET CHARACTER LENGTH DESIGNATOR
1641                                                                  ;FOR 8 BITS --- THIS IS THE DEFAULT
1642                                                                  ;LENGTH ASSUMED BY THE PROGRAM
1643                                                                  ;UNLESS THE USER CHANGES IT THRU
1644                                                                  ;THE QUESTION AND ANSWER CYCLE
1645                                                                  ;INITIATED BY SETTING SW<0> TO A 1
1646  001752  012767  000200  177322          MOV     #200,DLPRI      ;SET STANDARD PRIORITY LEVEL
1647                                                                  ;FOR DEVICE
1648  001760  032777  000400  177152          BIT #SW8,@SWR           ;IS THE 'LOOP ON TEST' SWITCH SET?
1649  001766  001411                          BEQ     1$              ;BRANCH IF NOT
1650                                  ;
1651                                  ;IF THE 'LOOP ON TEST' SWITCH WAS SET WE WILL TAKE THE NEXT BRANCH
1652                                  ;INSTRUCTION THUS BYPASSING TABLE CREATION
1653                                  ;
1654                                  ;IF THE USER DESIRED TO LOOP ON A TEST OF OTHER THAN THE DEFAULT DEVICE
1655                                  ;THEN HE SHOULD HAVE PREVIOUSLY FILLED THE FOLLOWING PROGRAM LOCATIONS
1656                                  ;WITH THE DESIRED DEVICE REGISTER VALUES:
1657                                  ;
1658                                  ;                       ********************************
```

```
1659                                    ;                UNDER   ;DL11 DEFINITIONS                ABOVE
1660                                    ;                        **********************************
1661                                    ;
1662                                    ;       DLRCSR:          PATCH THE ADDRESS OF THE RCVR CSR
1663                                    ;       DLRDBR:          PATCH THE ADDRESS OF THE RCVR DBR
1664                                    ;       DLXCSR:          PATCH THE ADDRESS OF THE XMIT CSR
1665                                    ;       DLXDBR:          PATCH THE ADDRESS OF THE XMIT DBR
1666                                    ;       DLVECT:          PATCH THE VECTOR ADDRESS OF THIS DL11
1667                                    ;
1668  001770  104401  017417                       TYPE,   STMES           ;PRINT OUT 'MAINDEC' NAME
1669  001774  104401  021453                       TYPE,   FAILSA          ;TYPE FAILSAFE MESSAGE
1670  002000  104000                               ERROR   +0              ;TYPE OUT THE PC VALUE
1671  002002  104401  022031                       TYPE,   PCMSG           ;FOLLOWED BY =PC
1672  002006  000000                               HALT                    ;WAIT FOR USER TO RESPOND
1673  002010  000443                               BR      ONCE            ;GO TO TEST DEVICE PATCHED IN BY USER
1674  002012                                1$:
1675                                        ;ENSURE THAT IF MULTIPLE DEVICE TESTING WAS BEING DONE
1676                                        ;AND THE USER 'HALTED' THE PROGRAM BEFORE ALL DEVICES
1677                                        ;WERE COMPLETED AND WENT BACK TO 'LOAD ADDRESS 200'
1678                                        ;TO RESTART THE PROGRAM THAT AS A BARE MINIMUM
1679                                        ;HE CAN RUN THE DEFAULT DEVICE (1ST RECEIVER
1680                                        ;STATUS REGISTER ADDRESS 175610)
1681                                        ;NOTE:  IF THIS IS NOT SUITABLE THE USER WILL
1682                                        ;       HAVE TO SET SW0=1 (OR UP) IN ORDER TO
1683                                        ;       RECREATE THE TABLE HE DESTROYED FROM
1684                                        ;       ABOVE
1685  002012  012767  175610  177240                MOV     #175610,DLBASE  ;1ST POSSIBLE RECEIVER CSR
1686  002020  004767  005754                       JSR     PC,DLADDR       ;FORM DL ADDRESSES FOR
1687                                                                        ;1ST POSSIBLE DEVICE
1688  002024  012767  000300  177366                MOV     #300,DLVECT     ;1ST POSSIBLE INTERRUPT VECTOR
1689  002032  005067  177220                       CLR     TABFLG          ;CLEAR TABLE CREATION FLAG
1690
1691  002036  012706  001100                RESTRT: MOV     #STACK,SP               ;SET UP STACK POINTER
1692  002042  012737  015602  000004                MOV     #BUSERR,@#ERRVEC        ;SET UP BUS ERROR VECTOR
1693  002050  012737  000340  000006                MOV     #340,@#ERRVEC+2
1694  002056  012737  015626  000010                MOV     #RSVERR,@#RESVEC        ;SET UP RSVD INSTR. VECTOR
1695  002064  012737  000340  000012                MOV     #340,@#RESVEC+2
1696
1697                                        ;THIS  NEXT SECTION WILL CHECK TO SEE IF MULTIPLE DEVICE TESTING
1698                                        ;WILL TAKE PLACE I.E.-
1699                                        ;       A)  HAS FREE RUNNING DEVICE TABLE ALREADY BEEN CREATED, AND/OR
1700                                        ;       B)  IF IT HAS, DOES USER WISH TO CHANGE IT, OR DO WE TEST DEFAULT DEVICE?
1701  002072  105767  177160                       TSTB    TABFLG          ;HAS TABLE CREATION BEEN PERFORMED?
1702  002076  001010                               BNE     ONCE            ;BRANCH IF YES TO SKIP 'MAINDEC
1703                                                                        ;TITLE' MESSAGE
1704  002100  104401  017417                       TYPE    ,STMES          ;OTHERWISE, PRINT OUT 'MAINDEC'
1705                                                                        ;NAME
1706  002104  105167  177146                       COMB    TABFLG          ;IF TABLE CREATION HAS NOT BEEN
1707                                                                        ;PERFORMED, THEN SET FLAG, AND DO SO
1708  002110  032777  000001  177022                BIT     #SW0,@SWR       ;THE PROGRAM HAS OBVIOUSLY BEEN
1709                                                                        ;RESTARTED - DOES USER WISH TO
1710                                                                        ;RESELECT VECTOR AND CONTROL REGISTER
1711                                                                        ;ADDRESSES I.E. - CREATE A NEW TABLE?
1712  002116  001012                               BNE     GO              ;BRANCH IF YES
1713  002120  005077  177270                ONCE:  CLR     @DLXCSR         ;CLEAR OUT BOTH CSR'S
1714  002124  005077  177260                       CLR     @DLRCSR
```

```
1715  002130  005777  177256                  TST     @DLRDBR         ;FLUSH RCVR 'DONE'' BIT
1716  002134  005777  177252                  TST     @DLRDBR
1717  002140  000167  000670                  JMP     TST1            ;OTHERWISE, GO WITH EXISTING
1718                                                                    ;TABLE OR NOT USE ANY TABLE AT
1719                                                                    ;ALL WHICHEVER THE CASE MAY BE
1720                                                                    ;(DEFAULT CASE IS 1ST POSSIBLE
1721                                                                    ;DEVICE)
1722                                  ;IF WE COME THIS PATH THE USER HAS DECIDED 1 OF 2 ALTERNATIVES:
1723                                  ;       A)  TO RUN MULTIPLE DEVICES
1724                                  ;       B)  TO CREATE A NEW TABLE TO RUN FROM, OR
1725                                  ;       C)  TO CHANGE THE CHARACTER LENGTH
1726  002144  104401  020013          GO:     TYPE,   LENGTH          ;ASK USER FOR THE CHARACTER LENGTH
1727                                                                    ;FOR WHICH HIS DEVICE IS SET
1728  002150  104413                          RDDEC                   ;ACCEPT THE ANSWER TYPED BY USER
1729                                  ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1730  002152  012600                          MOV     (SP)+,R0        ;GET THE ANSWER TYPED
1731  002154  020027  000010                  CMP     R0,#8.          ;IS THE NUMBER TOO HIGH?
1732  002160  101114                          BHI     RETRY           ;IF YES - GO TO RETRY SITUATION
1733  002162  020027  000005                  CMP     R0,#5.          ;IS THE NUMBER TOO LOW?
1734  002166  103511                          BLO     RETRY           ;IF YES - GO TO RETRY SITUATION
1735  002170  010067  177040                  MOV     R0,$TMP15       ;THE VALUE TYPED IS OK
1736                                                                    ;STORE FOR FUTURE USE
1737  002174  104401  020075                  TYPE,   DEFAULT         ;ASK USER IF HE WISHES TO TEST OTHER
1738                                                                    ;THAN THE DEFAULT DEVICE
1739  002200  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
1740  002202  005726                          TST     (SP)+           ;LOOK AT THE ANSWER
1741  002204  001002                          BNE     1$              ;BRANCH IF REPLY WAS YES
1742  002206  000137  002770                  JMP     @#FLUSH         ;OTHERWISE, SKIP REST OF INTERROGATION
1743  002212  012700  000300          1$:     MOV     #300,R0         ;START RESTORATION OF TRAPCATCHER
1744  002216  012701  000302                  MOV     #302,R1         ;AREA FROM LOCATIONS 300 TO 776
1745  002222  012702  000004                  MOV     #4,R2           ;SO THAT WE CREATE THE MULTIPLE
1746  002226  010110          2$:     MOV     R1,(R0)         ;DEVICES TABLE WITH A CLEAN SLATE
1747  002230  005011                  CLR     (R1)            ;       .
1748  002232  060200                  ADD     R2,R0           ;       .
1749  002234  060201                  ADD     R2,R1           ;       .
1750  002236  022701  001000          CMP     #1000,R1        ;       .
1751  002242  002771                  BLT     2$              ;       .
1752                                  ;THE TRAPCATCHER VECTOR AREA FROM 300 - 776 SHOULD NOW BE RESTORED.
1753                                  ;PROCEED TO FIND OUT THE 1ST DEVICE RECEIVER CONTROL REGISTER
1754                                  ;ADDRESS
1755  002244  104401  020202          FIRSTD: TYPE    ,MFIRSTD        ;ASK USER FOR THE RECEIVER CONTROL
1756                                                                    ;REGISTER ADDRESS OF HIS FIRST
1757                                                                    ;DEVICE
1758  002250  104412                  RDOCT                           ;ACCEPT THE ANSWER TYPED BY USER
1759                                                                    ;AND STORE ON TOP OF STACK
1760                                  ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1761  002252  012600                  MOV     (SP)+,R0        ;GET THE ANSWER TYPED
1762  002254  020027  176170          CMP     R0,#176170      ;IS THE NUMBER TOO HIGH?
1763  002260  101060                  BHI     RETRY0          ;IF YES-GO TO RETRY SITUATION
1764  002262  020027  175610          CMP     R0,#175610      ;IS THE NUMBER TOO LOW?
1765  002266  103455                  BLO     RETRY0          ;IF YES - GO TO RETRY SITUATION
1766  002270  132700  000001          BITB    #BIT0,R0        ;NUMBER IS IN RANGE BUT IS IT
1767                                                          ;ON AN EVEN BOUNDARY?
1768  002274  001052                  BNE     RETRY0          ;IF NO - GO TO RETRY SITUATION
1769                                  ;CHECK TO SEE IF USER RESPONSE WAS TRULY A RCVR STATUS REGISTER
1770  002276  032700  000007          BIT #7,R0               ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
```

```
1771
1772  002302  001047                        BNE     RETRY0          ;USER RESPONSE EQUAL TO A ZERO?
1773  002304  010067  176750                MOV     R0,DLBASE       ;BRANCH IF NOT
1774                                                                  ;THE 1ST ADDRESS VALUE TYPED IS OK
1775                                        ;NOW WE ARE READY TO FIND OUT THE DEVICE INTERRUPT VECTOR
1776  002310  016767  176744  176744        MOV     DLBASE,KEEPADD  ;GET 1ST ADDRESS VALUE
1777  002316  004767  005456                JSR     PC,DLADDR       ;GO FORM DL ADDRESSES FOR
1778                                                                  ;1ST DEVICE SELECTED
1779  002322  016767  176734  176734        MOV     KEEPADD,BASEADD ;RESTORE 1ST DEVICE ADDRESS
1780  002330  104401  020270          VECT: TYPE    ,MVECT          ;ASK USER FOR A VECTOR ADDRESS
1781  002334  104412                        RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
1782                                                                  ;AND STORE ON TOP OF STACK
1783                                        ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1784  002336  012600                        MOV     (SP)+,R0        ;GET THE ANSWER TYPED
1785  002340  020027  000776                CMP     R0,#776         ;IS THE NUMBER TOO HIGH?
1786  002344  101032                        BHI     RETRY1          ;IF YES - GO TO RETRY SITUATION
1787  002346  020027  000300                CMP     R0,#300         ;IS THE NUMBER TOO LOW?
1788  002352  103427                        BLO     RETRY1          ;IF YES - GO TO RETRY SITUATION
1789  002354  132700  000001                BITB    #BIT0,R0        ;NUMBER IS IN RANGE BUT IS IT
1790                                                                  ;ON AN EVEN BOUNDARY?
1791  002360  001024                        BNE     RETRY1          ;IF NO - GO TO RETRY SITUATION
1792                                        ;CHECK TO SEE IF THE USER RESPONSE WAS TRULY A RCVR VECTOR ADDRESS
1793  002362  032700  000007                BIT     #7,R0           ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
1794                                                                  ;USER RESPONSE EQUAL TO A ZERO?
1795  002366  001021                        BNE     RETRY1          ;BRANCH IF NOT
1796  002370  010067  177024                MOV     R0,DLVECT       ;THE VECTOR VALUE TYPED IS OK
1797                                                                  ;STORE FOR FUTURE USE
1798  002374  016767  177020  176664        MOV     DLVECT,KEEPIV   ;GET THE FIRST VECTOR VALUE
1799  002402  016767  177012  176660        MOV     DLVECT,BASEIV   ;SAVE FIRST VECTOR VALUE
1800  002410  000414                        BR      HOWMANY         ;GO TO SEE IF USER WANTS MORE
1801                                                                  ;THAN 1 DEVICE
1802  002412  104401  001252          RETRY: TYPE,   $QUES           ;TYPE '?' INDICATING USER TYPED
1803                                                                  ;SOMETHING WRONG FOR CHARACTER LENGTH
1804  002416  000167  177522                JMP     GO              ;GO BACK TO REISSUE QUESTION
1805  002422  104401  001252          RETRY0: TYPE  ,$QUES          ;TYPE '?' INDICATING USER TYPE
1806                                                                  ;SOMETHING WRONG FOR 1ST ADDRESS
1807  002426  000167  177612                JMP     FIRSTD          ;GO BACK TO REISSUE QUESTION
1808  002432  104401  001252          RETRY1: TYPE  ,$QUES          ;TYPE '?' INDICATING USER TYPED
1809                                                                  ;SOMETHING WRONG FOR VECTOR
1810  002436  000167  177666                JMP     VECT            ;GO BACK TO REISSUE QUESTION
1811  002442  104401  020346          HOWMANY:TYPE  ,MULDEV         ;ASK USER IF HE WISHES TO RUN
1812                                                                  ;MULTIPLE DEVICES
1813  002446  104412                        RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
1814                                                                  ;AND STORE ON TOP OF STACK
1815  002450  012600                        MOV     (SP)+,R0        ;GET THE ANSWER TYPED
1816  002452  005700                        TST     R0              ;WAS THE ANSWER YES?
1817  002454  001003                        BNE     1$              ;BRANCH IF IT WAS
1818  002456  005067  176610                CLR     MULTD           ;OTHERWISE, INITIALIZE FLAG TO
1819                                                                  ;INDICATE NON-MULTIPLE DEVICES
1820  002462  000402                        BR      2$              ;SKIP NEXT INSTRUCTION
1821  002464  105167  176602          1$:   COMB    MULTD           ;INITIALIZE FLAG TO INDICATE
1822                                                                  ;RUNNING OF MULTIPLE DEVICES
1823  002470                           2$:
1824  002470  105767  176576                TSTB    MULTD           ;ARE THERE MULTIPLE DEVICES ON
1825                                                                  ;THE SYSTEM?
1826  002474  100406                        BMI     LASTD   ;IF SO, GO TO ASK NEXT QUESTION
```

```
1827  002476  005067  176572                  CLR     ACTREG          ;CLEAR DEVICE ACTIVE FLAG TO
1828                                                                    ;INDICATE NO RUNNING OF MULTIPLE
1829                                                                    ;DEVICES
1830  002502  005067  176570                  CLR     ROTADD          ;CLEAR DEVICE ADDRESS POINTER IN
1831                                                                    ;USE WHEN RUNNING MULTIPLE DEVICES
1832  002506  000167  000160                  JMP     CONQUES         ;SKIP ASKING NEXT QUESTION
1833  002512                          LASTD:
1834                                  ;WE WILL NOW BEGIN TO SET UP THE DEVICE ACTIVE REGISTER FOR RUNNING
1835                                  ;MULTIPLE DL11 DEVICES
1836  002512  104401  020433                  TYPE    ,MLASTD         ;ASK USER FOR THE RECEIVER
1837                                                                    ;CONTROL REGISTER ADDRESS OF
1838                                                                    ;HIS LAST DEVICE
1839  002516  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY
1840                                                                    ;USER AND STORE ON TOP OF STACK
1841                                  ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
1842  002520  012600                  1$:     MOV     (SP)+,R0        ;GET THE ANSWER TYPED
1843  002522  020027  176170                  CMP     R0,#176170      ;IS THE NUMBER TOO HIGH?
1844  002526  101132                          BHI     RETRY2          ;IF YES - GO TO RETRY SITUATION
1845  002530  020027  175610                  CMP     R0,#175610      ;IS THE NUMBER TOO LOW?
1846  002534  103527                          BLO     RETRY2          ;IF YES - GO TO RETRY SITUATION
1847  002536  132700  000001                  BITB    #BIT0,R0        ;NUMBER IS IN RANGE BUT IS IT
1848                                                                    ;ON AN EVEN BOUNDARY?
1849  002542  001124                          BNE     RETRY2          ;IF NOT - GO TO RETRY SITUATION
1850                                  ;CHECK TO SEE IF USER RESPONSE WAS TRULY A RCVR STATUS REGISTER
1851  002544  032700  000007                  BIT     #7,R0           ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
1852                                                                    ;USER RESPONSE EQUAL TO A ZERO?
1853  002550  001121                          BNE     RETRY2          ;BRANCH IF NOT
1854  002552  010067  176522                  MOV     R0,LASTADD      ;THE LAST ADDRESS VALUE TYPED IS OK
1855                                                                    ;STORE FOR FUTURE USE
1856                                  ;NOW WE BEGIN TO ACTUALLY INITIALIZE THE DEVICE ACTIVE REGISTER
1857                                  ;FROM WHICH THE PROGRAM WILL CYCLE UNTIL ALL DEVICES HAVE BEEN TESTED
1858  002556  012767  000001  176512          MOV     #1,ROTADD       ;SET UP POINTER FOR 'ACTREG'
1859  002564  005067  176504                  CLR     ACTREG          ;CLEAR DEVICE ACTIVE REGISTER
1860  002570  056767  176502  176476  2$:     BIS     ROTADD,ACTREG   ;MAKE 1ST DEVICE ACTIVE
1861  002576  000241                          CLC                     ;CLEAR CARRY BIT FOR POINTER
1862                                                                    ;ROTATION
1863  002600  006167  176472                  ROL     ROTADD          ;ARE WE PAST 16 LINE RANGE?
1864  002604  103422                          BCS     3$              ;BRANCH IF YES
1865  002606  062767  000010  176450          ADD     #10,BASEADD     ;STEP UP BASE ADDRESS
1866  002614  026767  176460  176442          CMP     LASTADD,BASEADD ;IS THIS THE LAST DEVICE?
1867  002622  101362                          BHI     2$              ;BRANCH IF NOT
1868                                  ;NOTE:  IF THIS PATH IS TAKEN IT IS ASSUMED THAT AT LEAST 2 DEVICES
1869                                  ;       EXIST AND THAT ALL ADDRESSING IS CONTIGUOUS
1870  002624  056767  176446  176442          BIS     ROTADD,ACTREG   ;INDICATE NEXT DEVICE ACTIVE
1871  002632  012767  000001  176436          MOV     #1,ROTADD       ;RESET POINTER FOR 'ACTREG' FOR
1872                                                                    ;LATER USE IN END OF PASS ROUTINE
1873  002640  016767  176416  176416          MOV     KEEPAD,BASEADD  ;RESET 1ST DEVICE RECEIVER
1874                                                                    ;CONTROLLER REGISTER ADDRESS FOR
1875                                                                    ;LATER USE IN END OF PASS ROUTINE
1876  002646  000167  000020                  JMP     CONQUES         ;GO TO CONTINUE QUESTIGNING OF USER
1877  002652                          3$:
1878                                  ;IF WE TAKE THIS PATH IT APPEARS THAT THERE ARE NOT AT LEAST
1879                                  ;TWO DEVICES PRESENT - IN RESPONSE TO USER TYPING 'YES' TO MULTIPLE
1880                                  ;DEVICES QUESTION
1881  002652  016767  176404  176404          MOV     KEEPADD,BASEADD ;RESET 1ST DEVICE RECEIVER
1882                                                                    ;CONTROLLER REGISTER ADDRESS
```

```
1883  002660  104401  020531                  TYPE    ,MRANGE         ;INFORM USER TO CHECK AND RETYPE
1884                                                                    ;THE LAST DEVICE RCSR ADDRESS
1885  002664  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
1886                                                                    ;AND STORE ON TOP OF STACK
1887  002666  000167  177626                  JMP     1$
1888  002672                          CONQUES:
1889                                  ;IF WE HAVE REACHED THIS PORTION WE KNOW:
1890                                  ;       A)  THE 'RXCSR' ADDRESS OF THE 1ST DEVICE
1891                                  ;       B)  THE 'RXCSR' ADDRESS OF THE LAST DEVICE, SAND
1892                                  ;       C)  THE INTERRUPT VECTOR OF THE 1ST DEVICE
1893                                  ;NOW LET'S FIND THE PRIORITY LEVEL
1894  002672  104401  020615                  TYPE    ,PLEVEL         ;ASK USER FOR PRIORITY LEVEL
1895  002676  104412                          RDOCT                   ;ACCEPT ANSWER TYPED BY USER AND
1896                                                                    ;STORE ON TOP OF STACK
1897  002700  012600                          MOV     (SP)+,R0        ;GET THE ANSWER TYPED
1898  002702  020027  000007                  CMP     R0,#7           ;IS THE NUMBER TOO HIGH?
1899  002706  101046                          BHI     RETRY3          ;IF YES - GO TO RETRY SITUATION
1900  002710  020027  000004                  CMP     R0,#4           ;IS THE NUMBER TOO LOW?
1901  002714  103443                          BLO     RETRY3          ;IF YES GO TO RETRY SITUATION
1902  002716  010067  176360                  MOV     R0,DLPRI        ;THE PRIORITY TYPED IN IS OK
1903                                                                    ;STORE FOR FUTURE USE
1904
1905                                  ;THIS SECTION WILL CALCULATE THE PRIORITY LEVEL FOR THE
1906                                  ;PROCESSOR BASED ON THE USER RESPONSE FOR PRIORITY LEVEL OF THE
1907                                  ;DEVICE
1908  002722  006367  176354                  ASL     DLPRI           ;FORM BITS <7-5> OF PSW
1909  002726  006367  176350                  ASL     DLPRI           ;       .
1910  002732  006367  176344                  ASL     DLPRI           ;       .
1911  002736  006367  176340                  ASL     DLPRI           ;       .
1912  002742  006367  176334                  ASL     DLPRI           ;       .
1913  002746  016767  176330  176330          MOV     DLPRI,LESS1     ;START TO FORM LEVEL TO ALLOW
1914                                                                    ;INTERRUPTS
1915  002754  162767  000001  176322          SUB     #1,LESS1        ;DROP DEVICE LEVEL PRIORITY
1916                                                                    ;BY 1 LEVEL FOR PSW
1917  002762  042767  000037  176314          BIC     #37,LESS1       ;MAKE SURE THE T,N,Z,V & C
1918                                                                    ;BITS FOR THE PROCESSOR ARE CLEAR
1919  002770  005077  176420          FLUSH:  CLR     @DLXCSR         ;CLEAR OUT BOTH CSR'S
1920  002774  005077  176410                  CLR     @DLRCSR
1921  003000  005777  176406                  TST     @DLRDBR         ;FLUSH RCVR 'DONE'' BIT
1922  003004  005777  176402                  TST     @DLRDBR
1923  003010  000167  000020                  JMP     TST1            ;BEGIN TESTING
1924  003014  104401  001252          RETRY2: TYPE    ,$QUES          ;TYPE '?' INDICATING USER TYPED
1925                                                                    ;SOMETHING WRONG FOR LAST ADDRESS
1926  003020  000167  177466                  JMP     LASTD   ;GO BACK TO REISSUE QUESTION
1927  003024  104401  001252          RETRY3: TYPE    ,$QUES          ;TYPE'?' INDICATING USER TYPED
1928                                                                    ;SOMETHING WRONG FOR PRIORITY
1929  003030  000167  177636                  JMP     CONQUES         ;GO BACK TO REISSUE QUESTION
1930
1931
1932                                  ;;*****************************************************************
1933                                  ;*TEST 1        TEST THAT REFERENCE TO RCSR DOES NOT CAUSE TIMEOUT
1934                                  ;;*****************************************************************
1935  003034  000004                  TST1:   SCOPE
1936  003036  016746  174742                  MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
1937  003042  012767  003060  174734          MOV     #1$,ERRVEC      ;GO TO 1$ IF TIMEOUT
1938  003050  016702  176334                  MOV     DLRCSR,R2       ;REGADR = RCSR ADR
```

```
1939  003054  005712                          TST     (R2)            ;USE REGADR ON BUS
1940  003056  000407                          BR      3$              ;;<GO TO NEXT TEST IF NO TIMEOUT>
1941  003060  004767  011614          1$:     JSR     PC,SUERT1       ;GO SET UP ERROR INFO
1942  003064  012767  003074  176152          MOV     #2$,$ESCAPE     ;RETURN TO 2$ AFTER ERROR PRINT
1943  003072  104001                          ERROR+1                 ;DL REFERENCE CAUSED BUS TIMEOUT
1944  003074  022626                  2$:     CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
1945  003076  012667  174702          3$:     MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
1946
1947                                          ;;**************************************************************
1948                                          ;*TEST 2         TEST THAT REFERENCE TO XCSR DOES NOT CAUSE TIMEOUT
1949                                          ;;**************************************************************
1950  003102  000004                  TST2:   SCOPE
1951  003104  016746  174674                  MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
1952  003110  012767  003126  174666          MOV     #1$,ERRVEC      ;GO TO 1$ IF TIMEOUT
1953  003116  016702  176272                  MOV     DLXCSR,R2       ;REGADR = XCSR ADR
1954  003122  005712                          TST     (R2)            ;USE REGADR ON BUS
1955  003124  000407                          BR      3$              ;;<GO TO NEXT TEST IF NO TIMEOUT>
1956  003126  004767  011546          1$:     JSR     PC,SUERT1       ;GO SET UP ERROR INFO
1957  003132  012767  003142  176104          MOV     #2$,$ESCAPE     ;RETURN TO 2$ AFTER ERROR PRINT
1958  003140  104001                          ERROR+1                 ;DL REFERENCE CAUSED BUS TIMEOUT
1959  003142  022626                  2$:     CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
1960  003144  012667  174634          3$:     MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
1961
1962                                          ;;**************************************************************
1963                                          ;*TEST 3         TEST THAT REFERENCE TO RDBR DOES NOT CAUSE TIMEOUT
1964                                          ;;**************************************************************
1965  003150  000004                  TST3:   SCOPE
1966  003152  016746  174626                  MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
1967  003156  012767  003174  174620          MOV     #1$,ERRVEC      ;GO TO 1$ IF TIMEOUT
1968  003164  016702  176222                  MOV     DLRDBR,R2       ;REGADR = RDBR ADR
1969  003170  005712                          TST     (R2)            ;USE REGADR ON BUS
1970  003172  000407                          BR      3$              ;;<GO TO NEXT TEST IF NO TIMEOUT>
1971  003174  004767  011500          1$:     JSR     PC,SUERT1       ;GO SET UP ERROR INFO
1972  003200  012767  003210  176036          MOV     #2$,$ESCAPE     ;RETURN TO 2$ AFTER ERROR PRINT
1973  003206  104001                          ERROR+1                 ;DL REFERENCE CAUSED BUS TIMEOUT
1974  003210  022626                  2$:     CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
1975  003212  012667  174566          3$:     MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
1976
1977                                          ;;**************************************************************
1978                                          ;*TEST 4         TEST THAT REFERENCE TO XDBR DOES NOT CAUSE TIMEOUT
1979                                          ;;**************************************************************
1980  003216  000004                  TST4:   SCOPE
1981  003220  016746  174560                  MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
1982  003224  012767  003242  174552          MOV     #1$,ERRVEC      ;GO TO 1$ IF TIMEOUT
1983  003232  016702  176160                  MOV     DLXDBR,R2       ;REGADR = XDBR ADR
1984  003236  005712                          TST     (R2)            ;USE REGADR ON BUS
1985  003240  000407                          BR      3$              ;;<GO TO NEXT TEST IF NO TIMEOUT>
1986  003242  004767  011432          1$:     JSR     PC,SUERT1       ;GO SET UP ERROR INFO
1987  003246  012767  003256  175770          MOV     #2$,$ESCAPE     ;RETURN TO 2$ AFTER ERROR PRINT
1988  003254  104001                          ERROR+1                 ;DL REFERENCE CAUSED BUS TIMEOUT
1989  003256  022626                  2$:     CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
1990  003260  012667  174520          3$:     MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
1991
1992                                          ;;**************************************************************
1993                                          ;*TEST 5         TEST THAT RCSR IS ALL ZEROES ON ENTRY
1994                                          ;;**************************************************************
```

```
1995  003264  000004                  TST5:   SCOPE
1996  003266  005004                          CLR     R4                      ;RESULT IN RCSR S/B = 0
1997  003270  016702  176114                  MOV     DLRCSR,R2               ;REGADR = RCSR ADR
1998  003274  020412                          CMP     R4,(R2)                 ;[RCSR]=000000 ??
1999  003276  001403                          BEQ     TST6                    ;;<BR IF YES>
2000  003300  004767  011314                  JSR     PC,SUER2                ;GO SET UP ERROR INFO
2001  003304  104002                          ERROR+2                         ;RCSR NOT CLEAR ON START UP
2002                                  ;;**********************************************************************
2003                                  ;*TEST 6         TEST THAT 'READY' BIT IS ONLY BIT SET IN XCSR
2004                                  ;;**********************************************************************
2005  003306  000004                  TST6:   SCOPE
2006  003310  012704  000200                  MOV     #200,R4                 ;RESULT IN XCSR S/B = 000200
2007  003314  016702  176074                  MOV     DLXCSR,R2               ;REGADR = XCSR ADR
2008  003320  020412                          CMP     R4,(R2)                 ;[XCSR]=000200 ??
2009  003322  001403                          BEQ     TST7                    ;;<BR IF YES>
2010  003324  004767  011270                  JSR     PC,SUER2                ;GO SETUP ERROR INFO
2011  003330  104002                          ERROR+2                         ;[XCSR] INCORRECT ON START UP
2012                                  ;;**********************************************************************
2013                                  ;*TEST 7         TEST THAT 'MAINT' BIT CAN BE SET AND CLEARED
2014                                  ;;**********************************************************************
2015  003332  000004                  TST7:   SCOPE
2016  003334  012704  000204                  MOV     #204,R4                 ;RESULT IN XCSR S/B = 000204
2017  003340  016702  176050                  MOV     DLXCSR,R2               ;REGADR = XCSR ADR
2018  003344  052712  000004                  BIS     #BIT2,(R2)              ;SET THE 'MAINT' BIT
2019  003350  020412                          CMP     R4,(R2) ;RESULT IN XCSR OK ??
2020  003352  001403                          BEQ     1$                      ;;<BR IF YES>
2021  003354  004767  011240                  JSR     PC,SUER2                ;GO SET UP ERROR INFO
2022  003360  104002                          ERROR+2                         ;MAINT. BIT FAILED TO SET PROPERLY
2023  003362  012704  000200          1$:     MOV     #200,R4                 ;RESULT IN XCSR S/B = 000200
2024  003366  042712  000004                  BIC     #BIT2,(R2)              ;NOW CLEAR THE 'MAINT' BIT
2025  003372  020412                          CMP     R4,(R2)                 ;RESULT IN XCSR OK ??
2026  003374  001403                          BEQ     TST10                   ;;<BR IF YES>
2027  003376  004767  011216                  JSR     PC,SUER2                ;GO SET UP ERROR INFO
2028  003402  104002                          ERROR+2                         ;MAINT BIT FAILED TO CLEAR PROPERLY
2029                                  ;;**********************************************************************
2030                                  ;*TEST 10        TEST THAT XMIT I.E. CAN CAUSE AN INTR
2031                                  ;;**********************************************************************
2032  003404  000004                  TST10:  SCOPE
2033  003406  005067  176034                  CLR     INTFLG                  ;INIT SOFTWARE INTR FLAG
2034  003412  012767  000001  176024          MOV     #1,TIMR3                ;SET TIMER FOR DELAY                    (CHGC6)
2035  003420  016705  175774                  MOV     DLVECT,R5               ;GET VECTOR ADDRESS
2036  003424  012765  003520  000004          MOV     #2$,4(R5)               ;GO TO 4$ ON INTR
2037  003432  016765  175644  000006          MOV     DLPRI,6(R5)             ;PRIORITY LEVEL 4
2038  003440  005005                          CLR     R5                      ;INIT INTR. TIMER
2039  003442  012704  000200                  MOV     #200,R4                 ;RESULT IN XCSR S/B = 000200
2040  003446  016702  175742                  MOV     DLXCSR,R2               ;REGADR = XCSR ADR
2041  003452  052712  000100                  BIS     #100,(R2)               ;SET INTR. ENABLE BIT 06
2042  003456  005767  175764          1$:     TST     INTFLG                  ;DID INTR OCCUR YET ??
2043  003462  001023                          BNE     3$                      ;BR IF IT DID
2044  003464  005305                          DEC     R5                      ;COUNT THE TIMER
2045  003466  001373                          BNE     1$                      ;BR IF NO TIMEOUT
2046  003470  005367  175750                  DEC     TIMR3                   ;REDUCE ADDED DELAY                     (CHGC6)
2047  003474  001770                          BEQ     1$                      ;BRANCH BACK ON FIRST PASS              (CHGC6)
2048  003476  012704  000300                  MOV     #300,R4                 ;RESULT IN XCSR S/B = 000300
2049  003502  004767  011112                  JSR     PC,SUER2                ;GO SETUP ERROR INFO
2050  003506  012767  003516  175530          MOV     #4$,$ESCAPE             ;RETURN TO 4$ AFTER ERROR PRINT
```

```
2051  003514  104002                          ERROR+2                 ;INTR.  FAILED
2052  003516                          4$:
2053  003516  000412                          BR      TST11           ;;<GO TO NEXT TEST>
2054  003520  005167  175722          2$:     COM     INTFLG          ;SET THE SOFTWARE FLAG
2055  003524  042712  000100                  BIC     #100,(R2)       ;TURN OFF I.E. BIT
2056  003530  000002                          RTI                     ;RETURN CONTROL TO INTR. ROUTINE
2057  003532  020412                  3$:     CMP     R4,(R2)         ;RESULT IN XCSR OK ??
2058  003534  001403                          BEQ     TST11           ;;<BR IF YES>
2059  003536  004767  011056                  JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2060  003542  104002                          ERROR+2                 ;XMIT INTR. NOT SERVICED PROPERLY
2061                                  ;;*****************************************************************
2062                                  ;*TEST 11        TEST THAT RCVR I.E. BIT CAN BE SET AND CLEARED
2063                                  ;;*****************************************************************
2064  003544  000004                  TST11:  SCOPE
2065  003546  012704  000100                  MOV     #100,R4         ;RESULT IN RCSR S/B = 000100
2066  003552  016702  175632                  MOV     DLRCSR,R2       ;REGADR = RCSR ADR
2067  003556  052712  000100                  BIS     #BIT6,(R2)      ;SET I.E. BIT
2068  003562  020412                          CMP     R4,(R2)         ;DID IT SET PROPERLY ??
2069  003564  001403                          BEQ     1$              ;;<BR IF YES>
2070  003566  004767  011026                  JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2071  003572  104002                          ERROR+2                 ;RCVR I.E. BIT FAILED TO SET PROPERLY
2072  003574  005004                  1$:     CLR     R4              ;RESULT IN RCSR S/B = 000000
2073  003576  042712  000100                  BIC     #BIT6,(R2)      ;CLEAR THE I.E. BIT
2074  003602  020412                          CMP     R4,(R2)         ;DID IT CLEAR PROPERLY ??
2075  003604  001403                          BEQ     TST12           ;;<BR IF YES>
2076  003606  004767  011006                  JSR     PC,SUER2        ;GO SET UP ERROR INFO
2077  003612  104002                          ERROR+2                 ;RCVR I.E. BIT FAILED TO CLEAR PROPERLY
2078                                  ;;*****************************************************************
2079                                  ;*TEST 12        TEST THAT RCVR 'DONE'' CAN GENERATE AN INTR.
2080                                  ;;*****************************************************************
2081  003614  000004                  TST12:  SCOPE
2082  003616  016705  175576                  MOV     DLVECT,R5       ;GET THE VECTOR ADDRESS
2083  003622  012725  004000                  MOV     #3$,(R5)+       ;GO TO 3$ ON RCVR INTR.
2084  003626  016715  175450                  MOV     DLPRI,(R5)      ;AT LEVEL 4
2085  003632  005067  175610                  CLR     INTFLG          ;INIT THE SOFTWARE FLAG
2086  003636  005005                          CLR     R5              ;INIT INTR. TIMER
2087  003640  105067  175340                  CLRB    $TMP1           ;INIT WHERE DATA WILL BE STORED
2088  003644  016702  175540                  MOV     DLRCSR,R2       ;REGADR = RCSR ADR
2089  003650  005012                          CLR     (R2)            ;INIT THE RCSR TO 000000
2090  003652  052712  000100                  BIS     #BIT6,(R2)      ;ENABLE RCVR INTERRUPTS
2091  003656  052762  000004  000004          BIS     #BIT2,4(R2)     ;NOW TURN ON MAINT MODE
2092  003664  112767  000252  175312          MOVB    #252,$TMP1      ;GET DATA PATTERN AND
2093  003672  004767  011614                  JSR     PC,UPMASK       ;GO MASK OFF BITS AS A FUNCTION OF
2094                                                                  ;CHARACTER LENGTH ( 5, 6, 7, OR 8 BITS)
2095  003676  116700  175330                  MOVB    $TMP14,R0       ;SAVE DATA PATTERN FOR FURTHER USE
2096  003702  116762  175324  000006          MOVB    $TMP14,6(R2)    ;LOAD XMIT BUFFER REG.
2097  003710  005767  175532          1$:     TST     INTFLG          ;DID RCVR INTR. YET ??
2098  003714  001044                          BNE     4$              ;BR IF IT DID
2099  003716  005305                          DEC     R5              ;COUNT THE TIMER
2100  003720  001373                          BNE     1$              ;BR IF NO TIMEOUT
2101  003722  013767  177776  175252          MOV     @#PSW,$TMP0     ;SAVE ERROR PSW
2102  003730  042762  000004  000004          BIC     #BIT2,4(R2)     ;DISABLE MAINT MODE
2103  003736  042712  000100                  BIC     #100,(R2)       ;DISABLE RCVR INTR.
2104  003742  010667  175230                  MOV     SP,$REG6        ;SAVE THE ERROR SP
2105  003746  010201                          MOV     R2,R1           ;DEVADR = RCSR ADR
2106  003750  011203                          MOV     (R2),R3         ;GET THE WAS DATA
```

```
2107  003752  012704  000200                    MOV     #200,R4         ;[RCSR] S/B = 000200
2108  003756  004767  010664                    JSR     PC,SUERR1       ;GO SET UP ERROR INFO.
2109  003762  012767  003772  175254            MOV     #2$,$ESCAPE     ;RETURN TO 2$ AFTER ERROR ALWAYS
2110  003770  104002                            ERROR+2                 ;RCVR INTERRUPT FAILED
2111  003772  005762  000002          2$:       TST     2(R2)           ;REFERENCE RCVR DATA BUFFER
2112                                                                    ;TO CLEAR RCSR IN CASE RCVR
2113                                                                    ;INTERRUPTS COULD NOT BE ENABLED
2114  003776  000437                            BR      TST13           ;;<GO TO NEXT TEST>
2115  004000  042762  000004  000004  3$:       BIC     #BIT2,4(R2)     ;DISABLE THE MAINT MODE
2116  004006  116267  000002  175170            MOVB    2(R2),$TMP1     ;GET THE RECEIVED DATA
2117  004014  042712  000100                    BIC     #BIT6,(R2)      ;TURN OFF RCVR INTR. ENAB
2118  004020  005167  175422                    COM     INTFLG          ;SET THE SOFTWARE FLAG
2119  004024  000002                            RTI                     ;RETURN TO MAINLINE
2120  004026  005004                    4$:     CLR     R4              ;[RCSR] S/B=0
2121  004030  005712                            TST     (R2)            ;IS IT ALL ZEROES ??
2122  004032  001403                            BEQ     5$              ;;<BR IF YES>
2123  004034  004767  010560                    JSR     PC,SUER2        ;GO SET UP ERROR INFO
2124  004040  104002                            ERROR+2                 ;RCVR INTR NOT SERVICED PROPERLY
2125  004042  016701  175344          5$:       MOV     DLRDBR,R1       ;SAVE WAS ADDRESS
2126  004046  016702  175344                    MOV     DLXDBR,R2       ;SAVE THE S/B ADDRESS
2127  004052  004767  011434                    JSR     PC,UPMASK       ;GET THE WAS DATA AND
2128                                                                    ;GO MASK OFF BITS AS A FUNCTION OF
2129                                                                    ;CHARACTER LENGTH ( 5, 6, 7, OR 8 BITS)
2130  004056  116703  175150                    MOVB    $TMP14,R3       ;SET UP FOR ERROR CHECKING
2131  004062  110004                            MOVB    R0,R4           ;GET THE S/B DATA
2132  004064  020403                            CMP     R4,R3           ;WAS = S/B ??
2133  004066  001403                            BEQ     TST13           ;;<BR IF YES>
2134  004070  004767  010552                    JSR     PC,SUERR1       ;GO SET UP THE ERROR INFO
2135  004074  104003                            ERROR+3                 ;DATA COMPARE ERROR
2136                                    ;;**************************************************************************
2137                                    ;*TEST 13        TEST THAT 'REQ TO SEND'' ASSERTS 'RING''
2138                                    ;;**************************************************************************
2139  004076  000004                    TST13:  SCOPE
2140  004100  032777  010000  175032            BIT     #SW12,@SWR      ;ARE WE TESTING /C OR /D MODEL?
2141  004106  001047                            BNE     TST14           ;;<BRANCH IF YES>
2142  004110  012704  140004                    MOV     #140004,R4      ;RESULT IN RCSR S/B = 140004
2143  004114  016702  175270                    MOV     DLRCSR,R2       ;REGADR = RCSR ADR
2144  004120  005012                            CLR     (R2)            ;INIT THE RCSR TO 000000
2145  004122  052712  000004                    BIS     #BIT2,(R2)      ;SET 'REQ TO SEND''
2146  004126  032777  100000  175254            BIT     #BIT15,@DLRCSR  ;DID 'RING'' SET 'DATA SET INT'' ?
2147  004134  001003                            BNE     1$              ;;<BR IF YES>
2148  004136  004767  010456                    JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2149  004142  104002                            ERROR+2                 ;'RING'' TRANSITION FAILED TO SET 'DATA SET INT''
2150                                                                    ;NOTE: 'BIT  #BIT15,(R2)'' RESETS BIT15
2151  004144  012704  040004          1$:       MOV     #40004,R4       ;RESULT IN RCSR S/B = 40004
2152  004150  020412                            CMP     R4,(R2)         ;BOTH 'RING'' AND 'REQ TO SEND'' ASSERTED ?
2153  004152  001403                            BEQ     2$              ;;<BR IF YES>
2154  004154  004767  010440                    JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2155  004160  104002                            ERROR+2                 ;'RING'' OR 'REQ TO SEND'' FAILED TO SET
2156  004162  005004                    2$:     CLR     R4              ;RESULT IN RCSR S/B = 000000
2157  004164  042712  000004                    BIC     #BIT2,(R2)      ;CLEAR 'REQ TO SEND''
2158  004170  032777  100000  175212            BIT     #BIT15,@DLRCSR  ;DID 'DATA SET INT'' GET SET ??
2159  004176  001403                            BEQ     3$              ;;<BR IF NOT>
2160  004200  004767  010414                    JSR     PC,SUER2        ;GO SET UP ERROR INFO
2161  004204  104002                            ERROR+2                 ;CLEARING 'RING'' SET 'DATA SET INT''
2162  004206  020412                    3$:     CMP     R4,(R2)         ;RCSR CONTAIN ALL ZEROES ??
```

```
2163  004210  001406                          BEQ     TST14           ;;<BR IF YES>
2164  004212  004767  010402                  JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2165  004216  016767  000002  174754          MOV     .+6,$REG7       ;SAVE THE ERROR PC
2166  004224  104002                          ERROR+2                 ;CLEARING 'REQ TO SEND' FAILED TO CLEAR 'RING'
2167                          ;;*****************************************************************
2168                          ;*TEST 14        TEST THAT 'SEC XMIT' ASSERTS 'SEC REC' AND 'DATA SET INT'
2169                          ;;*****************************************************************
2170  004226  000004          TST14:  SCOPE
2171  004230  032777  010000  174702          BIT     #SW12,@SWR      ;ARE WE TESTING /C OR /D MODEL?
2172  004236  001046                          BNE     TST15           ;;<BRANCH IF YES>
2173  004240  016702  175144                  MOV     DLRCSR,R2       ;REGADR = RCSR ADR
2174  004244  005012                          CLR     (R2)            ;INIT RCSR TO 000000
2175  004246  012704  102010                  MOV     #102010,R4      ;CONTENTS OF RCSR S/B = 102010
2176  004252  052712  000010                  BIS     #BIT3,(R2)      ;SET 'SEC XMIT' BIT
2177  004256  032777  100000  175124          BIT     #BIT15,@DLRCSR  ;DID 'DATA SET INT' SET ??
2178  004264  001003                          BNE     1$              ;;<BR IF YES>
2179  004266  004767  010326                  JSR     PC,SUER2        ;GO SET UP ERROR INFO
2180  004272  104002                          ERROR+2                 ;'DATA SET INT' FAILED TO SET-NOTE THAT
2181                                                                  ;'BIT  #BIT15,(R2)' RESETS BIT15
2182  004274  012704  002010          1$:     MOV     #2010,R4        ;RESULT IN RCSR S/B = 2010
2183  004300  020412                          CMP     R4,(R2)         ;ARE 'SEC XMIT' AND 'SEC REC' BOTH SET ?
2184  004302  001403                          BEQ     2$              ;;<BR IF YES>
2185  004304  004767  010310                  JSR     PC,SUER2        ;GO SET UP ERROR INFO
2186  004310  104002                          ERROR+2                 ;'SEC XMIT' OR 'SEC REC' FAILED TO SET
2187                                                                  ;OR 'DATA SET INT' FAILED TO BE CLEARED
2188                                                                  ;WHEN REFERENCING RCSR
2189  004312  012704  100000          2$:     MOV     #BIT15,R4       ;RESULT IN RCSR S/B = 100000
2190  004316  042712  000010                  BIC     #BIT3,(R2)      ;CLEAR 'SEC XMIT' BIT
2191  004322  032777  100000  175060          BIT     #BIT15,@DLRCSR  ;DID CLEARING IT SET 'DATA SET INT'??
2192  004330  001003                          BNE     3$              ;;<BR IF YES>
2193  004332  004767  010262                  JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2194  004336  104002                          ERROR+2                 ;CLEARING 'SEC XMIT' FAILED TO SET 'DATA
2195                                                                  ;SET INT. (NOTE THAT REFERENCING RCSR
2196                                                                  ; CLEARS 'DATA SET INT'
2197  004340  005004          3$:     CLR     R4                      ;RESULT IN RCSR S/B = 000000
2198  004342  020412                          CMP     R4,(R2)         ;'SEC XMIT' AND 'SEC REC' CLEAR ?
2199  004344  001403                          BEQ     TST15           ;;<BR IF YES>
2200  004346  004767  010246                  JSR     PC,SUER2        ;GO SETUP ERROR INFO
2201  004352  104002                          ERROR+2                 ;'SEC XMIT' OR 'SEC REC' FAILED TO CLEAR
2202                                                                  ;OR REFERENCING RCSR  FAILED TO CLEAR 'DATA SET INT'
2203                          ;;*****************************************************************
2204                          ;*TEST 15        TEST THAT 'DTR' CAN ASSERT 'CLR TO SEND' AND 'CAR DET'
2205                          ;;*****************************************************************
2206  004354  000004          TST15:  SCOPE
2207  004356  032777  010000  174554          BIT     #SW12,@SWR      ;ARE WE TESTING /C OR /D MODEL?
2208  004364  001046                          BNE     TST16           ;;<BRANCH IF YES>
2209  004366  016702  175016                  MOV     DLRCSR,R2       ;REGADR = RCSR ADR
2210  004372  005012                          CLR     (R2)            ;INIT RCSR TO 000000
2211  004374  012704  130002                  MOV     #130002,R4      ;RESULT IN RCSR S/B = 130002
2212  004400  052712  000002                  BIS     #BIT1,(R2)      ;SET 'DTR' BIT
2213  004404  032777  100000  174776          BIT     #BIT15,@DLRCSR  ;DID 'DATA SET INT' SET ??
2214  004412  001003                          BNE     1$              ;;<BR IF YES>
2215  004414  004767  010200                  JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2216  004420  104002                          ERROR+2                 ;'DATA SET INT' FAILED TO SET -
2217                                                                  ;NOTE: THE REFERENCE TO RCSR  ABOVE WILL
2218                                                                  ;WILL UNCONDITIONALLY CLEAR RCSR BIT 15.
```

```
 2219  004422  012704  030002          1$:     MOV     #30002,R4       ;RESULT IN RCSR S/B = 30002
 2220  004426  020412                          CMP     R4,(R2)         ;'DTR','CLR TO SEND', AND ''CAR DET'' ALLSET
 2221  004430  001403                          BEQ     2$              ;;<BR IF ALL SET>
 2222  004432  004767  010162                  JSR     PC,SUER2        ;GO SET UP ERROR INFO
 2223  004436  104002                          ERROR+2                 ;'DTR','CLR TO SEND'' OR ''CAR DET'' FAILED
 2224                                                                  ;TO SET OR 'DATA SET INT'' FAILED TO CLEAR
```

```
  2225  004440  012704  100000          2$:     MOV     #BIT15,R4       ;RESULT IN RCSR S/B = 100000
```

```
2226  004444  042712  000002          BIC     #BIT1,(R2)          ;NOW CLEAR 'DTR''
2227  004450  032777  100000  174732  BIT     #BIT15,@DLRCSR      ;DID 'DATA SET INT'' SET ??
2228  004456  001003                  BNE     3$                  ;;<BR IF YES>
2229  004460  004767  010134          JSR     PC,SUER2            ;GO SETUP ERROR INFO
```

```
                                        ERROR+2                 ;'DATA SET INT'' FAILED TO SET WHEN 'DTR''
                                                                ;WENT TO A ZERO.
                                3$:     CLR     R4              ;RESULT IN RCSR S/B = 000000
                                        CMP     R4,(R2)         ;DID ALL BITS CLEAR??
                                        BEQ     TST16           ;;<BR IF YES>
                                        JSR     PC,SUER2        ;GO SET UP ERROR INFO
                                        ERROR+2                 ;'DTR'',''CLR TO SEND'' OR ''CAR DET'' FAILED
                                                                ;TO CLEAR PROPERLY
                                ;;*******************************************************************
                                ;*TEST 16       TEST THAT 'DATA SET INT ENAB'' CAN SET AND CLEAR
                                ;;*******************************************************************
                                TST16:  SCOPE
                                        BIT     #SW12,@SWR      ;ARE WE TESTING /C OR /D MODEL?
                                        BNE     TST17           ;;<BRANCH IF YES>
                                        MOV     DLRCSR,R2       ;REGADR = RCSR ADR
                                        MOV     #40,R4          ;RESULT IN RCSR S/B = 000040
                                        BIS     #BIT5,(R2)      ;SET THE 'DATA SET I.E.'' BIT
                                        CMP     R4,(R2)         ;DID IT SET OK ??
                                        BEQ     1$              ;;<BR IF YES>
                                        JSR     PC,SUER2        ;GO SET UP ERROR INFO
                                        ERROR+2                 ;'DAT SET I. E.'' FAILED TO SET
                                1$:     CLR     R4              ;MAKE S/B DATA = 000000
                                        BIC     #BIT5,(R2)      ;NOW CLEAR THE 'DATA SET I.E.'' BIT
                                        CMP     R4,(R2)         ;DID IT CLEAR OK ??
                                        BEQ     TST17           ;;<BR IF YES>
                                        JSR     PC,SUER2        ;GO SET UP ERROR INFO.
                                        ERROR+2                 ;'DATA SET I.E.'' FAILED TO CLEAR
                                ;;*******************************************************************
                                ;*TEST 17       TEST THE 'DATA SET I.E.'' CAN CAUSE A RCVR INTR
                                ;;*******************************************************************
                                TST17:  SCOPE
                                        BIT     #SW12,@SWR      ;ARE WE TESTING A /C OR /D MODEL?
                                        BNE     TST20           ;;<BRANCH IF YES>
                                        MOV     DLVECT,R5       ;GET THE VECTOR ADDR
                                        MOV     #3$,(R5)+       ;GO TO 3$ ON RCVR INTR.
                                        MOV     DLPRI,(R5)      ;AT LEVEL 4
                                        CLR     R5              ;INIT INTR. TIMER
                                        CLR     INTFLG          ;INIT SOFTWARE FLAG
                                        CLR     R4              ;RESULT IN RCSR S/B = 0 AFTER INTR.
                                        MOV     DLRCSR,R2       ;REGADR = RCSR ADR
                                        BIS     #BIT5,(R2)      ;SET THE 'DATA SET I.E.'' BIT
                                        BIS     #BIT1,(R2)      ;NOW SET 'DTR'' TO GEN INTR.
                                1$:     TST     INTFLG          ;DID INTR OCCUR YET ??
                                        BNE     4$              ;BR IF YES
                                        DEC     R5              ;COUNT THE TIMER
                                        BNE     1$              ;BR IF NO TIMEOUT
                                        JSR     PC,SUER2        ;GO SET UP ERROR INFO
                                        CLR     (R2)            ;TURN IT ALL OFF
                                        MOV     #2$,$ESCAPE     ;COME BACK TO 2$ IN ALL CASES
                                        ERROR+2                 ;'DATA SET'' INTR FAILED TO OCCUR
                                2$:
                                        BR      TST20           ;;<GO TO NEXT TEST>
                                3$:     CLR     (R2)            ;ZERO THE RCSR
                                        COM     INTFLG          ;SET THE SOFTWARE FLAG
                                        RTI                     ;RETURN TO SENDER
                                4$:     BIT     #BIT15,(R2)     ;DID 'DATA SET INT'' GET SET BY INTR. SERVICE ??
```

```
2286  004702  001003                        BNE     5$              ;;<BR IF YES>
2287  004704  004767  007710                JSR     PC,SUER2        ;GO SET UP ERROR INFO
2288  004710  104002                        ERROR+2                 ;DATA SET INTR. NOT SERVICED PROPERLY
2289  004712  020412                5$:     CMP     R4,(R2)         ;ALL BITS IN RCSR CLEAR ??
2290  004714  001403                        BEQ     TST20           ;;<BR IF YES>
2291  004716  004767  007676                JSR     PC,SUER2        ;GO SET UP ERROR INFO
2292  004722  104002                        ERROR+2                 ;INTR. SERVICE FAILED TO CLEAR RCSR
2293                                ;;**********************************************************************
2294                                ;*TEST 20        TEST THAT THE 'BREAK' BIT CAN BE SET AND CLEARED
2295                                ;;**********************************************************************
2296  004724  000004                TST20:  SCOPE
2297  004726  032777  010000  174204        BIT     #SW12,@SWR      ;ARE WE TESTING /C OR /D MODEL?
2298  004734  001024                        BNE     TST21           ;;<BRANCH IF YES>
2299  004736  012704  000201                MOV     #201,R4         ;RESULT S/B = 201 IN XCSR
2300  004742  016702  174446                MOV     DLXCSR,R2       ;SET UP REGADR
2301  004746  052712  000001                BIS     #BIT0,(R2)      ;SET THE 'BREAK' BIT
2302  004752  020412                        CMP     R4,(R2)         ;DID IT SET PROPERLY ??
2303  004754  001403                        BEQ     1$              ;;<BR IF YES>
2304  004756  004767  007636                JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2305  004762  104002                        ERROR+2                 ;'BREAK' BIT FAILED TO SET PROPERLY
2306  004764  012704  000200          1$:   MOV     #200,R4         ;RESULT S/B = 200 IN XCSR
2307  004770  042712  000001                BIC     #BIT0,(R2)      ;CLEAR THE 'BREAK' BIT
2308  004774  020412                        CMP     R4,(R2)         ;DID IT CLEAR PROPERLY ??
2309  004776  001403                        BEQ     TST21           ;;<BR IF YES>
2310  005000  004767  007614                JSR     PC,SUER2        ;GO SET UP ERROR INFO
2311  005004  104002                        ERROR+2                 ;'BREAK' FAILED TO CLEAR PROPERLY
2312
2313
2314                                ;;**********************************************************************
2315                                ;*TEST 21        TEST THAT A 'RESET' CLEARS THE 'BREAK' BIT
2316                                ;;**********************************************************************
2317  005006  000004                TST21:  SCOPE
2318  005010  012704  000200                MOV     #200,R4         ;RESULT S/B = 200
2319  005014  016702  174374                MOV     DLXCSR,R2       ;SET UP REGADR
2320  005020  052712  000001                BIS     #BIT0,(R2)      ;SET THE 'BREAK' BIT
2321  005024  000005                        RESET                   ;CLEAR IT WITH A 'RESET'
2322  005026  020412                        CMP     R4,(R2)         ;DID IT CLEAR ??
2323  005030  001403                        BEQ     TST22           ;;<BR IF YES>
2324  005032  004767  007562                JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2325  005036  104002                        ERROR+2                 ;RESET INSTR. FAILED TO CLEAR 'BREAK'
2326
```

```
                                        ;;*****************************************************************
                                        ;*TEST 22        TEST TO TURN AROUND NULL-DEL-NULL PATTERN
                                        ;;*****************************************************************
005040  000004                  TST22:  SCOPE
005042  012767  000001  174172          MOV     #1,$TIMES       ;;DO 1 ITERATION
005050  004767  007700                  JSR     PC,SUVEC        ;GO SET UP VECTORS
005054  005067  174350                  CLR     RTRY            ;INITIALIZE RETRY FLAG
005060  012767  015130  174350  1$:     MOV     #LDOUT1,LDOUT   ;SET POINTER TO LOAD ROUTINE
005066  004767  007710                  JSR     PC,PRIME        ;GO SET UP BUFFERS AND DEVICE
005072  005767  174324          2$:     TST     XFLGO           ;ANY HARD XMIT ERRORS ??
005076  001040                          BNE     5$              ;BR IF YES
005100  005767  174320                  TST     RFLGO           ;ANY HARD RECEIVER ERROR ??
005104  001053                          BNE     7$              ;BR IF YES
005106  005767  174314                  TST     RFLG1           ;ANY SOFT RECEIVER ERRORS ??
005112  001065                          BNE     9$              ;BR IF YES
005114  022767  023040  174312          CMP     #BUFEND,IPTR    ;RECEIVED 256. BYTES ??
005122  001003                          BNE     3$              ;BR IF NOT
005124  004767  010144                  JSR     PC,CHKDAT       ;GO CHECK THE DATA BUFFERS
005130  000500                          BR      TST23           ;;<GO TO NEXT TEST>
005132  005367  174302          3$:     DEC     TIMR1           ;DEC TIMEOUT COUNTER 1
005136  001355                          BNE     2$              ;BR IF NO TIMEOUT
005140  005367  174276                  DEC     TIMR2           ;DEC TIMEOUT COUNTER 2
005144  001352                          BNE     2$              ;BR IF NO TIMEOUT
005146  042777  000100  174234          BIC     #100,@DLRCSR    ;TURN OFF THE INTRS.
005154  042777  000104  174232          BIC     #104,@DLXCSR
005162  104401  017110                  TYPE    ,XMSG1          ;GO TYPE TIMEOUT MESSAGE
005166  012767  005176  174050          MOV     #4$,$ESCAPE     ;GO TO 4$ AFTER ERROR PRINT
005174  104000                          ERROR                   ;PRINT ERROR PC
005176                          4$:
005176  000455                          BR      TST23           ;;<GO TO NEXT TEST>
005200  016701  174204          5$:     MOV     DLRCSR,R1       ;PUT DEVADR IN R1
005204  016702  174204                  MOV     DLXCSR,R2       ;PUT REGADR IN R2
005210  011203                          MOV     (R2),R3         ;GET THE WAS DATA
005212  012704  000204                  MOV     #204,R4         ;PUT S/B DATA IN R4
005216  004767  007424                  JSR     PC,SUERR1       ;GO SET UP ERROR INFO
005222  012767  005232  174014          MOV     #6$,$ESCAPE     ;GO TO 6$ AFTER PRINTING ERROR
005230  104002                          ERROR+2                 ;TRANSMITTER FALSE INTERRUPT
005232                          6$:
005232  000437                          BR      TST23           ;;<GO TO NEXT TEST>
005234  016701  174150          7$:     MOV     DLRCSR,R1       ;SAVE THE DEVADR
005240  010102                          MOV     R1,R2           ;SAVE THE REGADR
005242  011203                          MOV     (R2),R3         ;GET THE WAS DATA
005244  012704  000200                  MOV     #200,R4         ;RESULT S/B = 200
005250  004767  007372                  JSR     PC,SUERR1       ;GO SET UP ERROR INFO
005254  012767  005264  173762          MOV     #8$,$ESCAPE     ;GO TO 8$ AFTER ERROR PRINT
005262  104002                          ERROR+2                 ;RECEIVER FALSE INTERRUPT
005264                          8$:
005264  000422                          BR      TST23           ;;<GO TO NEXT TEST>
005266  016701  174116          9$:     MOV     DLRCSR,R1       ;SAVE THE DEVADR
005272  016702  174114                  MOV     DLRDBR,R2       ;SAVE REGADR
005276  016703  173702                  MOV     $TMP1,R3        ;GET CONTENTS OF ERROR RDBR
005302  004767  007340                  JSR     PC,SUERR1       ;GO SETUP ERROR INFO
005306  012767  005316  173730          MOV     #10$,$ESCAPE    ;GO TO 10$ AFTER ERROR PRINT
005314  104005                          ERROR+5                 ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN
005316  005267  174106          10$:    INC     RTRY            ;COUNT ONE TRY
005322  022767  000003  174100          CMP     #3,RTRY         ;TRIED THREE TIMES
```

```
 2383  005330  001253                    BNE     1$              ;BR IF NOT
```

```
                                ;;*****************************************************************
                                ;*TEST 23       TEST TO TURN AROUND BINARY UP COUNT PATTERN
                                ;;*****************************************************************
005332  000004                  TST23:  SCOPE
005334  012767  000001  173700          MOV     #1,$TIMES       ;;DO 1 ITERATION
005342  004767  007406          JSR     PC,SUVEC        ;GO SET UP VECTORS
005346  005067  174056          CLR     RTRY            ;INITIALIZE RETRY FLAG
005352  012767  015152  174056  1$:     MOV     #LDOUT2,LDOUT   ;SET POINTER TO LOAD ROUTINE
005360  004767  007416          JSR     PC,PRIME        ;GO SET UP BUFFERS AND DEVICE
005364  005767  174032          2$:     TST     XFLGO           ;ANY HARD XMIT ERRORS ??
005370  001040                  BNE     5$              ;BR IF YES
005372  005767  174026          TST     RFLGO           ;ANY HARD RECEIVER ERROR ??
005376  001053                  BNE     7$              ;BR IF YES
005400  005767  174022          TST     RFLG1           ;ANY SOFT RECEIVER ERRORS ??
005404  001065                  BNE     9$              ;BR IF YES
005406  022767  023040  174020  CMP     #BUFEND,IPTR    ;RECEIVED 256. BYTES ??
005414  001003                  BNE     3$              ;BR IF NOT
005416  004767  007652          JSR     PC,CHKDAT       ;GO CHECK THE DATA BUFFERS
005422  000500                  BR      TST24           ;;<GO TO NEXT TEST>
005424  005367  174010          3$:     DEC     TIMR1           ;DEC TIMEOUT COUNTER 1
005430  001355                  BNE     2$              ;BR IF NO TIMEOUT
005432  005367  174004          DEC     TIMR2           ;DEC TIMEOUT COUNTER 2
005436  001352                  BNE     2$              ;BR IF NO TIMEOUT
005440  042777  000100  173742  BIC     #100,@DLRCSR    ;TURN OFF THE INTRS.
005446  042777  000104  173740  BIC     #104,@DLXCSR
005454  104401  017167          TYPE    ,XMSG2          ;GO TYPE TIMEOUT MESSAGE
005460  012767  005470  173556  MOV     #4$,$ESCAPE     ;GO TO 4$ AFTER ERROR PRINT
005466  104000                  ERROR                   ;PRINT ERROR PC
005470                          4$:
005470  000455                  BR      TST24           ;;<GO TO NEXT TEST>
005472  016701  173712          5$:     MOV     DLRCSR,R1       ;PUT DEVADR IN R1
005476  016702  173712          MOV     DLXCSR,R2       ;PUT REGADR IN R2
005502  011203                  MOV     (R2),R3         ;GET THE WAS DATA
005504  012704  000204          MOV     #204,R4         ;PUT S/B DATA IN R4
005510  004767  007132          JSR     PC,SUERR1       ;GO SET UP ERROR INFO
005514  012767  005524  173522  MOV     #6$,$ESCAPE     ;GO TO 6$ AFTER PRINTING ERROR
005522  104002                  ERROR+2                 ;TRANSMITTER FALSE INTERRUPT
005524                          6$:
005524  000437                  BR      TST24           ;;<GO TO NEXT TEST>
005526  016701  173656          7$:     MOV     DLRCSR,R1       ;SAVE THE DEVADR
005532  010102                  MOV     R1,R2           ;SAVE THE REGADR
005534  011203                  MOV     (R2),R3         ;GET THE WAS DATA
005536  012704  000200          MOV     #200,R4         ;RESULT S/B = 200
005542  004767  007100          JSR     PC,SUERR1       ;GO SET UP ERROR INFO
005546  012767  005556  173470  MOV     #8$,$ESCAPE     ;GO TO 8$ AFTER ERROR PRINT
005554  104002                  ERROR+2                 ;RECEIVER FALSE INTERRUPT
005556                          8$:
005556  000422                  BR      TST24           ;;<GO TO NEXT TEST>
005560  016701  173624          9$:     MOV     DLRCSR,R1       ;SAVE THE DEVADR
005564  016702  173622          MOV     DLRDBR,R2       ;SAVE REGADR
005570  016703  173410          MOV     $TMP1,R3        ;GET CONTENTS OF ERROR RDBR
005574  004767  007046          JSR     PC,SUERR1       ;GO SETUP ERROR INFO
005600  012767  005610  173436  MOV     #10$,$ESCAPE    ;GO TO 10$ AFTER ERROR PRINT
005606  104005                  ERROR+5                 ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN
005610  005267  173614          10$:    INC     RTRY            ;COUNT ONE TRY
005614  022767  000003  173606  CMP     #3,RTRY         ;TRIED THREE TIMES
```

```
 2440  005622  001253                    BNE     1$              ;BR IF NOT
```

```
                                        ;;*****************************************************************
                                        ;*TEST 24       TEST TO TURN AROUND BINARY DOWN COUNT PATTERN
                                        ;;*****************************************************************
005624  000004                  TST24:  SCOPE
005626  012767  000001  173406          MOV     #1,$TIMES       ;;DO 1 ITERATION
005634  004767  007114                  JSR     PC,SUVEC        ;GO SET UP VECTORS
005640  005067  173564                  CLR     RTRY            ;INITIALIZE RETRY FLAG
005644  012767  015172  173564  1$:     MOV     #LDOUT3,LDOUT   ;SET POINTER TO LOAD ROUTINE
005652  004767  007124                  JSR     PC,PRIME        ;GO SET UP BUFFERS AND DEVICE
005656  005767  173540          2$:     TST     XFLG0           ;ANY HARD XMIT ERRORS ??
005662  001040                          BNE     5$              ;BR IF YES
005664  005767  173534                  TST     RFLG0           ;ANY HARD RECEIVER ERROR ??
005670  001053                          BNE     7$              ;BR IF YES
005672  005767  173530                  TST     RFLG1           ;ANY SOFT RECEIVER ERRORS ??
005676  001065                          BNE     9$              ;BR IF YES
005700  022767  023040  173526          CMP     #BUFEND,IPTR    ;RECEIVED 256. BYTES ??
005706  001003                          BNE     3$              ;BR IF NOT
005710  004767  007360                  JSR     PC,CHKDAT       ;GO CHECK THE DATA BUFFERS
005714  000500                          BR      TST25           ;;<GO TO NEXT TEST>
005716  005367  173516          3$:     DEC     TIMR1           ;DEC TIMEOUT COUNTER 1
005722  001355                          BNE     2$              ;BR IF NO TIMEOUT
005724  005367  173512                  DEC     TIMR2           ;DEC TIMEOUT COUNTER 2
005730  001352                          BNE     2$              ;BR IF NO TIMEOUT
005732  042777  000100  173450          BIC     #100,@DLRCSR    ;TURN OFF THE INTRS.
005740  042777  000104  173446          BIC     #104,@DLXCSR
005746  104401  017250                  TYPE    ,XMSG3          ;GO TYPE TIMEOUT MESSAGE
005752  012767  005762  173264          MOV     #4$,$ESCAPE     ;GO TO 4$ AFTER ERROR PRINT
005760  104000                          ERROR                   ;PRINT ERROR PC
005762                          4$:
005762  000455                          BR      TST25           ;;<GO TO NEXT TEST>
005764  016701  173420          5$:     MOV     DLRCSR,R1       ;PUT DEVADR IN R1
005770  016702  173420                  MOV     DLXCSR,R2       ;PUT REGADR IN R2
005774  011203                          MOV     (R2),R3         ;GET THE WAS DATA
005776  012704  000204                  MOV     #204,R4         ;PUT S/B DATA IN R4
006002  004767  006640                  JSR     PC,SUERR1       ;GO SET UP ERROR INFO
006006  012767  006016  173230          MOV     #6$,$ESCAPE     ;GO TO 6$ AFTER PRINTING ERROR
006014  104002                          ERROR+2                 ;TRANSMITTER FALSE INTERRUPT
006016                          6$:
006016  000437                          BR      TST25           ;;<GO TO NEXT TEST>
006020  016701  173364          7$:     MOV     DLRCSR,R1       ;SAVE THE DEVADR
006024  010102                          MOV     R1,R2           ;SAVE THE REGADR
006026  011203                          MOV     (R2),R3         ;GET THE WAS DATA
006030  012704  000200                  MOV     #200,R4         ;RESULT S/B = 200
006034  004767  006606                  JSR     PC,SUERR1       ;GO SET UP ERROR INFO
006040  012767  006050  173176          MOV     #8$,$ESCAPE     ;GO TO 8$ AFTER ERROR PRINT
006046  104002                          ERROR+2                 ;RECEIVER FALSE INTERRUPT
006050                          8$:
006050  000422                          BR      TST25           ;;<GO TO NEXT TEST>
006052  016701  173332          9$:     MOV     DLRCSR,R1       ;SAVE THE DEVADR
006056  016702  173330                  MOV     DLRDBR,R2       ;SAVE REGADR
006062  016703  173116                  MOV     $TMP1,R3        ;GET CONTENTS OF ERROR RDBR
006066  004767  006554                  JSR     PC,SUERR1       ;GO SETUP ERROR INFO
006072  012767  006102  173144          MOV     #10$,$ESCAPE    ;GO TO 10$ AFTER ERROR PRINT
006100  104005                          ERROR+5                 ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN
006102  005267  173322          10$:    INC     RTRY            ;COUNT ONE TRY
006106  022767  000003  173314          CMP     #3,RTRY         ;TRIED THREE TIMES
```

```
 2497  006114  001253                    BNE     1$              ;BR IF NOT
```

```
2498                                    ;;*****************************************************************
2499                                    ;*TEST 25        TEST TO TURN AROUND WORST CASE PATTERN
2500                                    ;;*****************************************************************
2501  006116  000004                    TST25:  SCOPE
2502  006120  012767  000001  173114            MOV     #1,$TIMES       ;;DO 1 ITERATION
2503  006126  004767  006622                    JSR     PC,SUVEC        ;GO SET UP VECTORS
2504  006132  005067  173272                    CLR     RTRY            ;INITIALIZE RETRY FLAG
2505  006136  012767  015226  173272  1$:       MOV     #LDOUT4,LDOUT   ;SET POINTER TO LOAD ROUTINE
2506  006144  004767  006632                    JSR     PC,PRIME        ;GO SET UP BUFFERS AND DEVICE
2507  006150  005767  173246          2$:       TST     XFLGO           ;ANY HARD XMIT ERRORS ??
2508  006154  001042                            BNE     5$              ;BR IF YES
2509  006156  005767  173242                    TST     RFLGO           ;ANY HARD RECEIVER ERROR ??
2510  006162  001056                            BNE     7$              ;BR IF YES
2511  006164  005767  173236                    TST     RFLG1           ;ANY SOFT RECEIVER ERRORS ??
2512  006170  001071                            BNE     9$              ;BR IF YES
2513  006172  022767  023040  173234            CMP     #BUFEND,IPTR    ;RECEIVED 256. BYTES ??
2514  006200  001004                            BNE     3$              ;BR IF NOT
2515  006202  004767  007066                    JSR     PC,CHKDAT       ;GO CHECK THE DATA BUFFERS
2516  006206  000167  001642                    JMP             $EOP            ;GO TO NEXT TEST
2517  006212  005367  173222          3$:       DEC     TIMR1           ;DEC TIMEOUT COUNTER 1
2518  006216  001354                            BNE     2$              ;BR IF NO TIMEOUT
2519  006220  005367  173216                    DEC     TIMR2           ;DEC TIMEOUT COUNTER 2
2520  006224  001351                            BNE     2$              ;BR IF NO TIMEOUT
2521  006226  042777  000100  173154            BIC     #100,@DLRCSR    ;TURN OFF THE INTRS.
2522  006234  042777  000104  173152            BIC     #104,@DLXCSR
2523  006242  104401  017333                    TYPE    ,XMSG4          ;GO TYPE TIMEOUT MESSAGE
2524  006246  012767  006256  172770            MOV     #4$,$ESCAPE     ;GO TO 4$ AFTER ERROR PRINT
2525  006254  104000                            ERROR                   ;PRINT ERROR PC
2526  006256  000167  001572          4$:       JMP     $EOP            ;GO TO NEXT TEST
2527  006262  016701  173122          5$:       MOV     DLRCSR,R1       ;PUT DEVADR IN R1
2528  006266  016702  173122                    MOV     DLXCSR,R2       ;PUT REGADR IN R2
2529  006272  011203                            MOV     (R2),R3         ;GET THE WAS DATA
2530  006274  012704  000204                    MOV     #204,R4         ;PUT S/B DATA IN R4
2531  006300  004767  006342                    JSR     PC,SUERR1       ;GO SET UP ERROR INFO
2532  006304  012767  006314  172732            MOV     #6$,$ESCAPE     ;GO TO 6$ AFTER PRINTING ERROR
2533  006312  104002                            ERROR+2                 ;TRANSMITTER FALSE INTERRUPT
2534  006314  000167  001534          6$:       JMP     $EOP            ;GO TO NEXT TEST
2535  006320  016701  173064          7$:       MOV     DLRCSR,R1       ;SAVE THE DEVADR
2536  006324  010102                            MOV     R1,R2           ;SAVE THE REGADR
2537  006326  011203                            MOV     (R2),R3         ;GET THE WAS DATA
2538  006330  012704  000200                    MOV     #200,R4         ;RESULT S/B = 200
2539  006334  004767  006306                    JSR     PC,SUERR1       ;GO SET UP ERROR INFO
2540  006340  012767  006350  172676            MOV     #8$,$ESCAPE     ;GO TO 8$ AFTER ERROR PRINT
2541  006346  104002                            ERROR+2                 ;RECEIVER FALSE INTERRUPT
2542  006350  000167  001500          8$:       JMP     $EOP            ;GO TO NEXT TEST
2543  006354  016701  173030          9$:       MOV     DLRCSR,R1       ;SAVE THE DEVADR
2544  006360  016702  173026                    MOV     DLRDBR,R2       ;SAVE REGADR
2545  006364  016703  172614                    MOV     $TMP1,R3        ;GET CONTENTS OF ERROR RDBR
2546  006370  004767  006252                    JSR     PC,SUERR1       ;GO SETUP ERROR INFO
2547  006374  012767  006404  172642            MOV     #10$,$ESCAPE    ;GO TO 10$ AFTER ERROR PRINT
2548  006402  104005                            ERROR+5                 ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN
2549  006404  005267  173020          10$:      INC     RTRY            ;COUNT ONE TRY
2550  006410  022767  000003  173012            CMP     #3,RTRY         ;TRIED THREE TIMES
2551  006416  001247                            BNE     1$              ;BR IF NO
2552  006420  000167  001430                    JMP     $EOP            ;GO TO END OF PASS ROUTINE
2553
```

```
                                ;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
                                ;
006424  000240                  PRG2:   NOP
006426  104401  017460                  TYPE    ,PROG2M         ;INDICATE THAT USER SELECTED
                                                                ;PROGRAM #2
006432  104401  021102          PRG2A:  TYPE    ,LINTAD         ;ASK USER FOR THE TRANSMITTER
                                                                ;DATA BUFFER ADDRESS OF THE DEVICE
                                                                ;HE WISHES TO TEST
006436  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
                                                                ;AND STORE ON TOP OF STACK
                                ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
006440  012602                          MOV     (SP)+,R2        ;GET THE ANSWER TYPED
006442  020227  176176                  CMP     R2,#176176      ;IS THE NUMBER TOO HIGH?
006446  101065                          BHI     REDO1           ;IF YES - GO TO RETRY SITUATION
006450  020227  175616                  CMP     R2,#175616      ;IS THE NUMBER TOO LOW?
006454  103462                          BLO     REDO1           ;IF YES - GO TO RETRY SITUATION .
006456  132702  000001                  BITB    #BIT0,R2        ;NUMBER IS IN RANGE BUT IS IT
                                                                ;ON AN EVEN BOUNDARY?
006462  001057                          BNE     REDO1           ;IF NOT GO TO RETRY SITUATION
                                ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
006464  010203                          MOV     R2,R3           ;GET THE USER RESPONSE
006466  142703  000370                  BICB    #370,R3         ;MASK OFF LOWER BYTE EXCEPT FOR
                                                                ;LEAST SIGNIFICANT DIGIT
006472  122703  000006                  CMPB    #6,R3           ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
                                                                ;USER RESPONSE EQUAL TO A SIX?
006476  001051                          BNE     REDO1           ;BRANCH IF NOT
006500  010267  172476                  MOV     R2,$TMP0        ;THE TRANSMITTER ADDRESS
                                                                ;TYPED IS OK - STORE FOR
                                                                ;FUTURE USE
                                ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
006504  016746  171274                  MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
006510  012767  006522  171266          MOV     #2$,ERRVEC      ;SET UP TIMEOUT SERVICE ADDRESS
006516  005712                          TST     (R2)            ;IF PRESENT WE WILL EXECUTE THE
                                                                ;NEXT INSTRUCTION - IF NOT
                                                                ;WE GO TO 2$:
006520  000412                          BR      4$              ;BRANCH IF PRESENT
006522  004767  006200          2$:     JSR     PC,SUERT2       ;GO SET UP FOR ERROR INFORMATION
006526  012767  006536  172510          MOV     #3$,$ESCAPE     ;POINT OF RETURN AFTER ERROR REPORT
006534  104006                          ERROR   +6              ;XDBR REFERENCE CAUSED TIMEOUT
006536  022626                  3$:     CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
006540  012667  171240                  MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
006544  000426                          BR      REDO1           ;GO TO RETRY SITUATION
006546  012667  171232          4$:     MOV     (SP)+,ERRVEC    ;DEVICE REGISTER IS PRESENT!
                                                                ;RESTORE TIMEOUT VECTOR
                                ;WE ARE NOW READY FOR THE CHARACTER TO BE TRANSMITTED, AND THE
                                ;DELAY TIME (IN MILLISECONDS) THAT IS TO TRANSPIRE BETWEEN
                                ;SUCCESSIVE CHARACTER TRANSFERS
006552  104401  021163          PRG2B:  TYPE    ,SELCAR         ;ASK USER FOR THE CHARACTER HE
                                                                ;WISHES TO TRANSFER
```

```
2610  006556  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY
2611                                                                    ;USER AND STORE ON TOP OF STACK
2612  006560  012667  172420                  MOV     (SP)+,$TMP1     ;GET THE ANSWER TYPED
2613                                  ;NOTE:  THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE
2614                                  ;       OCTAL ASCII EQUIVALENT OF THE CHARACTER E.G. A=101
2615  006564  104401  021271                  TYPE    ,SELDLY         ;ASK THE USER FOR THE DELAY
2616                                                                    ;IN MSEC (OCTAL NO.) BETWEEN
2617                                                                    ;CHARACTER TRANSFERS
2618  006570  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY
2619                                                                    ;USER AND STORE ON TOP OF STACK
2620  006572  012667  172410                  MOV     (SP)+,$TMP2     ;GET THE ANSWER TYPED
2621  006576  116767  172404  000012  1$:     MOVB    $TMP2,2$        ;SET THE DELAY COUNT ARGUMENT
2622                                                                    ;FOR TIMER ROUTINE
2623  006604  116777  172374  172370          MOVB    $TMP1,@$TMP0    ;LOAD THE TRANSMITTER DATA
2624                                                                    ;BUFFER WITH THE CHARACTER
2625  006612  004767  005470                  JSR     PC,DELAY        ;GO OFF TO WAIT THE SPECIFIED
2626
2627
2628
2629
2630
2631                                                                    ;NO. OF MSEC. BEFORE ISSUING
2632                                                                    ;ANOTHER CHARACTER
2633  006616  000000                  2$:     .WORD   0               ;THIS IS WHERE THE DELAY COUNT RESIDES
2634  006620  000766                          BR      1$              ;GO BACK TO ISSUE ANOTHER CHARACTER
2635  006622  104401  001252          REDO1:  TYPE    ,$QUES          ;TYPE A QUESTION MARK(?)
2636  006626  000167  177600                  JMP     PRG2A           ;REITERATE THE XDBR QUESTION TO USER
2637                                  ;THIS IS PROGRAM #3
2638                                  ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
2639                                  ;       A)  SELECTION OF A TRANSMITTER DATA BUFFER
2640                                  ;       B)  SELECTION OF A CHARACTER FOR CONTINUOUS TRANSFER
2641                                  ;           IN MAINTENANCE MODE
2642                                  ;       C)  SELECTION OF AN EXPIRATION TIME IN MILLISECONDS
2643                                  ;           BETWEEN EACH TRANSMITTER DATA BUFFER CHARACTER TRANSFER
2644                                  ;       D)  A TIGHT SCOPE LOOP LOCK ON A SPECIFIC CHARACTER
2645                                  ;
2646
2647  006632  000240                  PRG3:   NOP
2648  006634  104401  017524                  TYPE    ,PROG3M         ;INDICATE THAT USER SELECTED
2649                                                                    ;PROGRAM #3
2650  006640  104401  021102          PRG3A:  TYPE    ,LINTAD         ;ASK USER FOR THE TRANSMITTER DATA
2651                                                                    ;BUFFER ADDRESS OF THE DEVICE
2652                                                                    ;HE WISHES TO TEST
2653  006644  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY
2654                                                                    ;USER AND STORE ON TOP OF STACK
2655                                  ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
2656  006646  012602                          MOV     (SP)+,R2        ;GET THE ANSWER TYPED
2657  006650  020227  176176                  CMP     R2,#176176      ;IS THE NUMBER TOO HIGH?
2658  006654  101071                          BHI     REDO2           ;IF YES - GO TO RETRY SITUATION
2659  006656  020227  175616                  CMP     R2,#175616      ;IS THE NUMBER TOO LOW?
2660  006662  103466                          BLO     REDO2           ;IF YES - GO TO RETRY SITUATION
2661  006664  132702  000001                  BITB    #BIT0,R2        ;NUMBER IS IN RANGE BUT IS IT
2662                                                                    ;ON AN EVEN BOUNDARY?
2663  006670  001063                          BNE     REDO2           ;IF NOT - GO TO RETRY SITUATION
2664                                  ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XDBR DBR ADDRESS
2665  006672  010203                          MOV     R2,R3           ;GET THE USER RESPONSE
```

```
2666  006674  142703  000370                  BICB    #370,R3          ;MASK OFF LOWER BYTE EXCEPT FOR
2667                                                                     ;LEAST SIGNIFICANT DIGIT
2668  006700  122703  000006                  CMPB    #6,R3            ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
2669                                                                     ;USER RESPONSE EQUAL TO A TWO?
2670  006704  001055                          BNE     REDO2            ;BRANCH IF NOT
2671  006706  010267  172270                  MOV     R2,$TMP0         ;THE TRANSMITTER ADDRESS TYPED IS
2672                                                                     ;OK - STORE FOR FUTURE USE
2673                                  ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
2674  006712  016746  171066                  MOV     ERRVEC,-(SP)     ;SAVE THE TIMEOUT VECTOR
2675  006716  012767  006730  171060          MOV     #2$,ERRVEC       ;SET UP TIMEOUT SERVICE ADDRESS
2676  006724  005712                          TST     (R2)             ;IF PRESENT WE WILL EXECUTE THE
2677                                                                     ;NEXT INSTRUCTION - IF NOT WE
2678                                                                     ;GO TO 2$:
2679  006726  000412                          BR      4$               ;BRANCH IF PRESENT
2680  006730  004767  005772          2$:     JSR     PC,SUERT2        ;GO SET UP FOR ERROR INFORMATION
2681  006734  012767  006744  172302          MOV     #3$,$ESCAPE      ;POINT OF RETURN AFTER ERROR REPORT
2682  006742  104006                          ERROR   +6               ;XDBR REFERENCE CAUSED TIMEOUT
2683  006744  022626                  3$:     CMP     (SP)+,(SP)+      ;CLEAN STACK FROM TIMEOUT
2684  006746  012667  171032                  MOV     (SP)+,ERRVEC     ;RESTORE TIMEOUT VECTOR
2685  006752  000432                          BR      REDO2            ;GO TO RETRY SITUATION
2686  006754  012667  171024          4$:     MOV     (SP)+,ERRVEC     ;DEVICE REGISTER IS PRESENT!
2687                                                                     ;RESTORE TIMEOUT VECTOR
2688                                  ;WE ARE NOW READY FOR THE CHARACTER TO BE TRANSMITTED, AND THE
2689                                  ;DELAY TIME (IN MILLISECONDS) THAT IS TO TRANSPIRE BETWEEN SUCCESSIVE
2690                                  ;CHARACTER TRANSFERS
2691  006760  104401  021163          PRG3B:  TYPE    ,SELCAR          ;ASK USER FOR THE CHARACTER
2692                                                                     ;HE WISHES TO TRANSFER
2693  006764  104412                          RDOCT                    ;ACCEPT THE ANSWER TYPED BY USER
2694                                                                     ;AND STORE ON TOP OF STACK
2695  006766  012667  172212                  MOV     (SP)+,$TMP1      ;GET THE ANSWER TYPED
2696                                  ;NOTE:  THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE
2697                                  ;       OCTAL ASCII EQUIVALENT OF THE CHARACTER E.G. B=102
2698  006772  104401  021271                  TYPE    ,SELDLY          ;ASK THE USER FOR THE DELAY
2699                                                                     ;IN MSEC (OCTAL NO.) BETWEEN
2700                                                                     ;CHARACTER TRANSFERS
2701  006776  104412                          RDOCT                    ;ACCEPT THE ANSWER TYPED BY
2702                                                                     ;USER AND STORE ON TOP OF STACK
2703  007000  012667  172202                  MOV     (SP)+,$TMP2      ;GET THE ANSWER TYPED
2704  007004  162702  000002                  SUB     #2,R2            ;GET THE CORRESPONDING XCSR
2705                                                                     ;ADDRESS FOR TRANSMITTER UNDER-
2706                                                                     ;GOING TEST
2707  007010  052712  000004          1$:     BIS     #BIT2,(R2)       ;SET MAINTENANCE BIT IN XCSR
2708  007014  116767  172166  000012          MOVB    $TMP2,2$         ;SET THE DELAY COUNT ARGUMENT
2709                                                                     ;FOR TIMER ROUTINE
2710  007022  116777  172156  172152          MOVB    $TMP1,@$TMP0     ;LOAD THE TRANSMITTER DATA BUFFER
2711                                                                     ;WITH THE CHARACTER
2712  007030  004767  005252                  JSR     PC,DELAY         ;GO OFF TO WAIT THE SPECIFIED
2713                                                                     ;NO. OF MSEC. BEFORE ISSUING
2714                                                                     ;ANOTHER CHARACTER
2715  007034  000000          2$:     .WORD   0                        ;THIS IS WHERE THE DELAY COUNT RESIDES
2716  007036  000764                  BR      1$                       ;GO BACK TO ISSUE ANOTHER CHARACTER
2717  007040  104401  001252          REDO2:  TYPE    ,$QUES           ;TYPE A QUESTION MARK(?)
2718  007044  000167  177570                  JMP     PRG3A            ;REITERATE THE XDBR QUESTION TO
2719                                                                     ;USER
2720
2721                                  ;THIS IS PROGRAM #4
```

```
                                ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
                                ;       A)  SELECTION OF A TRANSMITTER DATA BUFFER
                                ;       B)  SELECTION OF A SINGLE CHARACTER TO BE SENT, RECEIVED
                                ;           AND CHECKED WITH MAINTENANCE BIT SET
                                ;
                                ;
007050  000240                  PRG4:   NOP
007052  104401  017570                  TYPE    ,PROG4M         ;INDICATE THAT USER SELECTED
                                                                ;PROGRAM #4
007056  104401  021102          PRG4A:  TYPE    ,LINTAD         ;ASK USER FOR THE TRANSMITTER
                                                                ;DATA BUFFER ADDRESS OF THE
                                                                ;DEVICE HE WISHES TO TEST
007062  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY
                                                                ;USER AND STORE ON TOP OF STACK
                                ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
007064  012602                          MOV     (SP)+,R2        ;GET THE ANSWER TYPED
007066  020227  176176                  CMP     R2,#176176      ;IS THE NUMBER TOO HIGH?
007072  101136                          BHI     REDO3           ;IF YES - GO TO RETRY SITUATION
007074  020227  175616                  CMP     R2,#175616      ;IS THE NUMBER TOO LOW?
007100  103533                          BLO     REDO3           ;IF YES - GO TO RETRY SITUATION
007102  132702  000001                  BITB    #BIT0,R2        ;NUMBER IS IN RANGE BUT IS IT
                                                                ;ON AN EVEN BOUNDARY?
007106  001130                          BNE     REDO3           ;IF NO - GO TO RETRY SITUATION
                                ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
007110  010203                          MOV     R2,R3           ;GET THE USER RESPONSE
007112  142703  000370                  BICB    #370,R3         ;MASK OFF LOWER BYTE EXCEPT FOR
                                                                ;LEAST SIGNIFICANT DIGIT
007116  122703  000006                  CMPB    #6,R3           ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
                                                                ;USER RESPONSE EQUAL TO A SIX?
007122  001122                          BNE     REDO3           ;BRANCH IF NOT
007124  010267  172052                  MOV     R2,$TMP0        ;THE TRANSMITTER ADDRESS TYPED
                                                                ;IS OK - STORE FOR FUTURE USE
                                ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
007130  016746  170650                  MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
007134  012767  007146  170642          MOV     #2$,ERRVEC      ;SET UP TIMEOUT SERVICE ADDRESS
007142  005712                          TST     (R2)            ;IF PRESENT WE WILL EXECUTE THE
                                                                ;NEXT INSTRUCTION - IF NOT WE
                                                                ;GO TO 2$:
007144  000412                          BR      4$              ;BRANCH IF PRESENT
007146  004767  005554          2$:     JSR     PC,SUERT2       ;GO SET UP FOR ERROR INFORMATION
007152  012767  007162  172064          MOV     #3$,$ESCAPE     ;POINT OF RETURN AFTER ERROR REPORT
007160  104006                          ERROR   +6              ;XDBR REFERENCE CAUSED TIMEOUT
007162  022626                  3$:     CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
007164  012667  170614                  MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
007170  000477                          BR      REDO3           ;GO TO RETRY SITUATION
007172  012667  170606          4$:     MOV     (SP)+,ERRVEC    ;DEVICE REGISTER IS PRESENT!
                                                                ;RESTORE TIMEOUT VECTOR
007176  104401  021364                  TYPE    ,RSTALL         ;ASK THE USER IF HE DESIRES SOME
                                                                ;RANDOM NO. OF MSEC. WAIT TIME
                                                                ;BEFORE CHECKING FOR XCSR DONE
                                                                ;FLAG
007202  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
                                                                ;AND STORE ON TOP OF STACK
007204  012667  171776                  MOV     (SP)+,$TMP2     ;GET THE ANSWER TYPED
                                ;WE ARE NOW READY FOR THE CHARACTER TO BE TRANSMITTED
007210  104401  021163          PRG4B:  TYPE    ,SELCAR         ;ASK USER FOR THE CHARACTER HE
```

```
2778                                                    ;WISHES TO TRANSFER
2779  007214  104412                    RDOCT           ;ACCEPT THE ANSWER TYPED BY USER
2780                                                    ;AND STORE ON TOP OF STACK
2781  007216  012667  171762            MOV     (SP)+,$TMP1     ;GET THE ANSWER TYPED
2782                            ;NOTE:  THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE OCTAL
2783                            ;       ASCII EQUIVALENT OF THE CHARACTER E.G. C=103
2784                            ;
2785  007222  104401  020013    PRG4C:  TYPE,   LENGTH  ;ASK USER FOR THE CHARACTER LENGTH
2786                                                    ;FOR WHICH HIS DEVICE IS SET
2787  007226  104413                    RDDEC           ;ACCEPT THE ANSWER TYPED BY USER
2788                            ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
2789  007230  012600                    MOV     (SP)+,R0        ;GET THE ANSWER TYPED
2790  007232  020027  000010            CMP     R0,#8.          ;IS THE NUMBER TOO HIGH?
2791  007236  101060                    BHI     REDO3A          ;IF YES - GO TO RETRY SITUATION
2792  007240  020027  000005            CMP     R0,#5.          ;IS THE NUMBER TOO LOW?
2793  007244  103455                    BLO     REDO3A          ;IF YES - GO TO RETRY SITUATION
2794  007246  010067  171762            MOV     R0,$TMP15       ;THE VALUE TYPED IS OK
2795                                                    ;STORE FOR FUTURE USE
2796  007252  016767  171724  171730    MOV     $TMP0,$TMP3     ;GET THE XDBR ADDRESS
2797  007260  162767  000002  171722    SUB     #2,$TMP3        ;FORM THE XCSR ADDRESS
2798  007266  005767  171714    1$:     TST     $TMP2           ;DO WE RANDOM STALL?
2799  007272  001402                    BEQ     2$              ;BRANCH IF IT WASN'T DESIRED
2800  007274  004767  005052            JSR     PC,STALL        ;GO STALL RANDOM VALUE OF MSEC.
2801  007300  004767  005156    2$:     JSR     PC,TIMETX       ;GO WAIT FOR TRANSMITTER DONE
2802                                                    ;BIT TO SET
2803  007304  104401  017700            TYPE    ,XDB            ;TYPE TRANSMITTER DONE BIT MESSAGE
2804  007310  104000                    ERROR   +0              ;XCSR DONE BIT NEVER SET
2805  007312  052777  000004  171670    BIS     #BIT2,@$TMP3    ;SET THE MAINTENANCE BIT IN THE
2806                                                    ;TRANSMITTER CONTROL STATUS REGISTER
2807  007320  016777  171660  171654    MOV     $TMP1,@$TMP0    ;LOAD TRANSMITTER DATA BUFFER
2808                                                    ;WITH SELECTED CHARACTER
2809  007326  004767  005112            JSR     PC,TIMERX       ;GO WAIT FOR RECEIVER DONE BIT
2810                                                    ;TO SET
2811  007332  104401  017747            TYPE    ,RDB            ;TYPE RECEIVER DONE BIT MESSAGE
2812  007336  104000                    ERROR   +0              ;RCSR DONE BIT NEVER SET
2813  007340  016767  171644  171644    MOV     $TMP3,$TMP4     ;GET THE TRANSMITTER CONTROL
2814                                                    ;STATUS REGISTER ADDRESS
2815  007346  162767  000002  171636    SUB     #2,$TMP4        ;FORM THE RECEIVER DATA BUFFER
2816                                                    ;ADDRESS
2817  007354  017767  171632  171632    MOV     @$TMP4,$TMP5    ;STORE THE CHARACTER FROM THE
2818                                                    ;RECEIVER BUFFER + REST OF CONTENTS
2819  007362  004767  005132            JSR     PC,DATCHK       ;GO TO COMPARE EXPECTED & RECEIVED
2820                                                    ;DATA
2821  007366  000737                    BR      1$              ;GO BACK TO ISSUE ANOTHER CHARACTER
2822  007370  104401  001252    REDO3:  TYPE    ,$QUES          ;TYPE A QUESTION MARK(?)
2823  007374  000167  177456            JMP     PRG4A           ;REITERATE THE XDBR QUESTION TO USER
2824  007400  104401  001252    REDO3A: TYPE,   $QUES           ;TYPE '?' INDICATING USER TYPED
2825                                                    ;SOMETHING WRONG FOR CHARACTER LENGTH
2826  007404  000167  177612            JMP     PRG4C           ;GO BACK TO REISSUE QUESTION
2827
2828                            ;THIS IS PROGRAM #5
2829                            ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW USER PARAMETERS
2830                            ;FOR RUNNING A BINARY COUNT IN MAINTENANCE MODE
2831                            ;
2832
2833  007410  000240            PRG5:   NOP
```

```
2834  007412  104401  017634          TYPE    ,PROG5M         ;INDICATE THAT USER SELECTED
2835                                                          ;PROGRAM #5
2836  007416  104401  021102  PRG5A:  TYPE    ,LINTAD         ;ASK USER FOR THE TRANSMITTER DATA
2837                                                          ;BUFFER ADDRESS OF THE DEVICE
2838                                                          ;HE WISHES TO TEST
2839  007422  104412          RDOCT                           ;ACCEPT THE ANSWER TYPED BY USER
2840                                                          ;AND STORE ON TOP OF STACK
2841                  ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
2842  007424  012602          MOV     (SP)+,R2                ;GET THE ANSWER TYPED
2843  007426  020227  176176  CMP     R2,#176176              ;IS THE NUMBER TOO HIGH?
2844  007432  101152          BHI     REDO4                   ;IF YES - GO TO RETRY SITUATION
2845  007434  020227  175616  CMP     R2,#175616              ;IS THE NUMBER TOO LOW?
2846  007440  103547          BLO     REDO4                   ;IF YES - GO TO RETRY SITUATION
2847  007442  132702  000001  BITB    #BIT0,R2                ;NUMBER IS IN RANGE BUT IS IT
2848                                                          ;ON AN EVEN BOUNDARY?
2849  007446  001144          BNE     REDO4                   ;IF NOT - GO TO RETRY SITUATION
2850                  ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
2851  007450  010203          MOV     R2,R3                   ;GET THE USER RESPONSE
2852  007452  142703  000370  BICB    #370,R3 ;MASK OFF LOWER BYTE EXCEPT FOR
2853                                                          ;LEAST SIGNIFICANT DIGIT
2854  007456  122703  000006  CMPB    #6,R3                   ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
2855                                                          ;USER RESPONSE EQUAL TO A SIX?
2856  007462  001136          BNE     REDO4                   ;BRANCH IF NOT
2857  007464  010267  171512  MOV     R2,$TMP0                ;THE TRANSMITTER ADDRESS TYPED
2858                                                          ;IS OK - STORE FOR FUTURE USE
2859                  ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
2860  007470  016746  170310  MOV     ERRVEC,-(SP)            ;SAVE THE TIMOUT VECTOR
2861  007474  012767  007506  170302  MOV     #2$,ERRVEC      ;SET UP TIMEOUT SERVICE ADDRESS
2862  007502  005712          TST     (R2)                    ;IF PRESENT WE WILL EXECUTE THE
2863                                                          ;NEXT INSTRUCTION - IF NOT WE
2864                                                          ;GO TO 2$:
2865  007504  000412          BR      4$                      ;BRANCH IF PRESENT
2866  007506  004767  005214  2$:     JSR     PC,SUERT2       ;GO SETUP FOR ERROR INFORMATION
2867  007512  012767  007522  171524  MOV     #3$,$ESCAPE     ;POINT OF RETURN AFTER ERROR REPORT
2868  007520  104006          ERROR   +6                      ;XDBR REFERENCE CAUSED TIMEOUT
2869  007522  022626          3$:     CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
2870  007524  012667  170254  MOV     (SP)+,ERRVEC            ;RESTORE TIMEOUT VECTOR
2871  007530  000513          BR      REDO4                   ;GO TO RETRY SITUATION
2872  007532  012667  170246  4$:     MOV     (SP)+,ERRVEC    ;DEVICE REGISTER IS PRESENT!
2873                                                          ;ASK THE USER IF HE DESIRES SOME
2874                                                          ;RANDOM NO. OF MSEC. WAIT TIME
2875                                                          ;BEFORE CHECKING XCSR DONE FLAG
2876  007536  104401  020013  PRG5C:  TYPE,   LENGTH  ;ASK USER FOR THE CHARACTER LENGTH
2877                                                          ;FOR WHICH HIS DEVICE IS SET
2878  007542  104413          RDDEC                           ;ACCEPT THE ANSWER TYPED BY USER
2879                  ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
2880  007544  012600          MOV     (SP)+,R0                ;GET THE ANSWER TYPED
2881  007546  020027  000010  CMP     R0,#8.                  ;IS THE NUMBER TOO HIGH?
2882  007552  101106          BHI     REDO4A                  ;IF YES - GO TO RETRY SITUATION
2883  007554  020027  000005  CMP     R0,#5.                  ;IS THE NUMBER TOO LOW?
2884  007560  103503          BLO     REDO4A                  ;IF YES - GO TO RETRY SITUATION
2885  007562  010067  171446  MOV     R0,$TMP15               ;THE VALUE TYPED IS OK
2886                                                          ;STORE FOR FUTURE USE
2887  007566  104401  021364  TYPE    ,RSTALL                 ;RANDOM NO. OF MSEC. WAIT TIME
2888  007572  104412          RDOCT                           ;ACCEPT THE ANSWER TYPED BY USER
2889                                                          ;AND STORE ON TOP OF STACK.
```

```
                                        MOV     (SP)+,$TMP2     ;GET THE ANSWER TYPED
                                ;WE ARE NOW READY TO INITIALIZE THE BINARY COUNT AND GET
                                ;THE BINARY CHARACTER
                                ;
                                        MOV     #-1,$TMP11      ;SET LEAD IN VARIABLE TO -1
                        PRG5B:  MOV     $TMP11,$TMP12   ;STORE PREVIOUS BINARY CHARACTER
                                        INC     $TMP12          ;FLIP BINARY CHARACTER AGAIN
                                        BIC     #177400,$TMP12  ;MASK TO 8 BITS
                                        MOV     $TMP12,$TMP11   ;STORE BINARY CHARACTER
                                        MOV     $TMP12,$TMP1    ;STORE BINARY CHARACTER
                                                                ;FOR FUTURE USE
                                        MOV     $TMP0,$TMP3     ;GET THE XDBR ADDRESS
                                        SUB     #2,$TMP3        ;FORM THE XCSR ADDRESS
                                1$:     TST     $TMP2           ;DO WE RANDOM STALL?
                                        BEQ     2$              ;BRANCH IF IT WASN'T DESIRED
                                        JSR     PC,STALL        ;GO STALL RANDOM VALUE OF MSEC.
                                2$:     JSR     PC,TIMETX       ;GO WAIT FOR TRANSMITTER DONE
                                                                ;BIT TO SET
                                        TYPE    ,XDB            ;TYPE TRANSMITTER DONE BIT MESSAGE
                                        ERROR   +0              ;XCSR DONE BIT NEVER SET
                                        BIS     #BIT2,@$TMP3    ;SET THE MAINTENANCE BIT IN THE
                                                                ;TRANSMITTER CONTROL STATUS REGISTER
                                        MOV     $TMP1,@$TMP0    ;LOAD TRANSMITTER DATA BUFFER
                                                                ;WITH SELECTED CHARACTER
                                        JSR     PC,TIMERX       ;GO WAIT FOR RECEIVER DONE BIT TO SET
                                        TYPE    ,RDB            ;TYPE RECEIVER DONE BIT MESSAGE
                                        ERROR   +0              ;PCSR DONE BIT NEVER SET
                                        MOV     $TMP3,$TMP4     ;GET THE TRANSMITTER CONTROL
                                                                ;STATUS REGISTER ADDRESS
                                        SUB     #2,$TMP4        ;FORM THE RECEIVER DATA BUFFER
                                                                ;ADDRESS
                                        MOV     @$TMP4,$TMP5    ;STORE THE CHARACTER FROM THE
                                                                ;RECEIVER BUFFER + REST OF CONTENTS
                                        JSR     PC,DATCHK       ;GO TO COMPARE EXPECTED &  RECEIVED
                                                                ;DATA
                                        BR      PRG5B           ;GO BACK TO ISSUE ANOTHER BINARY
                                                                ;CHARACTER
                        REDO4:  TYPE    ,$QUES          ;TYPE A QUESTION MARK(?)
                                        JMP     PRG5A           ;GO BACK TO REITERATE XDBR QUESTION
                        REDO4A: TYPE,   $QUES           ;TYPE '?' INDICATING USER TYPED
                                                                ;SOMETHING WRONG FOR CHARACTER LENGTH
                                        JMP     PRG5C           ;GO BACK TO REISSUE QUESTION

                                ;THIS ROUTINE WILL SET UP:
                                ;       RCSR - RECEVER STATUS REGISTER
                                ;       RBUF - RECEIVER BUFFER REGISTER
                                ;       XCSR - TRANSMITTER STATUS REGISTER
                                ;       XBUF - TRANSMITTER BUFFER REGISTER
                                ;INITIALLY, IN RESPONSE TO USER REPLY TO 1ST DEVICE HE WANTS
                                ;TESTED, AND THEREAFTER, AT THE END OF A PROGRAM TO CYCLE THRU
                                ;ALL DEVICES FOR MULTIPLE DEVICE TESTING (IF REQUIRED)
                        DLADDR: MOV     DLBASE,DLRCSR   ;STORE RECEIVER STATUS REGISTER
                                                                ;OF CURRENT DEVICE
                                        ADD     #2,DLBASE       ;FORM RECEIVER BUFFER REGISTER
                                                                ;OF CURRENT DEVICE
                                        MOV     DLBASE,DLRDBR   ;STORE RECEIVER BUFFER REGISTER
```

```
                                                        ;OF CURRENT DEVICE
010022  062767  000002  171230          ADD     #2,DLBASE       ;FORM TRANSMITTER STATUS REGISTER
                                                        ;OF OF CURRENT DEVICE
010030  016767  171224  171356          MOV     DLBASE,DLXCSR   ;STORE TRANSMITTER STATUS REGISTER
                                                        ;OF CURRENT DEVICE
010036  062767  000002  171214          ADD     #2,DLBASE       ;FOR TRANSMITTER BUFFER REGISTER
                                                        ;OF CURRENT DEVICE
010044  016767  171210  171344          MOV     DLBASE,DLXDBR   ;STORE TRANSMITTER BUFFER REGISTER
                                                        ;OF CURRENT DEVICE
010052  000207                          RTS     PC              ;RETURN

                        .SBTTL  END OF PASS ROUTINE

                        ;;*****************************************************************
                        ;*INCREMENT THE PASS NUMBER ($PASS)
                        ;*TYPE 'END PASS #XXXXX'' (WHERE XXXXX IS A DECIMAL NUMBER)
                        ;*IF THERES A MONITOR GO TO IT
                        ;*IF THERE ISN'T JUMP TO RESTRT

010054                  $EOP:
                        ;THIS NEXT SECTION UP TO THE NEXT LINE OF ASTERISKS WAS
                        ;SUPPLIED BY THE MACRO 'EOPBEG'.  THE MACRO NAME APPEARS IN
                        ;THE SOURCE PROGRAM AS ONE OF THE ARGUMENTS TO THE .$EOP
                        ;SYSMAC UTILITY ROUTINE CALL.
010054  000004                  SCOPE
010056  105767  171210          TSTB    MULTD           ;ARE WE RUNNING MULTIPLE DEVICES?
010062  001501                  BEQ     3$              ;BRANCH IF NOT FOR NORMAL
                                                        ;'END PASS # XX' TYPEOUT
010064  005767  171204          TST     ACTREG          ;ARE ANY DEVICES ACTIVE?
010070  001011                  BNE     1$              ;BRANCH IF YES
010072  104401  020670          TYPE    ,FOULUP         ;INDICATE SOMETHING WRONG!
                                                        ;MULTIPLE DEVICES ARE BEING
                                                        ;RUN SUPPOSEDLY, BUT NONE ARE
                                                        ;SHOWN ACTIVE
010076  000000                  HALT                    ;WAIT FOR A USER RESPONSE
010100  005067  171166          CLR     MULTD           ;CLEAR MULTIPLE DEVICE FLAG
010104  005067  171146          CLR     TABFLG          ;CLEAR TABLE CREATION FLAG
010110  000167  171722          JMP     RESTRT          ;GET READY TO START ALL OVER
                                                        ;AGAIN - ALL DEVICES WERE
                                                        ;DESELECTED SOMEHOW!
010114  062767  000010  171142  1$:     ADD     #10,BASEADD     ;FORM A NEW BASE
                                                        ;ADDRESS FOR START OF NEXT BLOCK
                                                        ;OF REGISTERS FOR NEXT DEVICE
010122  062767  000010  171140          ADD     #10,BASEIV      ;FORM A NEW BASE ADDRESS FOR
                                                        ;START OF NEXT BLOCK OF VECTORS
                                                        ;FOR NEXT DEVICE
010130  000241                  CLC                     ;CLEAR LAST DEVICE INDICATOR
010132  006167  171140          ROL     ROTADD          ;UPDATE NEXT POSSIBLE DEVICE ACTIVE
                                                        ;POINTER
010136  103431                  BCS     2$              ;BRANCH IF THIS WAS THE
                                                        ;LAST DEVICE TO BE TESTED ON
                                                        ;THIS PASS
010140  036767  171132  171126          BIT     ROTADD,ACTREG   ;IS THIS DEVICE TRULY ACTIVE
                                                        ;(AS PER USER)
010146  001762                  BEQ     1$              ;BRANCH IF NOT TO SEE IF NEXT
                                                        ;ONE POSSIBLE IS
```

```
3002  010150  016767  171110  171102          MOV     BASEADD,DLBASE  ;FORM THE RECEIVER STATUS REGISTER
3003                                                                  ;ADDRESS OF NEXT DEVICE
3004  010156  016767  171106  171234          MOV     BASEIV,DLVECT   ;GET NEXT DEVICE RCVR VECTOR
3005  010164  000240                          NOP
3006  010166  004767  177606                  JSR     PC,DLADDR       ;GO FORM DL ADDRESSES FOR NEXT
3007                                                                  ;DEVICE SELECTED
3008  010172  005067  170704                  CLR     $TSTNM          ;INITIALIZE TEST NO. FOR A PROGRAM
3009                                                                  ;PASS OVER THE NEXT DEVICE ACTIVE
3010  010176  005077  171212                  CLR     @DLXCSR         ;CLEAR OUT BOTH CSR'S
3011  010202  005077  171202                  CLR     @DLRCSR
3012  010206  005777  171200                  TST     @DLRDBR         ;FLUSH RCVR 'DONE'' BIT
3013  010212  005777  171174                  TST     @DLRDBR
3014  010216  000167  172612                  JMP     TST1            ;START TESTING THIS DEVICE
3015  010222                          2$:
3016                                  ;IF WE TAKE THIS PATH WE HAVE MADE A CYCLE THRU THE PROGRAM ONCE
3017                                  ;PER DEVICE I.E. - WE HAVE MADE A COMPLETE PASS
3018                                  ;NOW WE NEED TO RESTORE EVERYTHING FOR THE NEXT COMPLETE PASS
3019  010222  012767  000001  171046          MOV     #1,ROTADD       ;SET UP ROTATING POINTER FOR NEXT
3020                                                                  ;MULTIPLE PASS
3021  010230  016767  171026  171026          MOV     KEEPADD,BASEADD ;RESTORE BASE ADDRESS
3022  010236  016767  171024  171024          MOV     KEEPIV,BASEIV   ;RESTORE BASE INTERRUPT VECTOR
3023  010244  016767  171014  171006          MOV     BASEADD,DLBASE  ;RESTORE 1ST DEVICE BASE ADDRESS
3024  010252  016767  171012  171140          MOV     BASEIV,DLVECT   ;RESTORE 1ST DEVICE VECTOR ADDRESS
3025  010260  000240                          NOP
3026  010262  004767  177512                  JSR     PC,DLADDR       ;FORM ADDRESSES FOR 1ST DEVICE
3027  010266                          3$:
3028                                  ;;**********************************************************************
3029  010266  005067  170610                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
3030  010272  005067  170744                  CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
3031  010276  005267  170576                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
3032  010302  042767  100000  170570          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
3033  010310  005327                          DEC     (PC)+           ;;LOOP?
3034  010312  000001                  $EOPCT: .WORD   1
3035  010314  003022                          BGT     $DOAGN          ;;YES
3036  010316  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
3037  010320  000001                  $ENDCT: .WORD   1
3038  010322  010312                          $EOPCT
3039  010324  104401  010371                  TYPE    ,$ENDMG         ;;TYPE 'END PASS #''
3040  010330  016746  170544                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
3041  010334  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
3042  010336  104401  010366                  TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
3043  010342  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
3044  010346  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
3045  010350  000005                          RESET                   ;;CLEAR THE WORLD
3046  010352  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
3047  010354  000240                          NOP                     ;;SAVE ROOM
3048  010356  000240                          NOP                     ;;FOR
3049  010360  000240                          NOP                     ;;ACT11
3050  010362                          $DOAGN:
3051  010362  000137                          JMP     @(PC)+          ;;RETURN
3052  010364  002036                  $RTNAD: .WORD   RESTRT
3053  010366     377     377     000  $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
3054  010371     015  042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS #/
3055  010376  050040  051501  020123
3056  010404  000043
3057                                  .SBTTL  SCOPE HANDLER ROUTINE
```

```
3058
3059                                    ;;*********************************************************************
3060                                    ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
3061                                    ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
3062                                    ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
3063                                    ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
3064                                    ;*SW14=1         LOOP ON TEST
3065                                    ;*SW11=1         INHIBIT ITERATIONS
3066                                    ;*SW09=1         LOOP ON ERROR
3067                                    ;*SW08=1         LOOP ON TEST IN SWR<7:0>
3068                                    ;*CALL
3069                                    ;*      SCOPE           ;;SCOPE=IOT
3070
3071  010406                            $SCOPE:
3072  010406  104407                            CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
3073  010410  032777  040000  170522    1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
3074  010416  001111                            BNE     $OVER           ;;YES IF SW14=1
3075                                    ;#####START OF CODE FOR THE XOR TESTER#####
3076  010420  000416                    $XTSTR: BR      6$              ;;IF RUNNING ON THE ''XOR'' TESTER CHANGE
3077                                                                    ;;THIS INSTRUCTION TO A ''NOP'' (NOP=240)
3078  010422  013746  000004                    MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
3079  010426  012737  010446  000004            MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
3080  010434  005737  177060                    TST     @#177060        ;;TIME OUT ON XOR?
3081  010440  012637  000004                    MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
3082  010444  000463                            BR      $SVLAD          ;;GO TO THE NEXT TEST
3083  010446  022626                    5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
3084  010450  012637  000004                    MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
3085  010454  000423                            BR      7$              ;;LOOP ON THE PRESENT TEST
3086  010456                            6$:;#####END OF CODE FOR THE XOR TESTER#####
3087  010456  032777  000400  170454            BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
3088  010464  001404                            BEQ     2$              ;;BR IF NO
3089  010466  127767  170446  170406            CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
3090  010474  001462                            BEQ     $OVER           ;;BR IF YES
3091  010476  105767  170401          2$:       TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
3092  010502  001421                            BEQ     3$              ;;BR IF NO
3093  010504  126767  170405  170371            CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
3094  010512  101015                            BHI     3$              ;;BR IF NO
3095  010514  032777  001000  170416            BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
3096  010522  001404                            BEQ     4$              ;;BR IF NO
3097  010524  016767  170360  170354  7$:       MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
3098  010532  000443                            BR      $OVER
3099  010534  105067  170343          4$:       CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
3100  010540  005067  170476                    CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
3101  010544  000415                            BR      1$              ;;ESCAPE TO THE NEXT TEST
3102  010546  032777  004000  170364  3$:       BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
3103  010554  001011                            BNE     1$              ;;BR IF YES
3104  010556  005767  170316                    TST     $PASS           ;;IF FIRST PASS OF PROGRAM
3105  010562  001406                            BEQ     1$              ;;      INHIBIT ITERATIONS
3106  010564  005267  170314                    INC     $ICNT           ;;INCREMENT ITERATION COUNT
3107  010570  026767  170446  170306            CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
3108  010576  002021                            BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
3109  010600  012767  000001  170276  1$:       MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
3110  010606  016767  000044  170426            MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
3111  010614  105267  170262          $SVLAD:   INCB    $TSTNM          ;;COUNT TEST NUMBERS
3112  010620  011667  170262                    MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
3113  010624  011667  170260                    MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
```

```
3114  010630  005067  170410                  CLR     $ESCAPE          ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
3115  010634  112767  000001  170253          MOVB    #1,$ERMAX        ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
3116  010642  016777  170234  170272  $OVER:  MOV     $TSTNM,@DISPLAY  ;;DISPLAY TEST NUMBER
3117  010650  016716  170232                  MOV     $LPADR,(SP)      ;;FUDGE RETURN ADDRESS
3118  010654  000002                          RTI                      ;;FIXES PS
3119  010656  000100                  $MXCNT: 100                      ;;MAX. NUMBER OF ITERATIONS
3120
3121                                   .SBTTL  ERROR HANDLER ROUTINE
3122
3123                                   ;;**********************************************************************
3124                                   ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
3125                                   ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
3126                                   ;*AND GO TO $ERRTYP ON ERROR
3127                                   ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
3128                                   ;*SW15=1         HALT ON ERROR
3129                                   ;*SW13=1         INHIBIT ERROR TYPEOUTS
3130                                   ;*SW10=1         BELL ON ERROR
3131                                   ;*SW09=1         LOOP ON ERROR
3132                                   ;*CALL
3133                                   ;*       ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
3134
3135  010660                           $ERROR:
3136  010660  104407                          CKSWR                    ;;TEST FOR CHANGE IN SOFT-SWR
3137  010662  105267  170215          7$:     INCB    $ERFLG           ;;SET THE ERROR FLAG
3138  010666  001775                          BEQ     7$               ;;DON'T LET THE FLAG GO TO ZERO
3139  010670  016777  170206  170244          MOV     $TSTNM,@DISPLAY  ;;DISPLAY TEST NUMBER AND ERROR FLAG
3140  010676  032777  002000  170234          BIT     #BIT10,@SWR      ;;BELL ON ERROR?
3141  010704  001402                          BEQ     1$               ;;NO - SKIP
3142  010706  104401  001246                  TYPE    ,$BELL           ;;RING BELL
3143  010712  005267  170174          1$:     INC     $ERTTL           ;;COUNT THE NUMBER OF ERRORS
3144  010716  011667  170174                  MOV     (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
3145  010722  162767  000002  170166          SUB     #2,$ERRPC
3146  010730  117767  170162  170156          MOVB    @$ERRPC,$ITEMB   ;;STRIP AND SAVE THE ERROR ITEM CODE
3147  010736  032777  020000  170174          BIT     #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
3148  010744  001004                          BNE     20$              ;;SKIP TYPEOUTS
3149  010746  004767  000046                  JSR     PC,$ERRTYP       ;;GO TO USER ERROR ROUTINE
3150  010752  104401  001253                  TYPE    ,$CRLF
3151  010756                           20$:
3152  010756  005777  170156          2$:     TST     @SWR             ;;HALT ON ERROR
3153  010762  100002                          BPL     3$               ;;SKIP IF CONTINUE
3154  010764  000000                          HALT                     ;;HALT ON ERROR!
3155  010766  104407                          CKSWR                    ;;TEST FOR CHANGE IN SOFT-SWR
3156  010770  032777  001000  170142  3$:     BIT     #BIT09,@SWR      ;;LOOP ON ERROR SWITCH SET?
3157  010776  001402                          BEQ     4$               ;;BR IF NO
3158  011000  016716  170104                  MOV     $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
3159  011004  005767  170234          4$:     TST     $ESCAPE          ;;CHECK FOR AN ESCAPE ADDRESS
3160  011010  001402                          BEQ     5$               ;;BR IF NONE
3161  011012  016716  170226                  MOV     $ESCAPE,(SP)     ;;FUDGE RETURN ADDRESS FOR ESCAPE
3162  011016                           5$:
3163  011016  000002                          RTI                      ;;RETURN
3164
3165                                   .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
3166
3167                                   ;;**********************************************************************
3168                                   ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
3169                                   ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
```

```
3170                                    ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
3171
3172  011020                            $ERRTYP:
3173  011020  104401  001253                  TYPE    ,$CRLF          ;;''CARRIAGE RETURN'' & 'LINE FEED''
3174  011024  010046                          MOV     R0,-(SP)        ;;SAVE R0
3175  011026  005000                          CLR     R0              ;;PICKUP THE ITEM INDEX
3176  011030  153700  001114                  BISB    @#$ITEMB,R0
3177  011034  001004                          BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
3178                                                                   ;;TYPE THE PC OF THE ERROR
3179  011036  016746  170054                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
3180                                                                   ;;ERROR ADDRESS
3181  011042  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3182  011044  000426                          BR      6$              ;;GET OUT
3183  011046  005300                  1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
3184  011050  006300                          ASL     R0              ;;      WORK FOR THE ERROR TABLE
3185  011052  006300                          ASL     R0
3186  011054  006300                          ASL     R0
3187  011056  062700  001310                  ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
3188  011062  012067  000004                  MOV     (R0)+,2$        ;;PICKUP ''ERROR MESSAGE'' POINTER
3189  011066  001404                          BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
3190  011070  104401                          TYPE                    ;;TYPE THE ''ERROR MESSAGE''
3191  011072  000000                  2$:     .WORD   0               ;;''ERROR MESSAGE'' POINTER GOES HERE
3192  011074  104401  001253                  TYPE    ,$CRLF          ;;''CARRIAGE RETURN'' & 'LINE FEED''
3193  011100  012067  000004          3$:     MOV     (R0)+,4$        ;;PICKUP ''DATA HEADER'' POINTER
3194  011104  001404                          BEQ     5$              ;;SKIP TYPEOUT IF 0
3195  011106  104401                          TYPE                    ;;TYPE THE ''DATA HEADER''
3196  011110  000000                  4$:     .WORD   0               ;;''DATA HEADER'' POINTER GOES HERE
3197  011112  104401  001253                  TYPE    ,$CRLF          ;;''CARRIAGE RETURN'' & 'LINE FEED''
3198  011116  011000                  5$:     MOV     (R0),R0         ;;PICKUP ''DATA TABLE'' POINTER
3199  011120  001004                          BNE     7$              ;;GO TYPE THE DATA
3200  011122  012600                  6$:     MOV     (SP)+,R0        ;;RESTORE R0
3201  011124  104401  001253                  TYPE    ,$CRLF          ;;''CARRIAGE RETURN'' & 'LINE FEED''
3202  011130  000207                          RTS     PC              ;;RETURN
3203  011132                            7$:
3204  011132  013046                          MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
3205  011134  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3206  011136  005710                          TST     (R0)            ;;IS THERE ANOTHER NUMBER?
3207  011140  001770                          BEQ     6$              ;;BR IF NO
3208  011142  104401  011150                  TYPE    ,8$             ;;TYPE TWO(2) SPACES
3209  011146  000771                          BR      7$              ;;LOOP
3210  011150  020040     000          8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
3211           011154                         .EVEN
3212
3213                                    .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
3214
3215                                    ;;**********************************************************************
3216                                    ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
3217                                    ;*OCTAL (ASCII) NUMBER AND TYPE IT.
3218                                    ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
3219                                    ;*CALL:
3220                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
3221                                    ;*      TYPOS                   ;;CALL FOR TYPEOUT
3222                                    ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3223                                    ;*      .BYTE   M               ;;M=1 OR 0
3224                                    ;*                                      ;;1=TYPE LEADING ZEROS
3225                                    ;*                                      ;;0=SUPPRESS LEADING ZEROS
```

```
                                        ;*
                                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                        ;*$TYPOS OR $TYPOC
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT

011154  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
011160  116667  000001  000211          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
011166  112667  000207                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
011172  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
011176  000406                          BR      $TYPON
011200  112767  000001  000171  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
011206  112767  000006  000165          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
011214  112767  000005  000154  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
011222  010346                          MOV     R3,-(SP)        ;;SAVE R3
011224  010446                          MOV     R4,-(SP)        ;;SAVE R4
011226  010546                          MOV     R5,-(SP)        ;;SAVE R5
011230  116704  000145                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
011234  005404                          NEG     R4
011236  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
011242  110467  000132                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
011246  116704  000125                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
011252  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
011256  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
011260  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
011262  000404                          BR      3$              ;;GO DO MSB
011264  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
011266  006105                          ROL     R5
011270  006105                          ROL     R5
011272  010503                          MOV     R5,R3
011274  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
011276  105367  000076                  DECB    $OMODE          ;;TYPE THIS DIGIT?
011302  100016                          BPL     7$              ;;BR IF NO
011304  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
011310  001002                          BNE     4$              ;;TEST FOR 0
011312  005704                          TST     R4              ;;SUPPRESS THIS 0?
011314  001403                          BEQ     5$              ;;BR IF YES
011316  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
011320  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
011324  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
011330  110367  000040                  MOVB    R3,8$           ;;SAVE FOR TYPING
011334  104401  011374                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
011340  105367  000032          7$:     DECB    $OCNT           ;;COUNT BY 1
011344  003347                          BGT     2$              ;;BR IF MORE TO DO
011346  002402                          BLT     6$              ;;BR IF DONE
011350  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
011352  000744                          BR      2$              ;;GO DO THE LAST DIGIT
011354  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
011356  012604                          MOV     (SP)+,R4        ;;RESTORE R4
011360  012603                          MOV     (SP)+,R3        ;;RESTORE R3
```

```
3282  011362  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
3283  011370  012616                          MOV     (SP)+,(SP)
3284  011372  000002                          RTI                     ;;RETURN
3285  011374     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
3286  011375     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
3287  011376     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
3288  011377     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
3289  011400  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
3290
3291                                  .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
3292
3293                                  ;;**************************************************************************
3294                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
3295                                  ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
3296                                  ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
3297                                  ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
3298                                  ;*REPLACED WITH SPACES.
3299                                  ;*CALL:
3300                                  ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
3301                                  ;*      TYPDS                   ;;GO TO THE ROUTINE
3302
3303  011402                          $TYPDS:
3304  011402  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3305  011404  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3306  011406  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3307  011410  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3308  011412  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
3309  011414  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
3310  011420  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
3311  011424  100004                          BPL     1$              ;;BR IF INPUT IS POS.
3312  011426  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
3313  011430  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
3314  011436  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
3315  011440  012703  011616                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
3316  011444  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
3317  011450  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
3318  011452  016001  011606                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
3319  011456  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
3320  011460  002402                          BLT     4$              ;;BR IF DONE
3321  011462  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
3322  011464  000774                          BR      3$
3323  011466  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
3324  011470  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
3325  011472  001002                          BNE     5$              ;;FALL THROUGH IF 0
3326  011474  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
3327  011476  100407                          BMI     7$              ;;BR IF YES
3328  011500  106316                  5$:     ASLB    (SP)            ;;MSD?
3329  011502  103003                          BCC     6$              ;;BR IF NO
3330  011504  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
3331  011512  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
3332  011516  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
3333  011522  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
3334  011524  005720                          TST     (R0)+           ;;JUST INCREMENTING
3335  011526  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
3336  011532  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
3337  011534  003002                          BGT     8$              ;;GO TO EXIT
```

```
3338  011536  010502                        MOV     R5,R2           ;;GET THE LSD
3339  011540  000764                        BR      6$              ;;GO CHANGE TO ASCII
3340  011542  105726                8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
3341  011544  100003                        BPL     9$              ;;BR IF NO
3342  011546  116663  177777  177776        MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
3343  011554  105013                9$:     CLRB    (R3)            ;;SET THE TERMINATOR
3344  011556  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
3345  011560  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
3346  011562  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
3347  011564  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
3348  011566  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
3349  011570  104401  011616                TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
3350  011574  016666  000002  000004        MOV     2(SP),4(SP)     ;;ADJUST THE STACK
3351  011602  012616                        MOV     (SP)+,(SP)
3352  011604  000002                        RTI                     ;;RETURN TO USER
3353  011606  023420                $DTBL:  10000.
3354  011610  001750                        1000.
3355  011612  000144                        100.
3356  011614  000012                        10.
3357  011616  000004                $DBLK:  .BLKW   4
3358
3359                                 .SBTTL  TTY INPUT ROUTINE
3360
3361                                 ;;**************************************************************************
3362                                 .ENABL  LSB
3363  011626  000000                $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
3364  011630  000000                $TKQIN: .WORD   0               ;;INPUT POINTER
3365  011632  000000                $TKQOUT: .WORD  0               ;;OUTPUT POINTER
3366  011634  000010                $TKQSRT: .BLKB  8.              ;;TTY KEYBOARD QUEUE
3367          011644                $TKQEND=.
3368
3369                                 ;*TK INITIALIZE ROUTINE
3370                                 ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
3371                                 ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
3372                                 ;
3373                                 ;*CALL:
3374                                 ;*      JSR     PC,$TKINT
3375                                 ;*      RETURN
3376                                 ;
3377  011644  005067  177756        $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
3378  01165C  012767  011634  177752        MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
3379  011656  016767  177746  177746        MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
3380  011664  012737  011714  000060        MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
3381  011672  012737  000200  000062        MOV     #200,@#TKVEC+2  ;;'BR' LEVEL 4
3382  011700  005777  167242                TST     @$TKB           ;;CLEAR DONE FLAG
3383  011704  012777  000100  167232        MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
3384  011712  000207                        RTS     PC              ;;RETURN TO CALLER
3385
3386                                 ;*TK SERVICE ROUTINE
3387                                 ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
3388                                 ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
3389                                 ;*IT IN THE QUEUE.
3390                                 ;*IF THE CHARACTER IS A 'CONTROL-C' (^C) $TKINT IS CALLED AND
3391                                 ;*UPON RETURN EXIT IS MADE TO THE 'CONTROL-C' RESTART ADDRESS (200)
3392                                 ;
3393  011714  117746  167226        $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
```

```
3394  011720  042716  177600                  BIC     #^C177,(SP)     ;;STRIP THE JUNK
3395  011724  021627  000003                  CMP     (SP),#3         ;;IS IT A CONTROL C?
3396  011730  001007                          BNE     1$              ;;BRANCH IF NO
3397  011732  104401  013030                  TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
3398  011736  004767  177702                  JSR     PC,$TKINT       ;;INIT THE KEYBOARD
3399  011742  005726                          TST     (SP)+           ;;CLEAN UP STACK
3400  011744  000167  166230                  JMP     200             ;;CONTROL C RESTART
3401  011750  021627  000007          1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
3402  011754  001004                          BNE     2$              ;;BRANCH IF NO
3403  011756  022767  000176  167154          CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
3404  011764  001500                          BEQ     6$              ;;GO TO SWR CHANGE
3405
3406  011766                          2$:
3407  011766  022767  000010  177632          CMP     #8.,$TKCNT      ;;IS THE QUEUE FULL?
3408  011774  001004                          BNE     3$              ;;BRANCH IF NO
3409  011776  104401  001246                  TYPE    ,$BELL          ;;RING THE TTY BELL
3410  012002  005726                          TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
3411  012004  000451                          BR      5$              ;;EXIT
3412  012006  021627  000023          3$:     CMP     (SP),#23        ;;IS IT A CONTROL-S?
3413  012012  001021                          BNE     32$             ;;BRANCH IF NO
3414  012014  005077  167124                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
3415  012020  005726                          TST     (SP)+           ;;CLEAN CHAR OFF STACK
3416  012022  105777  167116          31$:    TSTB    @$TKS           ;;WAIT FOR A CHAR
3417  012026  100375                          BPL     31$             ;;LOOP UNTIL ITS THERE
3418  012030  117746  167112                  MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
3419  012034  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
3420  012040  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
3421  012044  001366                          BNE     31$             ;;BRANCH IF NO
3422  012046  012777  000100  167070          MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
3423  012054  000002                          RTI                     ;;RETURN
3424  012056  005267  177544          32$:    INC     $TKCNT          ;;COUNT THIS CHARACTER
3425  012062  021627  000140                  CMP     (SP),#140       ;;IS IT UPPER CASE?
3426  012066  002405                          BLT     4$              ;;BRANCH IF YES
3427  012070  021627  000175                  CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
3428  012074  003002                          BGT     4$              ;;BRANCH IF YES
3429  012076  042716  000040                  BIC     #40,(SP)        ;;MAKE IT UPPER CASE
3430  012102  112677  177522          4$:     MOVB    (SP)+,@$TKQIN   ;;AND PUT IT IN QUEUE
3431  012106  005267  177516                  INC     $TKQIN          ;;UPDATE THE POINTER
3432  012112  026727  177512  011644          CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
3433  012120  001003                          BNE     5$              ;;BRANCH IF NO
3434  012122  012767  011634  177500          MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
3435  012130  000002                  5$:     RTI                     ;;RETURN
3436
3437                                  ;;**********************************************************************
3438                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
3439                                  ;*ROUTINE IS ENTERED  FROM THE TRAP HANDLER, AND WILL
3440                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
3441                                  ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
3442  012132  022767  000176  167000  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
3443  012140  001124                          BNE     15$             ;;EXIT IF NOT
3444  012142  105777  166776                  TSTB    @$TKS           ;;IS A CHAR WAITING?
3445  012146  100121                          BPL     15$             ;;IF NOT, EXIT
3446  012150  117746  166772                  MOVB    @$TKB,-(SP)     ;;YES
3447  012154  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
3448  012160  021627  000007                  CMP     (SP),#7         ;;IS IT A CONTROL-G?
3449  012164  001300                          BNE     2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
```

```
                                                        ;;AND EXIT
                                                ;;**********************************************************************
                                                ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
                                                ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
                                                ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
012166  126727  166742  000001  6$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
012174  001674                          BEQ     2$              ;;BRANCH IF YES
012176  005726                          TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
012200  004767  177440                  JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
012204  005077  166734                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
012210  112767  000001  166717          MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR

012216  104401  013042          $GTSWR: TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
012222  104401  013047                  TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
012226  016746  165744                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
012232  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
012234  104401  013060                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
012240  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
012242  005046                          CLR     -(SP)           ;;THE NEW SWR
012244  105777  166674          7$:     TSTB    @$TKS           ;;CHAR THERE?
012250  100375                          BPL     7$              ;;IF NOT TRY AGAIN

012252  117746  166670                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
012256  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII

012262  021627  000003                  CMP     (SP),#3         ;;IS IT A CONTROL-C?
012266  001015                          BNE     9$              ;;BRANCH IF NOT
012270  104401  013030                  TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
012274  062706  000006                  ADD     #6,SP           ;;CLEAN UP STACK
012300  126727  166631  000001          CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
012306  001003                          BNE     8$              ;;BRANCH IF NO
012310  012777  000100  166626          MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
012316  000167  165656          8$:     JMP     200             ;;CONTROL-C RESTART

012322  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
012326  001005                          BNE     10$             ;;BRANCH IF NOT
012330  104401  013035                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
012334  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
012340  000737                          BR      19$             ;;LET'S TRY IT AGAIN

012342  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
012346  001022                          BNE     16$             ;;BRANCH IF NO
012350  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
012354  001403                          BEQ     11$             ;;BRANCH IF YES
012356  016677  000002  166554          MOV     2(SP),@SWR      ;;SAVE NEW SWR
012364  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
012370  104401  001253          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
012374  126727  166535  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
012402  001003                          BNE     15$             ;;BRANCH IF NOT
012404  012777  000100  166532          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
012412  000002                  15$:    RTI                     ;;RETURN
012414  004767  000762          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
012420  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
```

```
3506  012424  002420                        BLT     18$             ;;BRANCH IF YES
3507  012426  021627  000067                CMP     (SP),#67        ;;CHAR > 7?
3508  012432  003015                        BGT     18$             ;;BRANCH IF YES
3509  012434  042726  000060                BIC     #60,(SP)+       ;;STRIP-OFF ASCII
3510  012440  005766  000002                TST     2(SP)           ;;IS THIS THE FIRST CHAR
3511  012444  001403                        BEQ     17$             ;;BRANCH IF YES
3512  012446  006316                        ASL     (SP)            ;;NO, SHIFT PRESENT
3513  012450  006316                        ASL     (SP)            ;;   CHAR OVER TO MAKE
3514  012452  006316                        ASL     (SP)            ;;   ROOM FOR NEW ONE.
3515  012454  005266  000002        17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
3516  012460  056616  177776                BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
3517  012464  000667                        BR      7$              ;;GET THE NEXT ONE
3518  012466  104401  001252        18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
3519  012472  000720                        BR      20$             ;;SIMULATE CONTROL-U
3520                                .DSABL  LSB
3521
3522
3523                                ;;**********************************************************************
3524                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
3525                                ;*CALL:
3526                                ;*      RDCHR                   ;;GET A CHARACTER FROM THE QUEUE
3527                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
3528                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
3529                                ;
3530
3531  012474  011646                $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
3532  012476  016666  000004  000002        MOV     4(SP),2(SP)     ;;THE PS
3533  012504  005066  000004                CLR     4(SP)           ;;GET READY FOR A CHARACTER
3534  012510  005046                        CLR     -(SP)           ;;PUT NEW PS ON STACK
3535  012512  012746  012520                MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
3536  012516  000002                        RTI                     ;;POP NEW PC AND PS
3537  012520                        64$:
3538  012520  005767  177102        1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
3539  012524  001775                        BEQ     1$
3540  012526  005367  177074                DEC     $TKCNT          ;;DECREMENT THE COUNTER
3541  012532  117766  177074  000004        MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
3542  012540  005267  177066                INC     $TKQOUT         ;;UPDATE THE POINTER
3543  012544  026727  177062  011644        CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
3544  012552  001003                        BNE     2$              ;;BRANCH IF NO
3545  012554  012767  011634  177050        MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
3546  012562  000002                2$:     RTI                     ;;RETURN
3547                                ;;**********************************************************************
3548                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
3549                                ;*CALL:
3550                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
3551                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3552                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
3553
3554  012564  010346                $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
3555  012566  005046                        CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
3556  012570  012703  013020        1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
3557  012574  022703  013030        2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
3558  012600  101456                        BLOS    4$              ;;BR IF YES
3559  012602  104410                        RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
3560  012604  112613                        MOVB    (SP)+,(R3)      ;;GET CHARACTER
3561  012606  122713  000177        10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
```

```
3562  012612  001022                        BNE     5$              ;;BR IF NO
3563  012614  005716                        TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
3564  012616  001007                        BNE     6$              ;;BR IF NO
3565  012620  112767  000134  000170        MOVB    #'\,9$          ;;TYPE A BACK SLASH
3566  012626  104401  013016                TYPE    ,9$
3567  012632  012716  177777                MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
3568  012636  005303                6$:     DEC     R3              ;;BACKUP BY ONE
3569  012640  020327  013020                CMP     R3,#$TTYIN      ;;STACK EMPTY?
3570  012644  103434                        BLO     4$              ;;BR IF YES
3571  012646  111367  000144                MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
3572  012652  104401  013016                TYPE    ,9$             ;;GO TYPE
3573  012656  000746                        BR      2$              ;;GO READ ANOTHER CHAR.
3574  012660  005716                5$:     TST     (SP)            ;;RUBOUT KEY SET?
3575  012662  001406                        BEQ     7$              ;;BR IF NO
3576  012664  112767  000134  000124        MOVB    #'\,9$          ;;TYPE A BACK SLASH
3577  012672  104401  013016                TYPE    ,9$
3578  012676  005016                        CLR     (SP)            ;;CLEAR THE RUBOUT KEY
3579  012700  122713  000025        7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
3580  012704  001003                        BNE     8$              ;;BR IF NO
3581  012706  104401  013035                TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U''
3582  012712  000726                        BR      1$              ;;GO START OVER
3583  012714  122713  000022        8$:     CMPB    #22,(R3)        ;;IS CHARACTER A ''^R''?
3584  012720  001011                        BNE     3$              ;;BRANCH IF NO
3585  012722  105013                        CLRB    (R3)            ;;CLEAR THE CHARACTER
3586  012724  104401  001253                TYPE    ,$CRLF          ;;TYPE A ''CR'' & 'LF''
3587  012730  104401  013020                TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
3588  012734  000717                        BR      2$              ;;GO PICKUP ANOTHER CHACTER
3589  012736  104401  001252        4$:     TYPE    ,$QUES          ;;TYPE A '?'
3590  012742  000712                        BR      1$              ;;CLEAR THE BUFFER AND LOOP
3591  012744  111367  000046        3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
3592  012750  104401  013016                TYPE    ,9$
3593  012754  122723  000015                CMPB    #15,(R3)+       ;;CHECK FOR RETURN
3594  012760  001305                        BNE     2$              ;;LOOP IF NOT RETURN
3595  012762  105063  177777                CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
3596  012766  104401  001254                TYPE    ,$LF            ;;TYPE A LINE FEED
3597  012772  005726                        TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
3598  012774  012603                        MOV     (SP)+,R3        ;;RESTORE R3
3599  012776  011646                        MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
3600  013000  016666  000004  000002        MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
3601  013006  012766  013020  000004        MOV     #$TTYIN,4(SP)
3602  013014  000002                        RTI                     ;;RETURN
3603  013016     000                9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
3604  013017     000                        .BYTE   0               ;;TERMINATOR
3605  013020  000010                $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
3606  013030  041536  005015     000 $CNTLC: .ASCIZ  /^C/<15><12>    ;;CONTROL ''C''
3607  013035     136  006525  000012 $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL 'U''
3608  013042  043536  005015     000 $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL ''G''
3609  013047     015  051412  051127 $MSWR:  .ASCIZ  <15><12>/SWR = /
3610  013054  036440  000040
3611  013060  020040  042516  020127 $MNEW:  .ASCIZ  /  NEW = /
3612  013066  020075     000
3613          013072                .EVEN
3614
3615                                .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
3616
3617                                ;;**************************************************************
```

```
                                        ;*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

013072  011646                 $RDOCT: MOV     (SP),-(SP)              ;;PROVIDE SPACE FOR THE
013074  016666  000004  000002          MOV     4(SP),2(SP)             ;;INPUT NUMBER
013102  010046                          MOV     R0,-(SP)                ;;PUSH R0 ON STACK
013104  010146                          MOV     R1,-(SP)                ;;PUSH R1 ON STACK
013106  010246                          MOV     R2,-(SP)                ;;PUSH R2 ON STACK
013110  104411                  1$:     RDLIN                           ;;READ AN ASCIZ LINE
013112  012600                          MOV     (SP)+,R0                ;;GET ADDRESS OF 1ST CHARACTER
013114  010067  000100                  MOV     R0,5$                   ;;AND SAVE IT
013120  005001                          CLR     R1                      ;;CLEAR DATA WORD
013122  005002                          CLR     R2
013124  112046                  2$:     MOVB    (R0)+,-(SP)             ;;PICKUP THIS CHARACTER
013126  001420                          BEQ     3$                      ;;IF ZERO GET OUT
013130  122716  000060                  CMPB    #'0,(SP)                ;;MAKE SURE THIS CHARACTER
013134  003026                          BGT     4$                      ;;IS AN OCTAL DIGIT
013136  122716  000067                  CMPB    #'7,(SP)
013142  002423                          BLT     4$
013144  006301                          ASL     R1                      ;;*2
013146  006102                          ROL     R2
013150  006301                          ASL     R1                      ;;*4
013152  006102                          ROL     R2
013154  006301                          ASL     R1                      ;;*8
013156  006102                          ROL     R2
013160  042716  177770                  BIC     #^C7,(SP)               ;;STRIP THE ASCII JUNK
013164  062601                          ADD     (SP)+,R1                ;;ADD IN THIS DIGIT
013166  000756                          BR      2$                      ;;LOOP
013170  005726                  3$:     TST     (SP)+                   ;;CLEAN TERMINATOR FROM STACK
013172  010166  000012                  MOV     R1,12(SP)               ;;SAVE THE RESULT
013176  010267  000026                  MOV     R2,$HIOCT
013202  012602                          MOV     (SP)+,R2                ;;POP STACK INTO R2
013204  012601                          MOV     (SP)+,R1                ;;POP STACK INTO R1
013206  012600                          MOV     (SP)+,R0                ;;POP STACK INTO R0
013210  000002                          RTI                             ;;RETURN
013212  005726                  4$:     TST     (SP)+                   ;;CLEAN PARTIAL FROM STACK
013214  105010                          CLRB    (R0)                    ;;SET A TERMINATOR
013216  104401                          TYPE                            ;;TYPE UP THRU THE BAD CHAR.
013220  000000                  5$:     .WORD   0
013222  104401  001252                  TYPE    ,$QUES                  ;;"?" "CR" & "LF"
013226  000730                          BR      1$                      ;;TRY AGAIN
013230  000000                  $HIOCT: .WORD   0                       ;;HIGH ORDER BITS GO HERE

                                .SBTTL  TYPE ROUTINE

                                ;;**************************************************************
                                ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
```

```
                                        ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                        ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                        ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                        ;*
                                        ;*CALL:
                                        ;*1) USING A TRAP INSTRUCTION
                                        ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                                        ;*OR
                                        ;*      TYPE
                                        ;*      MESADR
                                        ;*
013232  105767  165721          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
013236  100002                         BPL     1$              ;;BR IF YES
013240  000000                         HALT                    ;;HALT HERE IF NO TERMINAL
013242  000407                         BR      3$              ;;LEAVE
013244  010046                  1$:    MOV     R0,-(SP)        ;;SAVE R0
013246  017600  000002                 MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
013252  112046                  2$:    MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
013254  001005                         BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
013256  005726                         TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
013260  012600                  60$:   MOV     (SP)+,R0        ;;RESTORE R0
013262  062716  000002           3$:    ADD     #2,(SP)         ;;ADJUST RETURN PC
013266  000002                         RTI                     ;;RETURN
013270  122716  000011           4$:    CMPB    #HT,(SP)        ;;BRANCH IF <HT>
013274  001430                         BEQ     8$
013276  122716  000200                 CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
013302  001006                         BNE     5$
013304  005726                         TST     (SP)+           ;;POP  <CR><LF> EQUIV
013306  104401                         TYPE                    ;;TYPE A CR AND LF
013310  001253                         $CRLF
013312  105067  000130                 CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
013316  000755                         BR      2$              ;;GET NEXT CHARACTER
013320  004767  000056           5$:    JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
013324  126726  165626           6$:    CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
013330  001350                         BNE     2$              ;;IF NO GO GET NEXT CHAR.
013332  016746  165616                 MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
013336  105366  000001           7$:    DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
013342  002770                         BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
013344  004767  000032                 JSR     PC,$TYPEC       ;;GO TYPE A NULL
013350  105367  000072                 DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
013354  000770                         BR      7$              ;;LOOP

                                        ;HORIZONTAL TAB PROCESSOR

013356  112716  000040           8$:    MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
013362  004767  000014           9$:    JSR     PC,$TYPEC       ;;TYPE A SPACE
013366  132767  000007  000052         BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
013374  001372                         BNE     9$              ;;TAB STOP
013376  005726                         TST     (SP)+           ;;POP SPACE OFF STACK
013400  000724                         BR      2$              ;;GET NEXT CHARACTER
013402  105777  165542          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
013406  100375                         BPL     $TYPEC
013410  116677  000002  165534         MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
013416  122766  000015  000002         CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
```

```
3730  013424  001003                          BNE     1$              ;;BRANCH IF NO
3731  013426  105067  000014                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
3732  013432  000406                          BR      $TYPEX          ;;EXIT
3733  013434  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
3734  013442  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
3735  013444  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
3736  013446  000000                $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
3737  013450  000207                $TYPEX: RTS     PC
3738
3739
3740                                .SBTTL  READ A DECIMAL NUMBER FROM THE TTY
3741
3742                                ;;**********************************************************************
3743                                ;*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
3744                                ;*CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
3745                                ;*ARE READ A ''?'' FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
3746                                ;*THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
3747                                ;*USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
3748                                ;*POSITIVE 32767 TO NEGATIVE 32768.
3749                                ;*CALL:
3750                                ;*      RDDEC                   ;;READ A DECIMAL NUMBER
3751                                ;*      RETURN HERE             ;;NUMBER IS ON TOP OF THE STACK
3752                                ;
3753
3754  013452  011646                $RDDEC: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR
3755  013454  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE INPUT NUMBER
3756  013462  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3757  013464  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3758  013466  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3759  013470  104411                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
3760  013472  012600                          MOV     (SP)+,R0        ;;ADDRESS OF 1ST CHAR.
3761  013474  010067  000120                  MOV     R0,6$           ;;SAVE INCASE OF BAD INPUT
3762  013500  005046                          CLR     -(SP)           ;;CLEAR DATA WORD
3763  013502  005002                          CLR     R2              ;;SIGN SET POSITIVE
3764  013504  122710  000055                  CMPB    #'-,(R0)        ;;SEE IF A MINUS SIGN WAS TYPED
3765  013510  001001                          BNE     2$              ;;BR IF NO MINUS SIGN
3766  013512  112002                          MOVB    (R0)+,R2        ;;SAVE FOR LATER USE
3767  013514  112001                  2$:     MOVB    (R0)+,R1        ;;PICKUP THIS CHARACTER
3768  013516  001424                          BEQ     3$              ;;GET OUT IF ZERO
3769  013520  122701  000060                  CMPB    #'0,R1          ;;MAKE SURE THIS CHARACTER
3770  013524  003032                          BGT     5$              ;;IS A DIGIT BETWEEN 0 & 9
3771  013526  122701  000071                  CMPB    #'9,R1
3772  013532  002427                          BLT     5$
3773  013534  032716  170000                  BIT     #^C7777,(SP)    ;;DON'T LET NUMBER GET TO BIG
3774  013540  001024                          BNE     5$              ;;BR IF NUMBER WOULD OVERFLOW
3775  013542  006316                          ASL     (SP)            ;;*2
3776  013544  011646                          MOV     (SP),-(SP)      ;;SAVE FOR LATER
3777  013546  006316                          ASL     (SP)            ;;*4
3778  013550  006316                          ASL     (SP)            ;;*8.
3779  013552  062616                          ADD     (SP)+,(SP)      ;;*10.
3780  013554  102416                          BVS     5$              ;;OVERFLOW ISN'T ALLOWED
3781  013556  162701  000060                  SUB     #'0,R1          ;;STRIP AWAY THE ASCII JUNK
3782  013562  060116                          ADD     R1,(SP)         ;;ADD IN THIS DIGIT
3783  013564  102412                          BVS     5$              ;;OVERFLOW ISN'T ALLOWED
3784  013566  000752                          BR      2$              ;;LOOP
3785  013570  005702                  3$:     TST     R2              ;;CHECK IF NUMBER IS NEG
```

```
3786  013572  001401                        BEQ     4$              ;;BR IF NO
3787  013574  005416                        NEG     (SP)            ;;YES--NEGATE THE NUMBER
3788  013576  012666  000012          4$:   MOV     (SP)+,12(SP)    ;;SAVE THE RESULT
3789  013602  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
3790  013604  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
3791  013606  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
3792  013610  000002                        RTI                     ;;RETURN
3793
3794  013612  005726                  5$:   TST     (SP)+           ;;CLEAN PARTIAL NUMBER FROM STACK
3795  013614  105010                        CLRB    (R0)            ;;SET A TERMINATOR
3796  013616  104401                        TYPE                    ;;TYPE THE INPUT UP TO BAD CHAR.
3797  013620  000000                  6$:   .WORD   0               ;;POINTER GOES HERE
3798  013622  104401  001252                TYPE    ,$QUES          ;;''?'' ''CR'' &'LF''
3799  013626  000720                        BR      1$              ;;TRY AGAIN
3800
3801                                  .SBTTL  TRAP DECODER
3802
3803                                  ;;**********************************************************************
3804                                  ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE ''TRAP'' INSTRUCTION
3805                                  ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3806                                  ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3807                                  ;*GO TO THAT ROUTINE.
3808
3809  013630  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
3810  013632  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
3811  013636  005740                          TST     -(R0)           ;;BACKUP BY 2
3812  013640  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
3813  013642  006300                          ASL     R0              ;;POSITION FOR INDEXING
3814  013644  016000  013664                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
3815  013650  000200                          RTS     R0              ;;GO TO ROUTINE
3816
3817
3818                                  ;;THIS IS USE TO HANDLE THE ''GETPRI'' MACRO
3819
3820  013652  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
3821  013654  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
3822  013662  000002                          RTI                     ;;RESTORE THE PSW
3823
3824                                  .SBTTL  TRAP TABLE
3825
3826                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3827                                  ;*BY THE ''TRAP'' INSTRUCTION.
3828
3829                                  ;       ROUTINE
3830                                  ;       -------
3831  013664  013652                  $TRPAD: .WORD   $TRAP2
3832  013666  013232                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
3833  013670  011200                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3834  013672  011154                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
3835  013674  011214                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
3836  013676  011402                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
3837
3838  013700  012222                          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
3839
3840  013702  012132                          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
3841  013704  012474                          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
```

```
                                        $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
                                        $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
                                        $RDDEC  ;;CALL=RDDEC    TRAP+13(104413) READ A DECIMAL NUMBER FROM TTY

                                .SBTTL  POWER DOWN AND UP ROUTINES

                                ;;******************************************************************
                                ;POWER DOWN ROUTINE
013714  012737  014060  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
013722  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
013730  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
013732  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
013734  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
013736  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
013740  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
013742  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
013744  017746  165170                MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
013750  010667  000110                MOV     SP,$SAVR6       ;;SAVE SP
013754  012737  013766  000024        MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
013762  000000                        HALT
013764  000776                        BR      .-2             ;;HANG UP

                                ;;******************************************************************
                                ;POWER UP ROUTINE
013766  012737  014060  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
013774  016706  000064                MOV     $SAVR6,SP       ;;GET SP
014000  005067  000060                CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
014004  005267  000054          1$:   INC     $SAVR6          ;;WAIT FOR THE INC
014010  001375                        BNE     1$              ;;OF  WORD
014012  012677  165122                MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
014016  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
014020  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
014022  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
014024  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
014026  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
014030  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
014032  012737  013714  000024        MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
014040  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
014046  104401                        TYPE                    ;REPORT THE POWER FAILURE
014050  014066          $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
014052  012716                        MOV     (PC)+,(SP)      ;;RESTART AT RESTRT
014054  002036          $PWRAD: .WORD   RESTRT          ;;RESTART ADDRESS
014056  000002                        RTI
014060  000000          $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
014062  000776                        BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
014064  000000          $SAVR6: 0                       ;;PUT THE SP HERE
014066  005015  047520  042527  $POWER: .ASCIZ  <15><12>'POWER'
014074  000122
                                        .EVEN

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

014076  105777  165312          XINT:   TSTB    @DLXCSR         ;'READY' SET ??
014102  100416                        BMI     1$              ;BR IF YES
```

```
3898  014104  013767  177776  165070          MOV     @#PSW,$TMP0     ;SAVE THE ERROR PSW
3899  014112  010667  165060                  MOV     SP,$REG6        ;SAVE THE ERROR STACK POINTER
3900  014116  005167  165300                  COM     XFLG0           ;SET XMIT SOFTWARE ERROR FLAG
3901  014122  042777  000100  165260          BIC     #100,@DLRCSR    ;TURN OFF THE INTERRUPT ENABLES
3902  014130  042777  000100  165256          BIC     #100,@DLXCSR
3903  014136  000411                          BR      2$              ;GO TO EXIT
3904  014140  022767  022440  165264  1$:     CMP     #DLBUFI,OPTR    ;XMITTED 256. BYTES YET ??
3905  014146  001405                          BEQ     2$              ;BR IF YES
3906  014150  117777  165256  165240          MOVB    @OPTR,@DLXDBR   ;OUTPUT A BYTE
3907  014156  005267  165250                  INC     OPTR            ;UPDATE BUFFER POINTER
3908  014162  000002                  2$:     RTI                     ;RETURN TO MAINLINE TEST
3909
3910                                  ;;*****************************************************************
3911                                  ;RECEIVER INTERRUPT SERVICE ROUTINE FOR 256. BYTE BLOCK TRANSFERS
3912                                  ;;*****************************************************************
3913
3914  014164  105777  165220          RINT:   TSTB    @DLRCSR ;'DONE' SET ??
3915  014170  100410                          BMI     1$              ;BR IF YES
3916  014172  013767  177776  165002          MOV     @#PSW,$TMP0     ;SAVE THE ERROR PSW
3917  014200  010667  164772                  MOV     SP,$REG6        ;SAVR THE ERROR STACK POINTER
3918  014204  005167  165214                  COM     RFLG0           ;SET HARD RCVR ERROR FLAG
3919  014210  000415                          BR      2$              ;GO EXIT
3920  014212  005777  165174          1$:     TST     @DLRDBR         ;ANY SOFT ERRORS ??
3921  014216  100021                          BPL     3$              ;BR IF NOT
3922  014220  013767  177776  164754          MOV     @#PSW,$TMP0     ;SAVE THE ERROR PSW
3923  014226  010667  164744                  MOV     SP,$REG6        ;SAVE THE ERROR STACK POINTER
3924  014232  017767  165154  164744          MOV     @DLRDBR,$TMP1   ;SAVE THE ERROR REGISTER IN TMP1
3925  014240  005167  165162                  COM     RFLG1           ;SET THE SOFT ERROR FLAG
3926  014244  042777  000100  165142  2$:     BIC     #100,@DLXCSR    ;TURN OFF THE INTR. ENABLES
3927  014252  042777  000100  165130          BIC     #100,@DLRCSR
3928  014260  000411                          BR      4$              ;GO TO EXIT
3929  014262  022767  023040  165144  3$:     CMP     #BUFEND,IPTR    ;RECEIVED 256. BYTES YET ??
3930  014270  001405                          BEQ     4$              ;BR IF YES
3931  014272  117777  165114  165134          MOVB    @DLRDBR,@IPTR   ;INPUT A BYTE FROM THE DL11
3932  014300  005267  165130                  INC     IPTR            ;UPDATE BUFFER POINTER
3933  014304  000002                  4$:     RTI                     ;RETURN TO MAINLINE TEST
3934
3935                                  ;THE FOLLOWING ROUTINE IS USED BY THE USER UTILITY PROGRAMS TO WAIT
3936                                  ;A SPECIFIED NO. OF MILLISECONDS BETWEEN CHARACTER TRANSFERS
3937
3938  014306  017667  000000  000034  DELAY:  MOV     @(R6),DELCNT    ;GET THE NO. OF MSEC. DELAY COUNT
3939                                                                  ;TYPED IN BY USER
3940  014314  062716  000002                  ADD     #2,(R6)         ;SET UP THIS ROUTINE'S EXIT ADDRESS
3941  014320  005767  000024                  TST     DELCNT          ;IS THE DELAY COUNT ZERO?
3942  014324  001410                          BEQ     3$              ;BRANCH IF YES
3943  014326  012746  000226          1$:     MOV     #226,-(SP)      ;PUSH A 1 MSEC. COUNT TO STACK
3944  014332  005316                  2$:     DEC     (SP)            ;DECREMENT THE 1 MSEC. COUNT BY 1
3945  014334  001376                          BNE     2$              ;BRANCH IF 1 MSEC. NOT EATEN
3946                                                                  ;AWAY YET
3947  014336  005726                          TST     (SP)+           ;RESET STACK AFTER 1 MSEC. TIME UP
3948  014340  005367  000004                  DEC     DELCNT          ;DECREMENT THE TOTAL NO. OF
3949                                                                  ;MSECS. COUNT
3950  014344  001370                          BNE     1$              ;BRANCH IF WE HAVE MORE MSECS.
3951                                                                  ;TO WAIT
3952  014346  000207                  3$:     RTS     PC              ;GO BACK TO REISSUE A CHARACTER
3953  014350  000000          DELCNT: .WORD   0                       ;THE NO. OF MSECS. NEEDED TO
```

```
                                                        ;TRANSPIRE RESIDES HERE
                                ;THE FOLLOWING ROUTINE IS USED BY USER PROGRAM #4 AND WILL ALLOW
                                ;A RANDOM NUMBER OF MILLISECONDS BEFORE TRANSMISSION OF CHARACTER
                                ;
014352  016700  000062          STALL:  MOV     NUMONE,R0       ;GET THE LOW LIMIT
014356  006100                          ROL     R0              ;MULTIPLY BY 4
014360  006100                          ROL     R0              ;
014362  066700  000054                  ADD     NUMTWO,R0       ;ADD IN THE HIGH LIMIT
014366  010067  000046                  MOV     R0,NUMONE       ;STORE THIS AS NEW LOW LIMIT
014372  006100                          ROL     R0              ;MULTIPLY NEW LOW LIMIT BY 4
014374  006100                          ROL     R0              ;
014376  066700  000040                  ADD     NUMTWO,R0       ;ADD IN THE HIGH LIMIT
014402  006100                          ROL     R0              ;MULTIPLY BY 4 AGAIN
014404  006100                          ROL     R0              ;
014406  010067  000030                  MOV     R0,NUMTWO       ;STORE THIS AS NEW HIGH LIMIT
014412  016700  000022                  MOV     NUMONE,R0       ;SAVE THE RANDOMLY GENERATED NO.
014416  046700  164664                  BIC     STLMSK,R0       ;STRIP ALL BUT 1ST 5 BITS SO AS
                                                                ;NOT TO ALLOW THE STALL TO BE TOO
                                                                ;LARGE
014422  001405                          BEQ     2$              ;BRANCH IF RESULT WAS ZERO
014424  010067  000004                  MOV     R0,1$           ;SET STALL TIME FOR DELAY ROUTINE
014430  004767  177652                  JSR     PC,DELAY        ;GO OFF TO STALL
014434  000000                  1$:     .WORD   0               ;THIS IS WHERE STALL TIME RESIDES
014436  000207                  2$:     RTS     PC              ;RETURN TO ISSUE CHARACTER
014440  001233                  NUMONE: 1233                    ;LOW LIMIT FOR RANDOM NO.
014442  007622                  NUMTWO: 7622                    ;HIGH LIMIT FOR RANDOM NO.
                                ;THE FOLLOWING ROUTINE CHECKS THE 'DONE' BIT FOR BOTH THE RECEIVER
                                ;AND TRANSMITTER.  THIS ROUTINE IS USED BY PROGRAM #4
                                ;
014444  016767  164540  000044  TIMERX: MOV     $TMP3,DUT       ;GET THE TRANSMITTER CONTROL
                                                                ;STATUS REGISTER ADDRESS
014452  162767  000004  000036          SUB     #4,DUT          ;FORM THE RECEIVER CONTROL
                                                                ;STATUS REGISTER ADDRESS
014460  000403                          BR      TCONT           ;GO TO TIME OUT THE RECEIVERS'
                                                                ;DONE BIT
014462  016767  164522  000026  TIMETX: MOV     $TMP3,DUT       ;GET THE TRANSMITTER CONTROL
                                                                ;STATUS REGISTER ADDRESS
014470  005067  164522          TCONT:  CLR     $TMP6           ;INITIALIZE A TIME COUNT
014474  005267  164516          1$:     INC     $TMP6           ;INCREMENT THE TIME COUNT
014500  001405                          BEQ     2$              ;BRANCH IF TIME COUNTER OVERFLOWED
                                                                ;INDICATING DONE BIT NEVER SET
                                                                ;WITH PLENTY OF TIME ELAPSED
014502  105777  000010                  TSTB    @DUT            ;SEE IF DONE BIT IS SET YET
014506  100372                          BPL     1$              ;WAIT SOME MORE IF IT ISN'T
014510  062716  000006                  ADD     #6,@R6          ;DONE BIT IS SET - SET UP EXIT
                                                                ;RETURN TO SKIP ERROR REPORT
014514  000207                  2$:     RTS     PC              ;RETURN TO PROGRAM #4
014516  000000                  DUT:    .WORD   0               ;THIS IS WHERE THE RCSR OR XCSR
                                                                ;ADDRESS RESIDES
                                ;THIS ROUTINE IS USED BY PROGRAMS #4 & 5, AND WILL CHECK FOR CORRECT
                                ;EXPECTED AND RECEIVED DATA, IN ADDITION TO ANY ERROR BITS
                                ;
014520  016767  164470  164470  DATCHK: MOV     $TMP5,$TMP6     ;GET THE CONTENTS OF THE RECEIVER
                                                                ;BUFFER
014526  016767  164460  164432          MOV     $TMP4,$REG2     ;STORE THE ADDRESS OF THE RECEIVER
                                                                ;DATA BUFFER
```

```
 4010  014534  016767  164426  164422          MOV     $REG2,$REG1     ;GET THE ADDRESS OF THE RECEIVER
 4011                                                                    ;DATA BUFFER
 4012  014542  162767  000002  164414          SUB     #2,$REG1        ;FORM THE ADDRESS OF THE RECEIVER
 4013                                                                    ;STATUS REGISTER FROM IT
 4014  014550  016767  164440  164412          MOV     $TMP5,$REG3     ;STORE THE CONTENTS OF THE RECEIVER
 4015                                                                    ;DATA BUFFER
 4016  014556  032767  170000  164432          BIT     #170000,$TMP6   ;ARE ANY ERROR BITS SET?
 4017  014564  001013                          BNE     1$              ;BRANCH IF YES
 4018  014566  004767  000720                  JSR     PC,UPMASK       ;GO TO MASK OFF BITS AS A FUNCTION OF
 4019                                                                    ;CHARACTER LENGTH( 5, 6, 7, OR 8 BITS)
 4020  014572  026767  164416  164432          CMP     $TMP5,$TMP14    ;WAS RECEIVED CHARACTER THE
 4021                                                                    ;SAME AS THE ONE TRANSMITTED?
 4022  014600  001406                          BEQ     2$              ;BRANCH IF YES
 4023  014602  016767  164424  164362          MOV     $TMP14,$REG4    ;STORE WHAT THE CONTENTS OF THE
 4024                                                                    ;RECEIVER DATA BUFFER SHOULD BE
 4025  014610  104010                          ERROR   +10             ;DATA RECEIVED WRONG!
 4026  014612  000401                          BR      2$              ;GET SET TO RETURN AFTER ERROR REPORT
 4027  014614  104007                  1$:     ERROR   +7              ;ERROR BIT/S SET FROM TRANSMISSION
 4028  014616  000207                  2$:     RTS     PC              ;RETURN TO PROGRAM #4
 4029
 4030                                  ;;*****************************************************************
 4031                                          ;SUBROUTINE TO SETUP ERROR INFORMATION FOR ERROR MESSAGES
 4032                                  ;;*****************************************************************
 4033
 4034
 4035  014620  013767  177776  164354  SUER2:  MOV     @#PSW,$TMP0     ;SAVE THE [PSW]
 4036  014626  016701  164556                  MOV     DLRCSR,R1       ;PUT DEVADR IN R1
 4037  014632  011203                          MOV     (R2),R3         ;PUT WAS INFO IN R3
 4038  014634  010667  164336                  MOV     SP,$REG6        ;SAVE THE [SP]
 4039  014640  062767  000002  164330          ADD     #2,$REG6        ;CORRECT FOR CALLING JSR
 4040  014646  116700  164230          SUERR1: MOVB    $TSTNM,R0       ;PUT TEST NO. IN R0
 4041  014652  010067  164304                  MOV     R0,$REG0        ;SAVE [R0] THRU [R4]
 4042  014656  010167  164302                  MOV     R1,$REG1
 4043  014662  010267  164300                  MOV     R2,$REG2
 4044  014666  010367  164276                  MOV     R3,$REG3
 4045  014672  010467  164274                  MOV     R4,$REG4
 4046  014676  000207                          RTS     PC              ;RETURN TO CALLING TEST
 4047
 4048  014700  013767  177776  164274  SUERT1: MOV     @#PSW,$TMP0             ;SAVE THE [PSW]
 4049  014706  116700  164170                  MOVB    $TSTNM,R0               ;PUT TEST NO. IN R0
 4050  014712  016701  164472                  MOV     DLRCSR,R1               ;PUT DEVADR IN R1
 4051  014716  010067  164240                  MOV     R0,$REG0                ;SAVE [R0]
 4052  014722  010167  164236                  MOV     R1,$REG1                ;SAVE [R1]
 4053  014726  013767  177776  164250  SUERT2: MOV     @#PSW,$TMP1             ;SAVE THE [PSW]
 4054  014734  010667  164236                  MOV     SP,$REG6                ;SAVE THE [SP]
 4055  014740  062767  000002  164230          ADD     #2,$REG6                ;CORRECT FOR CALLING JSR
 4056  014746  010267  164214                  MOV     R2,$REG2                ;SAVE [R2]
 4057  014752  000207                          RTS     PC                      ;RETURN
 4058
 4059                                  ;SUBROUTINE TO SETUP VECTORS FOR 256. BYTE BLOCK TRANSFER TESTS
 4060
 4061  014754  016705  164440          SUVEC:  MOV     DLVECT,R5       ;GET FIRST VECTOR ADDRESS
 4062  014760  012725  014164                  MOV     #RINT,(R5)+     ;SET UP RCVR VECTOR
 4063  014764  016725  164312                  MOV     DLPRI,(R5)+
 4064  014770  012725  014076                  MOV     #XINT,(R5)+     ;SET UP XMIT VECTOR
 4065  014774  016715  164302                  MOV     DLPRI,(R5)
```

```
4066  015000  000207                        RTS     PC              ;RETURN TO CALLER
4067
4068                                  ;SUBROUTINE TO PRIME DATA BUFFERS AND DEVICE FOR 256. BYTE TRANSFER
4069
4070  015002  005077  164406          PRIME:  CLR     @DLXCSR         ;CLEAR XMIT AND RCVR CSR'S
4071  015006  005077  164376                  CLR     @DLRCSR
4072  015012  005067  164404                  CLR     XFLG0           ;INITIALIZE ERROR FLAGS
4073  015016  005067  164402                  CLR     RFLG0
4074  015022  005067  164400                  CLR     RFLG1
4075  015026  012767  022040  164376          MOV     #DLBUF0,OPTR    ;SET UP OUTPUT POINTER
4076  015034  012767  022440  164372          MOV     #DLBUFI,IPTR    ;SET UP INPUT POINTER
4077  015042  004767  000044                  JSR     PC,CLDLBF       ;GO CLEAR THE BUFFERS
4078  015046  004777  164364                  JSR     PC,@LDOUT       ;GO SET UP THE PATTERN
4079  015052  005067  164362                  CLR     TIMR1           ;INIT TIMEOUT COUNTERS
4080  015056  012767  000036  164356          MOV     #30.,TIMR2
4081  015064  005777  164322                  TST     @DLRDBR         ;FLUSH "DONE" BIT IN RCVR CSR
4082  015070  005777  164316                  TST     @DLRDBR
4083  015074  052777  000100  164306          BIS     #100,@DLRCSR    ;ENABLE RCVR INTR.
4084  015102  052777  000104  164304          BIS     #104,@DLXCSR    ;ENABLE XMIT INTR. AND MAINT MODE
4085  015110  000207                          RTS     PC
4086
4087
4088
4089
4090                                  ;THIS ROUTINE IS CALLED TO CLEAR THE INPUT AND OUTPUT BUFFERS
4091
4092  015112  012705  022040          CLDLBF: MOV     #DLBUF0,R5      ;R5 POINTS TO BEGINNING OF BUFFER AREA
4093  015116  005025                  1$:     CLR     (R5)+           ;CLEAR A WORD
4094  015120  022705  023040                  CMP     #BUFEND,R5      ;DONE ALL WORDS ??
4095  015124  001374                          BNE     1$              ;BR IF NOT
4096  015126  000207                          RTS     PC              ;RETURN TO CALLER
4097
4098                                  ;THIS ROUTINE IS CALLED TO SET UP THE NULL-DEL-NULL PATTERN
4099
4100  015130  012705  022040          LDOUT1: MOV     #DLBUF0,R5      ;R5 POINTS TO OUTPUT BUFFER
4101  015134  105025                  1$:     CLRB    (R5)+           ;MOVE A NULL CHAR
4102  015136  112725  000377                  MOVB    #377,(R5)+      ;MOV A DEL CHAR
4103  015142  022705  022440                  CMP     #DLBUFI,R5      ;ALL DONE ??
4104  015146  001372                          BNE     1$              ;BR IF NOT
4105  015150  000207                          RTS     PC              ;RETURN TO CALLER
4106
4107                                  ;THIS ROUTINE IS USED TO LOAD AN ASCENDING BINARY COUNT PATTERN
4108
4109  015152  005005                  LDOUT2: CLR     R5              ;START WITH 000
4110  015154  110565  022040          1$:     MOVB    R5,DLBUF0(R5)   ;LOAD ONE BYTE
4111  015160  005205                          INC     R5              ;INCREMENT BYTE
4112  015162  022705  000400                  CMP     #400,R5         ;DONE 000 THRU 377 ??
4113  015166  001372                          BNE     1$              ;BR IF NOT
4114  015170  000207                          RTS     PC              ;RETURN TO CALLER
4115
4116                                  ;THIS ROUTINE IS USED TO LOAD A DESCENDING BINARY COUNT PATTERN
4117
4118  015172  112767  000377  164020  LDOUT3: MOVB    #377,$TMP7      ;START WITH A 377 BYTE
4119  015200  012705  022040                  MOV     #DLBUF0,R5      ;R5 POINTS TO OUTPUT BUFFER
4120  015204  116725  164010          1$:     MOVB    $TMP7,(R5)+     ;LOAD ONE BYTE
4121  015210  022705  022440                  CMP     #DLBUFI,R5      ;ALL DONE ??
```

```
 4122  015214  001403                          BEQ     2$              ;BR IF YES
 4123  015216  105367  163776                  DECB    $TMP7           ;GENERATE NEXT BYTE
 4124  015222  000770                          BR      1$              ;GO MOVE IT
 4125  015224  000207                  2$:     RTS     PC              ;RETURN TO CALLER
 4126
 4127
 4128                                  ;THIS ROUTINE LOADS A COMPLEMENTING WORST CASE PATTERN
 4129
 4130  015226  012705  022040          LDOUT4: MOV     #DLBUF0,R5      ;R5 POINTS TO OUTPUT BUFFER
 4131  015232  005067  163762                  CLR     $TMP7           ;INIT. BYTE GENERATOR
 4132  015236  116725  163756          1$:     MOVB    $TMP7,(R5)+     ;MOVE A BYTE
 4133  015242  105167  163752                  COMB    $TMP7           ;COMPLEMENT IT
 4134  015246  116725  163746                  MOVB    $TMP7,(R5)+     ;NOW LOAD THE 1'S COMPLEMENT
 4135  015252  105267  163743                  INCB    $TMP7+1         ;INCREMENT THE BYTE
 4136  015256  116767  163737  163734          MOVB    $TMP7+1,$TMP7   ;SET UP TO LOAD NEXT TWO
 4137  015264  022705  022440                  CMP     #DLBUF1,R5      ;ALL DONE ??
 4138  015270  001362                          BNE     1$              ;BR IF NOT
 4139  015272  000207                          RTS     PC              ;RETURN TO CALLER
 4140
 4141                                  ;THIS ROUTINE CHECKS FOR DATA COMPARE ERRORS IN 256. BYTE BLOCK TRANSFERS
 4142
 4143  015274  042777  000104  164112  CHKDAT: BIC     #104,@DLXCSR    ;DISABLE BOTH XMIT AND RCVR INTR. ENAB.
 4144  015302  042777  000100  164100          BIC     #100,@DLRCSR
 4145  015310  012702  022040                  MOV     #DLBUF0,R2      ;R2 POINTS TO S/B DATA IN OUTPUT BUFFER
 4146  015314  004767  000070                  JSR     PC,MASKING      ;GO TO MASK OFF BITS AS A FUNCTION OF
 4147                                                                  ;CHARACTER LENGTH(5, 6, 7, OR 8 BITS)
 4148  015320  012701  022440                  MOV     #DLBUF1,R1      ;R1 POINTS TO WAS DATA IN RCVR. BUFFER
 4149  015324  122221                  1$:     CMPB    (R2)+,(R1)+     ;DID S/B = WAS ??
 4150  015326  001004                          BNE     3$              ;BR IF NOT
 4151  015330  022701  023040          2$:     CMP     #BUFEND,R1      ;CHECKED ALL BYTES ??
 4152  015334  001373                          BNE     1$              ;BR IF NOT
 4153  015336  000207                          RTS     PC              ;RETURN TO CALLER
 4154  015340  013767  177776  163634  3$:     MOV     @#PSW,$TMP0     ;SAVE THE [PSW]
 4155  015346  010667  163624                  MOV     SP,$REG6        ;SAVE THE [SP]
 4156  015352  114204                          MOVB    -(R2),R4        ;GET THE S/B DATA
 4157  015354  042704  177400                  BIC     #177400,R4      ;CLEAR JUNK FROM HI BYTE
 4158  015360  114103                          MOVB    -(R1),R3        ;GET THE WAS DATA
 4159  015362  042703  177400                  BIC     #177400,R3      ;CLEAR JUNK FROM HI BYTE
 4160  015366  004767  177254                  JSR     PC,SUERR1       ;GO SET UP ERROR INFO.
 4161  015372  012767  015402  163644          MOV     #4$,$ESCAPE     ;RETURN TO 4$ AFTER ERROR PRINT
 4162  015400  104003                          ERROR+3                 ;DATA COMPARE ERROR
 4163  015402  005202                  4$:     INC     R2              ;REPOSITION BUFFER POINTERS
 4164  015404  005201                          INC     R1
 4165  015406  000750                          BR      2$              ;GO CHECK NEXT BYTE
 4166
 4167                                  ;THIS ROUTINE IS USED BY THE PATTERN TESTS
 4168                                  ;IT WILL MASK OFF THE CHARACTER SENT OUT BY THE XMITTER
 4169                                  ;BEFORE THE COMPARISON OF DATA OF WHAT WAS RECEIVED AND WHAT WAS TRANSMITTED
 4170                                  ;IS DONE.  THE MASKING IS DONE AS A FUNCTION OF CHARACTER LENGTH WHICH
 4171                                  ;CAN BE EITHER 5, 6, 7, OR 8 BITS .
 4172
 4173  015410  005005                  MASKING:        CLR     R5      ;INITIALIZE TABLE OFFSET
 4174                                                                  ;FOR PICKING UP MASK WORD
```

```
 4175  015412  022767  000010  163614          CMP     #8.,$TMP15      ;IS THE CHARACTER LENGTH 8 BITS?
 4176  015420  001427                          BEQ     3$              ;BRANCH IF IT IS
 4177  015422  062705  000002                  ADD     #2,R5           ;SET UP FOR NEXT MASK WORD
 4178                                                                  ;IT COULD BE THIS ONE
 4179  015426  022767  000007  163600          CMP     #7.,$TMP15      ;IS THE CHARACTER LENGTH 7 BITS?
 4180  015434  001410                          BEQ     1$              ;BRANCH IF IT IS
 4181  015436  062705  000002                  ADD     #2,R5           ;SET UP FOR NEXT MASK WORD
 4182                                                                  ;IT COULD BE THIS ONE
 4183  015442  022767  000006  163564          CMP     #6.,$TMP15      ;IS THE CHARACTER LENGTH 6 BITS?
 4184  015450  001402                          BEQ     1$              ;BRANCH IF IT IS
 4185  015452  062705  000002                  ADD     #2,R5           ;SET UP FOR NEXT MASK WORD
 4186                                                                  ;IT MUST BE THIS ONE!!!!
 4187  015456  016505  015502          1$:     MOV     CHARL(R5),R5    ;PICK UP THE MASK WORD
 4188  015462  005105                          COM     R5              ;FORM THE BITS THAT ARE TO BE MASKED
 4189  015464  140522                  2$:     BICB    R5,(R2)+        ;MASK A BYTE
 4190  015466  022702  022440                  CMP     #DLBUFI,R2      ;ARE WE AT THE END OF THE XMITTER
 4191                                                                  ;OUTPUT BUFFER
 4192  015472  001374                          BNE     2$              ;BRANCH IF NO TO MASK NEXT BYTE
 4193  015474  012702  022040                  MOV     #DLBUFO,R2      ;RESTORE R2 BEFORE RETURNING
 4194  015500  000207                  3$:     RTS     PC              ;RETURN TO MAINLINE CODE
 4195                                  ;TABLE OF MASK WORDS
 4196  015502  000377                  CHARL:  .WORD   377             ;8. BITS IN LENGTH
 4197  015504  000177                          .WORD   177             ;7. BITS IN LENGTH
 4198  015506  000077                          .WORD   77              ;6. BITS IN LENGTH
 4199  015510  000037                          .WORD   37              ;5. BITS IN LENGTH
 4200
 4201                                  ;THIS ROUTINE IS USED BY PROGRAMS #4 & 5
 4202                                  ;IT WILL MASK OFF THE CHARACTER SENT OUT BY THE TRANSMITTER
 4203                                  ;BEFORE THE COMPARISON OF DATA OF WHAT WAS RECEIVED AND WHAT WAS
 4204                                  ;TRANSMITTED IS DONE.  THE MASKING IS DONE AS A FUNCTION OF CHARACTER
 4205                                  ;LENGTH WHICH CAN BE EITHER 5, 6, 7, OR 8 BITS.
 4206
 4207  015512  016767  163466  163512  UPMASK: MOV     $TMP1,$TMP14    ;PICK UP THE CHARACTER THAT WAS
 4208                                                                  ;SENT OUT FROM THE XMITTER
 4209  015520  005005                          CLR     R5              ;INITIIALIZE TABLE OFFSET
 4210                                                                  ;FOR PICKING UP MASK WORD
 4211  015522  022767  000010  163504          CMP     #8.,$TMP15      ;IS THE CHARACTER LENGTH 8 BITS?
 4212  015530  001423                          BEQ     2$              ;BRANCH IF IT IS
 4213  015532  062705  000002                  ADD     #2,R5           ;SET UP FOR NEXT MASK WORD
 4214                                                                  ;IT COULD BE THIS ONE
 4215  015536  022767  000007  163470          CMP     #7.,$TMP15      ;IS THE CHARACTER LENGTH 7 BITS?
 4216  015544  001410                          BEQ     1$              ;BRANCH IF IT IS
 4217  015546  062705  000002                  ADD     #2,R5           ;SET UP FOR NEXT MASK WORD
 4218                                                                  ;IT COULD BE THIS ONE
 4219  015552  022767  000006  163454          CMP     #6.,$TMP15      ;IS THE CHARACTER LENGTH 6 BITS?
 4220  015560  001402                          BEQ     1$              ;BRANCH IF IT IS
 4221  015562  062705  000002                  ADD     #2,R5           ;SET UP FOR NEXT MASK WORD
 4222                                                                  ;IT MUST BE THIS ONE!!!!
 4223  015566  016505  015502          1$:     MOV     CHARL(R5),R5    ;PICK UP THE MASK WORD
 4224  015572  005105                          COM     R5              ;FORM THE BITS THAT ARE TO BE MASKED
 4225  015574  140567  163432                  BICB    R5,$TMP14       ;MASK THE LOW BYTE
 4226  015600  000207                  2$:     RTS     PC              ;RETURN TO MAINLINE CODE
 4227
 4228                                  ;ROUTINE TO SERVICE BUS ERROR TRAPS
 4229
 4230  015602  112767  000060  000632  BUSERR: MOVB    #60,EM4+46      ;SET UP ERROR MESSAGE
```

```
 4231  015610  112767  000060  000625          MOVB    #60,EM4+47
 4232  015616  112767  000064  000620          MOVB    #64,EM4+50
 4233  015624  000412                          BR      TRPCOM          ;GO SET UP AND REPORT BUS ERROR
 4234
 4235                                  ;ROUTINE TO SERVICE RSVD INSTRUCTION TRAPS
 4236
 4237  015626  112767  000060  000606  RSVERR: MOVB    #60,EM4+46      ;SET UP ERROR MESSAGE
 4238  015634  112767  000061  000601          MOVB    #61,EM4+47
 4239  015642  112767  000060  000574          MOVB    #60,EM4+50
 4240  015650  000400                          BR      TRPCOM          ;GO SET UP AND REPORT RSVD INSTR. ERROR
 4241
 4242                                  ;ROUTINE TO SET UP AND REPORT BUS ERROR AND RSVD INSTR ERRORS
 4243
 4244  015652  010667  163320          TRPCOM: MOV     SP,$REG6        ;SAVE THE TRAP SP
 4245  015656  116700  163220                  MOVB    $TSTNM,R0       ;PUT TEST NO. IN R0
 4246  015662  010067  163274                  MOV     R0,$REG0        ;SAVE TEST #
 4247  015666  016667  000002  163306          MOV     2(SP),$TMP0     ;SAVE THE ERROR PSW
 4248  015674  012767  015710  163342          MOV     #1$,$ESCAPE     ;GO TO 1$ AFTER ERROR PRINT
 4249  015702  011667  163272                  MOV     (SP),$REG7      ;SAVE THE ERROR PC
 4250  015706  104004                          ERROR+4                 ;REPORTED TRAP ERROR
 4251  015710  000137  002036          1$:     JMP     @#RESTRT        ;ATTEMPT TO RESTART THE PROGRAM
 4252                                                                  ;AND TRY AGAIN
 4253
 4254
 4255                                  ;;*****************************************************************
 4256                                  ;ERROR MESSAGE INFORMATION
 4257                                  ;;*****************************************************************
 4258
 4259                                  ;INFORMATION FOR ERROR MESSAGE 1
 4260
 4261  015714  046104  030461  051040  EM1:     .ASCIZ  'DL11 REGISTER REFERENCE CAUSED TIMEOUT'
 4262  015722  043505  051511  042524
 4263  015730  020122  042522  042506
 4264  015736  042522  041516  020105
 4265  015744  040503  051525  042105
 4266  015752  052040  046511  047505
 4267  015760  052125     000
 4268  015763     040  050050  024503  DH1:     .ASCIZ  ' (PC)    (PS)    (SP)    TEST   DEVADR  REGADR'
 4269  015770  020040  020040  050050
 4270  015776  024523  020040  020040
 4271  016004  051450  024520  020040
 4272  016012  020040  042524  052123
 4273  016020  020040  042040  053105
 4274  016026  042101  020122  051040
 4275  016034  043505  042101  000122
 4276                                  .EVEN
 4277  016042  001116  001202  001176  DT1:     .WORD   $ERRPC,$TMP0,$REG6,$REG0,$REG1,$REG2,0
 4278  016050  001162  001164  001166
 4279  016056  000000
 4280
 4281                                  ;INFORMATION FOR ERROR MESSAGE 2
 4282
 4283  016060  046104  030461  051040  EM2:     .ASCIZ  'DL11 REGISTER ERROR'
 4284  016066  043505  051511  042524
 4285  016074  020122  051105  047522
 4286  016102  000122
```

```
 4287  016104  024040  041520  020051  DH2:    .ASCIZ  ' (PC)    (PS)    (SP)    TEST   DEVADR  REGADR   WAS     S/B'
 4288  016112  020040  024040  051520
 4289  016120  020051  020040  024040
 4290  016126  050123  020051  020040
 4291  016134  052040  051505  020124
 4292  016142  020040  042504  040526
 4293  016150  051104  020040  042522
 4294  016156  040507  051104  020040
 4295  016164  053440  051501  020040
 4296  016172  020040  051440  041057
 4297  016200     000
 4298             016202                .EVEN
 4299  016202  001116  001202  001176  DT2:    .WORD   $ERRPC,$TMP0,$REG6,$REG0,$REG1,$REG2,$REG3,$REG4,0
 4300  016210  001162  001164  001166
 4301  016216  001170  001172  000000
 4302
 4303                                  ;INFORMATION FOR MESSAGE 3
 4304
 4305  016224  046104  030461  042040  EM3:    .ASCIZ  'DL11 DATA COMPARE ERROR'
 4306  016232  052101  020101  047503
 4307  016240  050115  051101  020105
 4308  016246  051105  047522  000122
 4309  016254  024040  041520  020051  DH3:    .ASCIZ  ' (PC)    (PS)    (SP)    TEST   WASADR  SHBADR   WAS     S/B'
 4310  016262  020040  024040  051520
 4311  016270  020051  020040  024040
 4312  016276  050123  020051  020040
 4313  016304  052040  051505  020124
 4314  016312  020040  040527  040523
 4315  016320  051104  020040  044123
 4316  016326  040502  051104  020040
 4317  016334  020040  040527  020123
 4318  016342  020040  020040  027523
 4319  016350  000102
 4320                                  .EVEN
 4321  016352  001116  001202  001176  DT3:    .WORD   $ERRPC,$TMP0,$REG6,$REG0,$REG1,$REG2,$REG3,$REG4,0
 4322  016360  001162  001164  001166
 4323  016366  001170  001172  000000
 4324
 4325                                  ;INFORMATION FOR MESSAGE 4
 4326
 4327  016374  047125  054105  042520  EM4:    .ASCIZ  'UNEXPECTED TRAP TO VECTOR AT LOCATION    '
 4328  016402  052103  042105  052040
 4329  016410  040522  020120  047524
 4330  016416  053040  041505  047524
 4331  016424  020122  052101  046040
 4332  016432  041517  052101  047511
 4333  016440  020116  020040  000040
 4334  016446  024040  041520  020051  DH4:    .ASCIZ  ' (PC)    (PS)    (SP)    TEST'
 4335  016454  020040  024040  051520
 4336  016462  020051  020040  024040
 4337  016470  050123  020051  020040
 4338  016476  052040  051505  000124
 4339                                  .EVEN
 4340  016504  001200  001202  001176  DT4:    .WORD   $REG7,$TMP0,$REG6,$REG0,0
 4341  016512  001162  000000
 4342
```

```
                                        ;ERROR INFORMATION FOR ERROR MESSAGE 5
                                        ;
016516  046104  030461  051440  EM5:    .ASCIZ  'DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN)'
016524  043117  020124  051105
016532  047522  020122  050050
016540  051101  052111  026131
016546  051106  046501  047111
016554  026107  047440  020122
016562  053117  051105  052522
016570  024516     000
016573     040  050050  024503  DH5:    .ASCIZ  ' (PC)    (PS)    (SP)    TEST   DEVADR  REGADR   (REG)'
016600  020040  020040  050050
016606  024523  020040  020040
016614  051450  024520  020040
016622  020040  042524  052123
016630  020040  042040  053105
016636  042101  020122  051040
016644  043505  042101  020122
016652  020040  051050  043505
016660  000051
                                        .EVEN
016662  001116  001202  001176  DT5:    .WORD   $ERRPC,$TMP0,$REG6,$REG0,$REG1,$REG2,$REG3,0
016670  001162  001164  001166
016676  001170  000000
                                        ;
                                        ;INFORMATION FOR ERROR MESSAGE 6
                                        ;
016702  024040  041520  020051  DH6:    .ASCIZ  ' (PC)    (PS)    (SP)   REGADR'
016710  020040  024040  051520
016716  020051  020040  024040
016724  050123  020051  020040
016732  042522  040507  051104
016740     000
        016742
                                        .EVEN
016742  001116  001204  001176  DT6:    .WORD   $ERRPC,$TMP1,$REG6,$REG2,0
016750  001166  000000
                                        ;
                                        ;INFORMATION FOR ERROR MESSAGE 7
                                        ;
016754  024040  041520  020051  DH7:    .ASCIZ  ' (PC)   DEVADR  REGADR   (REG)'
016762  020040  042504  040526
016770  051104  020040  042522
016776  040507  051104  020040
017004  024040  042522  024507
017012     000
        017014
                                        .EVEN
017014  001116  001164  001166  DT7:    .WORD   $ERRPC,$REG1,$REG2,$REG3,0
017022  001170  000000
                                        ;
                                        ;INFORMATION FOR ERROR MESSAGE 10
                                        ;
017026  024040  041520  020051  DH10:   .ASCIZ  ' (PC)   DEVADR  REGADR   (REG)    S/B'
017034  020040  042504  040526
017042  051104  020040  042522
017050  040507  051104  020040
017056  024040  042522  024507
```

```
 4399  017064  020040  020040  027523
 4400  017072  000102
 4401                                  .EVEN
 4402  017074  001116  001164  001166  DT10:   .WORD   $ERRPC,$REG1,$REG2,$REG3,$REG4,0
 4403  017102  001170  001172  000000
 4404                                  ;MISCELLANEOUS MESSAGES
 4405
 4406  017110  052516  046114  042055  XMSG1:  .ASCIZ  'NULL-DEL-NULL SEQUENCE TIMEOUT AT FOLLOWING PC'
 4407  017116  046105  047055  046125
 4408  017124  020114  042523  052521
 4409  017132  047105  042503  052040
 4410  017140  046511  047505  052125
 4411  017146  040440  020124  047506
 4412  017154  046114  053517  047111
 4413  017162  020107  041520     000
 4414  017167     102  047111  051101  XMSG2:  .ASCIZ  'BINARY UP COUNT SEQUENCE TIMEOUT AT FOLLOWING PC'
 4415  017174  020131  050125  041440
 4416  017202  052517  052116  051440
 4417  017210  050505  042525  041516
 4418  017216  020105  044524  042515
 4419  017224  052517  020124  052101
 4420  017232  043040  046117  047514
 4421  017240  044527  043516  050040
 4422  017246  000103
 4423  017250  044502  040516  054522  XMSG3:  .ASCIZ  'BINARY DOWN COUNT SEQUENCE TIMEOUT AT FOLLOWING PC'
 4424  017256  042040  053517  020116
 4425  017264  047503  047125  020124
 4426  017272  042523  052521  047105
 4427  017300  042503  052040  046511
 4428  017306  047505  052125  040440
 4429  017314  020124  047506  046114
 4430  017322  053517  047111  020107
 4431  017330  041520     000
 4432  017333     127  051117  052123  XMSG4:  .ASCIZ  'WORST CASE PATTERN SEQUENCE TIMEOUT AT FOLLOWING PC'
 4433  017340  041440  051501  020105
 4434  017346  040520  052124  051105
 4435  017354  020116  042523  052521
 4436  017362  047105  042503  052040
 4437  017370  046511  047505  052125
 4438  017376  040440  020124  047506
 4439  017404  046114  053517  047111
 4440  017412  020107  041520     000
 4441
 4442  017417     015  041412  042132  STMES:  .ASCIZ  <15><12>'CZDLCCO DL11-C,D,E OFLNE TST'<15><12>
 4443  017424  041514  030103  042040
 4444  017432  030514  026461  026103
 4445  017440  026104  020105  043117
 4446  017446  047114  020105  051524
 4447  017454  006524  000012
 4448
 4449  017460  005015  047531  020125  PROG2M: .ASCIZ  <15><12>'YOU HAVE SELECTED PROGRAM NO. 2'<15><12>
 4450  017466  040510  042526  051440
 4451  017474  046105  041505  042524
 4452  017502  020104  051120  043517
 4453  017510  040522  020115  047516
 4454  017516  020056  006462  000012
```

```
 4455  017524  005015  047531  020125  PROG3M: .ASCIZ  <15><12>'YOU HAVE SELECTED PROGRAM NO. 3'<15><12>
 4456  017532  040510  042526  051440
 4457  017540  046105  041505  042524
 4458  017546  020104  051120  043517
 4459  017554  040522  020115  047516
 4460  017562  020056  006463  000012
 4461  017570  005015  047531  020125  PROG4M: .ASCIZ  <15><12>'YOU HAVE SELECTED PROGRAM NO. 4'<15><12>
 4462  017576  040510  042526  051440
 4463  017604  046105  041505  042524
 4464  017612  020104  051120  043517
 4465  017620  040522  020115  047516
 4466  017626  020056  006464  000012
 4467  017634  005015  047531  020125  PROG5M: .ASCIZ  <15><12>'YOU HAVE SELECTED PROGRAM NO. 5'<15><12>
 4468  017642  040510  042526  051440
 4469  017650  046105  041505  042524
 4470  017656  020104  051120  043517
 4471  017664  040522  020115  047516
 4472  017672  020056  006465  000012
 4473  017700  005015  051124  047101  XDB:    .ASCIZ  <15><12>'TRANSMITTER DONE BIT NEVER SET  PC= '
 4474  017706  046523  052111  042524
 4475  017714  020122  047504  042516
 4476  017722  041040  052111  047040
 4477  017730  053105  051105  051440
 4478  017736  052105  020040  041520
 4479  017744  020075     000
 4480  017747     015  051012  041505  RDB:    .ASCIZ  <15><12>'RECEIVER DONE BIT NEVER SET  PC= '
 4481  017754  044505  042526  020122
 4482  017762  047504  042516  041040
 4483  017770  052111  047040  053105
 4484  017776  051105  051440  052105
 4485  020004  020040  041520  020075
 4486  020012     000
 4487                                  ;MESSAGES SEEKING USER RESPONSE
 4488
 4489  020013     015  053412  040510  LENGTH: .ASCIZ  <15><12>'WHAT IS THE CHARACTER LENGTH (5,6,7 OR 8 BITS)?'
 4490  020020  020124  051511  052040
 4491  020026  042510  041440  040510
 4492  020034  040522  052103  051105
 4493  020042  046040  047105  052107
 4494  020050  020110  032450  033054
 4495  020056  033454  047440  020122
 4496  020064  020070  044502  051524
 4497  020072  037451     000
 4498  020075     015  042012  020117  DEFAULT:        .ASCII  <15><12>'DO YOU WISH TO TEST OTHER THAN THE'
 4499  020102  047531  020125  044527
 4500  020110  044123  052040  020117
 4501  020116  042524  052123  047440
 4502  020124  044124  051105  052040
 4503  020132  040510  020116  044124
 4504  020140     105
 4505  020141     015  042012  043105          .ASCIZ  <15><12>'DEFAULT DEVICE (1/0 = YES/NO)?'
 4506  020146  052501  052114  042040
 4507  020154  053105  041511  020105
 4508  020162  030450  030057  036440
 4509  020170  054440  051505  047057
 4510  020176  024517  000077
```

```
 4511  020202  005015  044127  052101  MFIRSTD:.ASCIZ  <15><12>'WHAT IS THE 1ST RECEIVER STATUS REGISTER ADDRESS?  '
 4512  020210  044440  020123  044124
 4513  020216  020105  051461  020124
 4514  020224  042522  042503  053111
 4515  020232  051105  051440  040524
 4516  020240  052524  020123  042522
 4517  020246  044507  052123  051105
 4518  020254  040440  042104  042522
 4519  020262  051523  020077  000040
 4520  020270  005015  044127  052101  MVECT:  .ASCIZ  <15><12>'WHAT IS THE 1ST RECEIVER''S VECTOR ADDRESS?  '
 4521  020276  044440  020123  044124
 4522  020304  020105  051461  020124
 4523  020312  042522  042503  053111
 4524  020320  051105  020123  042526
 4525  020326  052103  051117  040440
 4526  020334  042104  042522  051523
 4527  020342  020077  000040
 4528  020346  005015  047504  054440  MULDEV: .ASCIZ  <15><12>'DO YOU WANT TO TEST MULTIPLE DEVICES 1/0=YES/NO?  '
 4529  020354  052517  053440  047101
 4530  020362  020124  047524  052040
 4531  020370  051505  020124  052515
 4532  020376  052114  050111  042514
 4533  020404  042040  053105  041511
 4534  020412  051505  030440  030057
 4535  020420  054475  051505  047057
 4536  020426  037517  020040     000
 4537  020433     015  053412  040510  MLASTD: .ASCIZ  <15><12>'WHAT IS THE STATUS REGISTER ADDRESS OF THE LAST RECEIVER?  '
 4538  020440  020124  051511  052040
 4539  020446  042510  051440  040524
 4540  020454  052524  020123  042522
 4541  020462  044507  052123  051105
 4542  020470  040440  042104  042522
 4543  020476  051523  047440  020106
 4544  020504  044124  020105  040514
 4545  020512  052123  051040  041505
 4546  020520  044505  042526  037522
 4547  020526  020040     000
 4548  020531     015  051412  046517  MRANGE: .ASCIZ  <15><12>'SOMETHING WRONG-ANSWER THE LAST QUESTION AGAIN!  '
 4549  020536  052105  044510  043516
 4550  020544  053440  047522  043516
 4551  020552  040455  051516  042527
 4552  020560  020122  044124  020105
 4553  020566  040514  052123  050440
 4554  020574  042525  052123  047511
 4555  020602  020116  043501  044501
 4556  020610  020516  020040     000
 4557  020615     015  053412  040510  PLEVEL: .ASCIZ  <15><12>'WHAT IS YOUR INTERRUPT PRIORITY LEVEL?  '
 4558  020622  020124  051511  054440
 4559  020630  052517  020122  047111
 4560  020636  042524  051122  050125
 4561  020644  020124  051120  047511
 4562  020652  044522  054524  046040
 4563  020660  053105  046105  020077
 4564  020666  000040
 4565  020670  005015  051120  043517  FOULUP: .ASCII  <15><12>'PROGRAM DEVICE ACTIVE LOCATION SHOWS NO DEVICE ACTIVE'
 4566  020676  040522  020115  042504
```

```
 4567  020704  044526  042503  040440
 4568  020712  052103  053111  020105
 4569  020720  047514  040503  044524
 4570  020726  047117  051440  047510
 4571  020734  051527  047040  020117
 4572  020742  042504  044526  042503
 4573  020750  040440  052103  053111
 4574  020756     105
 4575  020757     015  051412  052105          .ASCII  <15><12>'SET SWITCH 0 TO A ONE (1) AND'
 4576  020764  051440  044527  041524
 4577  020772  020110  020060  047524
 4578  021000  040440  047440  042516
 4579  021006  024040  024461  040440
 4580  021014  042116
 4581  021016  005015  044510  020124          .ASCIZ  <15><12>'HIT CONTINUE TO GO BACK TO DEVICE SELECTION AGAIN'
 4582  021024  047503  052116  047111
 4583  021032  042525  052040  020117
 4584  021040  047507  041040  041501
 4585  021046  020113  047524  042040
 4586  021054  053105  041511  020105
 4587  021062  042523  042514  052103
 4588  021070  047511  020116  043501
 4589  021076  044501  000116
 4590  021102  005015  044127  052101  LINTAD: .ASCIZ  <15><12>'WHAT IS THE TRANSMITTER DATA BUFFER ADDRESS?  '
 4591  021110  044440  020123  044124
 4592  021116  020105  051124  047101
 4593  021124  046523  052111  042524
 4594  021132  020122  040504  040524
 4595  021140  041040  043125  042506
 4596  021146  020122  042101  051104
 4597  021154  051505  037523  020040
 4598  021162     000
 4599  021163     015  053412  040510  SELCAR: .ASCIZ  <15><12>'WHAT IS THE CHARACTER TO BE TRANSMITTED (OCTAL ASCII E.G. A=101
 4600  021170  020124  051511  052040
 4601  021176  042510  041440  040510
 4602  021204  040522  052103  051105
 4603  021212  052040  020117  042502
 4604  021220  052040  040522  051516
 4605  021226  044515  052124  042105
 4606  021234  024040  041517  040524
 4607  021242  020114  051501  044503
 4608  021250  020111  027105  027107
 4609  021256  040440  030475  030460
 4610  021264  037451  020040     000
 4611  021271     015  053412  040510  SELDLY: .ASCIZ  <15><12>'WHAT IS THE DESIRED MSEC. DELAY (OCTAL E.G. 10=8(10))?  '
 4612  021276  020124  051511  052040
 4613  021304  042510  042040  051505
 4614  021312  051111  042105  046440
 4615  021320  042523  027103  042040
 4616  021326  046105  054501  024040
 4617  021334  041517  040524  020114
 4618  021342  027105  027107  030440
 4619  021350  036460  024070  030061
 4620  021356  024451  020077  000040
 4621  021364  005015  051511  040440  RSTALL: .ASCIZ  <15><12>'IS A RANDOM WAIT TIME (MSEC.) DESIRED  1/0=YES/NO?  '
 4622  021372  051040  047101  047504
```

```
 4623  021400  020115  040527  052111
 4624  021406  052040  046511  020105
 4625  021414  046450  042523  027103
 4626  021422  020051  042504  044523
 4627  021430  042522  020104  030440
 4628  021436  030057  054475  051505
 4629  021444  047057  037517  020040
 4630  021452     000
 4631  021453     015  054412  052517  FAILSA: .ASCII  <15><12>'YOU HAVE SWR8 SET INDICATING LOOP ON TEST'
 4632  021460  044040  053101  020105
 4633  021466  053523  034122  051440
 4634  021474  052105  044440  042116
 4635  021502  041511  052101  047111
 4636  021510  020107  047514  050117
 4637  021516  047440  020116  042524
 4638  021524  052123
 4639  021526  005015  040510  042526          .ASCII  <15><12>'HAVE YOU MODIFIED THE PROPER LOCATIONS FOR THE'
 4640  021534  054440  052517  046440
 4641  021542  042117  043111  042511
 4642  021550  020104  044124  020105
 4643  021556  051120  050117  051105
 4644  021564  046040  041517  052101
 4645  021572  047511  051516  043040
 4646  021600  051117  052040  042510
 4647  021606  005015  042504  044526          .ASCII  <15><12>'DEVICE THAT YOU WANT TO TEST?'
 4648  021614  042503  052040  040510
 4649  021622  020124  047531  020125
 4650  021630  040527  052116  052040
 4651  021636  020117  042524  052123
 4652  021644     077
 4653  021645     015  044412  020106          .ASCII  <15><12>'IF SO - PRESS THE CONTINUE SWITCH'
 4654  021652  047523  026440  050040
 4655  021660  042522  051523  052040
 4656  021666  042510  041440  047117
 4657  021674  044524  052516  020105
 4658  021702  053523  052111  044103
 4659  021710  005015  043111  04704SG:          .ASCII  ' = PC'
 4676  022036  000040                                    .ASCIZ  ' '
 4677
 4678                                  .EVEN
```

```
4679                                    ;512. WORDS RESERVED FOR TWO 256. BYTE INPUT/OUTPUT DATA BUFFERS
4680
4681  022040  000400                    DLBUFO: .BLKB   256.            ;RSVD FOR OUTPUT BUFFER
4682                                                                    ;THIS IS THE DATA BEING SENT OUT
4683                                                                    ;BY THE TRANSMITTER
4684  022440  000400                    DLBUFI: .BLKB   256.            ;RSVD FOR INPUT BUFFER
4685                                                                    ;THIS IS THE DATA THAT WAS PICKED
4686                                                                    ;UP BY THE RECEIVER (I.E. DATA
4687                                                                    ;SENT BY THE TRANSMITTER - HOPEFULLY)
4688  023040  000000                    BUFEND: 0                       ;TAG MARKS END OF BUFFERS
4689
4690          000001                    .END
```

```
ACTREG  001274          1459#  1827*  1859*  1860*  1870*  2974   2998
BASEAD  001264          1445#  1779*  1865*  1866   1873*  1881*  2986*  3002   3021*  3023
BASEIV  001270          1452#  1799*  2989*  3004   3022*  3024
BEGIN   001452          1243   1588#
BIT0  = 000001          1345#  1766   1789   1847   2301   2307   2320   2577   2661   2742   2847
BIT00 = 000001          1335#  1345
BIT01 = 000002          1334#  1344
BIT02 = 000004          1333#  1343
BIT03 = 000010          1332#  1342
BIT04 = 000020          1331#  1341
BIT05 = 000040          1330#  1340
BIT06 = 000100          1329#  1339
BIT07 = 000200          1328#  1338
BIT08 = 000400          1327#  1337   3087
BIT09 = 001000          1326#  1336   3095   3156
BIT1  = 000002          1344#  2212   2226   2271
BIT10 = 002000          1325#  3140
BIT11 = 004000          1324#  3102
BIT12 = 010000          1323#
BIT13 = 020000          1322#  3147
BIT14 = 040000          1321#  3073
BIT15 = 100000          1320#  2146   2158   2177   2189   2191   2213   2225   2227   2285
BIT2  = 000004          1343#  2018   2024   2091   2102   2115   2145   2157   2707   2805   2910
BIT3  = 000010          1342#  2176   2190
BIT4  = 000020          1341#
BIT5  = 000040          1340#  2246   2252   2270
BIT6  = 000100          1339#  2067   2073   2090   2117
BIT7  = 000200          1338#
BIT8  = 000400          1337#
BIT9  = 001000          1336#
BPTVEC= 000014          1352#
BUFEND  023040          2342   2399   2456   2513   3929   4094   4151   4688#
BUSERR  015602          1692   4230#
CHARL   015502          4187   4196#  4223
CHKDAT  015274          2344   2401   2458   2515   4143#
CKSWR = 104407          3072   3136   3155   3840#
CLDLBF  015112          4077   4092#
CONQUE  002672          1832   1876   1888#  1929
CR    = 000015          1260#  3729   3739
CRLF  = 000200          1261#  3700   3739
DATCHK  014520          2819   2923   4006#
DDISP = 177570          1267#  1389   1615
DEFAUL  020075          1737   4498#
DELAY   014306          2625   2712   3938#  3975
DELCNT  014350          3938*  3941   3948*  3953#
DH1     015763          1510   4268#
DH10    017026          1559   4394#
DH2     016104          1517   4287#
DH3     016254          1524   4309#
DH4     016446          1531   4334#
DH5     016573          1538   4353#
DH6     016702          1545   4370#
DH7     016754          1552   4382#
DISPLA  001142          1389#  1615*  1623*  3116*  3139*
DISPRE  000174          1232#  1623
DLADDR  010000          1686   1777   2941#  3006   3026
```

```
DLBASE  001260          1439# 1685* 1773* 1776  2941  2943* 2945  2947* 2949  2951* 2953  3002* 3023*
DLBUFI  022440          3904  4076  4103  4121  4137  4148  4190  4684#
DLBUFO  022040          4075  4092  4100  4110* 4119  4130  4145  4193  4681#
DLPRI   001302          1475# 1646* 1902* 1908* 1909* 1910* 1911* 1912* 1913  2037  2084  2265  4063
                        4065
DLRCSR  001410          1567# 1714* 1920* 1938  1997  2066  2088  2143  2146  2158  2173  2177  2191
                        2209  2213  2227  2244  2269  2350* 2357  2366  2375  2407* 2414  2423  2432
                        2464* 2471  2480  2489  2521* 2527  2535  2543  2941* 3011* 3901* 3914  3927*
                        4036  4050  4071* 4083* 4144*
DLRDBR  001412          1568# 1715  1716  1921  1922  1968  2125  2376  2433  2490  2544  2945* 3012
                        3013  3920  3924  3931  4081  4082
DLVECT  001420          1571# 1688* 1796* 1798  1799  2035  2082  2263  3004* 3024* 4061
DLXCSR  001414          1569# 1713* 1919* 1953  2007  2017  2040  2300  2319  2351* 2358  2408* 2415
                        2465* 2472  2522* 2528  2949* 3010* 3896  3902* 3926* 4070* 4084* 4143*
DLXDBR  001416          1570# 1983  2126  2953* 3906*
DSWR  = 177570          1266# 1388  1614
DT1     016042          1511  4277#
DT10    017074          1560  4402#
DT2     016202          1518  4299#
DT3     016352          1525  4321#
DT4     016504          1532  4340#
DT5     016662          1539  4364#
DT6     016742          1546  4377#
DT7     017014          1553  4389#
DUT     014516          3983* 3985* 3989* 3996  4001#
EMTVEC= 000030          1355# 1599* 1600*
EM1     015714          1509  1544  4261#
EM2     016060          1516  4283#
EM3     016224          1523  1558  4305#
EM4     016374          1530  4230* 4231* 4232* 4237* 4238* 4239* 4327#
EM5     016516          1537  1551  4345#
ERRVEC= 000004          1348# 1612  1613* 1624* 1692* 1693* 1936  1937* 1945* 1951  1952* 1960* 1966
                        1967* 1975* 1981  1982* 1990* 2591  2592* 2601* 2603* 2674  2675* 2684* 2686*
                        2755  2756* 2765* 2767* 2860  2861* 2870* 2872* 3078  3079* 3081* 3084*
FAILSA  021453          1669  4631#
FIRSTD  002244          1755# 1807
FLUSH   002770          1742  1919#
FOULUP  020670          2976  4565#
GNS   = ****** U        3832  3833  3834  3835  3836  3838  3840  3841  3842  3843  3844
GO      002144          1712  1726# 1804
GTSWR = 104406          1631  3838#
HOWMAN  002442          1800  1811#
HT    = 000011          1258# 3698  3739
INTFLG  001446          1582# 2033* 2042  2054* 2085* 2097  2118* 2267* 2272  2283*
IOTVEC= 000020          1353# 1597* 1598*
IPTR    001434          1577# 2342  2399  2456  2513  3929  3931* 3932* 4076*
KEEPAD  001262          1441# 1776* 1779  1873  1881  3021
KEEPIV  001266          1448# 1798* 3022
LASTAD  001300          1471# 1854* 1866
LASTD   002512          1826  1833# 1926
LDOUT   001436          1578# 2334* 2391* 2448* 2505* 4078
LDOUT1  015130          2334  4100#
LDOUT2  015152          2391  4109#
LDOUT3  015172          2448  4118#
LDOUT4  015226          2505  4130#
LENGTH  020013          1726  2785  2876  4489#
```

```
LESS1   001304          1477#   1913*   1915*   1917*
LF    = 000012          1259#   3733    3739
LINTAD  021102          2566    2650    2731    2836    4590#
MASKIN  015410          4146    4173#
MFIRST  020202          1755    4511#
MLASTD  020433          1836    4537#
MRANGE  020531          1883    4548#
MULDEV  020346          1811    4528#
MULTD   001272          1455#   1637*   1818*   1821*   1824    2971    2981*
MVECT   020270          1780    4520#
NUMONE  014440          3958    3962*   3969    3978#
NUMTWO  014442          3961    3965    3968*   3979#
ONCE    002120          1673    1702    1713#
OPTR    001432          1576#   3904    3906    3907*   4075*
PCMSG   022031          1671    4675#
PIRQ  = 177772          1265#
PIRQVE= 000240          1359#
PLEVEL  020615          1894    4557#
PRG1    001734          1234    1637#
PRG2    006424          1236    2563#
PRG2A   006432          2566#   2636
PRG2B   006552          2608#
PRG3    006632          1238    2647#
PRG3A   006640          2650#   2718
PRG3B   006760          2691#
PRG4    007050          1240    2728#
PRG4A   007056          2731#   2823
PRG4B   007210          2777#
PRG4C   007222          2785#   2826
PRG5    007410          1242    2833#
PRG5A   007416          2836#   2928
PRG5B   007606          2895#   2925
PRG5C   007536          2876#   2931
PRIME   015002          2335    2392    2449    2506    4070#
PROG2M  017460          2564    4449#
PROG3M  017524          2648    4455#
PROG4M  017570          2729    4461#
PROG5M  017634          2834    4467#
PR0   = 000000          1282#
PR1   = 000040          1283#
PR2   = 000100          1284#
PR3   = 000140          1285#
PR4   = 000200          1286#
PR5   = 000240          1287#
PR6   = 000300          1288#
PR7   = 000340          1289#
PS    = 177776          1262#   1263
PSW   = 177776          1263#   2101    3898    3916    3922    4035    4048    4053    4154
PWRVEC= 000024          1354#   1603*   1604*   3850*   3851*   3860*   3866*   3878*   3879*
RDB     017747          2811    2915    4480#
RDCHR = 104410          3559    3841#
RDDEC = 104413          1728    2787    2878    3844#
RDLIN = 104411          3634    3759    3842#
RDOCT = 104412          1739    1758    1781    1813    1839    1885    1895    2569    2610    2618    2653    2693    2701
                        2734    2773    2779    2839    2888    3843#
REDO1   006622          2574    2576    2579    2586    2602    2635#
```

```
REDO2   007040          2658    2660    2663    2670    2685    2717#
REDO3   007370          2739    2741    2744    2751    2766    2822#
REDO3A  007400          2791    2793    2824#
REDO4   007760          2844    2846    2849    2856    2871    2927#
REDO4A  007770          2882    2884    2929#
RESTRT  002036          1691#   2983    3052    3883    4251
RESVEC= 000010          1349#   1694*   1695*
RETRY   002412          1732    1734    1802#
RETRY0  002422          1763    1765    1768    1772    1805#
RETRY1  002432          1786    1788    1791    1795    1808#
RETRY2  003014          1844    1846    1849    1853    1924#
RETRY3  003024          1899    1901    1927#
RFLG0   001424          1573#   2338    2395    2452    2509    3918*   4073*
RFLG1   001426          1574#   2340    2397    2454    2511    3925*   4074*
RINT    014164          3914#   4062
ROTADD  001276          1466#   1830*   1858*   1860    1863*   1870    1871*   2993*   2998    3019*
RSTALL  021364          2769    2887    4621#
RSVERR  015626          1694    4237#
RTRY    001430          1575#   2333*   2381*   2382    2390*   2438*   2439    2447*   2495*   2496    2504*   2549*   2550
SELCAR  021163          2608    2691    2777    4599#
SELDLY  021271          2615    2698    4611#
STACK = 001100          1253#   1595    1691
STAD    001450          1234*   1236*   1238*   1240*   1242*   1583#   1636
STALL   014352          2800    2905    3958#
STCONT  000246          1235    1237    1239    1241    1243#
STKLMT= 177774          1264#
STLMSK  001306          1483#   3970
STMES   017417          1668    1704    4442#
SUERR1  014646          2108    2134    2361    2370    2378    2418    2427    2435    2475    2484    2492    2531    2539
                        2546    4040#   4160
SUERT1  014700          1941    1956    1971    1986    4048#
SUERT2  014726          2597    2680    2761    2866    4053#
SUER2   014620          2000    2010    2021    2027    2049    2059    2070    2076    2123    2148    2154    2160    2164
                        2179    2185    2193    2200    2215    2222    2229    2235    2249    2255    2276    2287    2291
                        2304    2310    2324    4035#
SUVEC   014754          2332    2389    2446    2503    4061#
SWR     001140          1388#   1593    1614*   1616    1622*   1629    1648    1708    2140    2171    2207    2242    2261
                        2297    3073    3087    3089    3095    3102    3140    3147    3152    3156    3403    3442    3497*
                        3858    3871*
SWREG   000176          1233#   1622    1629    3403    3442    3465
SW0     = 000001        1317#   1708
SW00    = 000001        1307#   1317
SW01    = 000002        1306#   1316
SW02    = 000004        1305#   1315
SW03    = 000010        1304#   1314
SW04    = 000020        1303#   1313
SW05    = 000040        1302#   1312
SW06    = 000100        1301#   1311
SW07    = 000200        1300#   1310
SW08    = 000400        1299#   1309
SW09    = 001000        1298#   1308
SW1     = 000002        1316#
SW10    = 002000        1297#
SW11    = 004000        1296#
SW12    = 010000        1295#   2140    2171    2207    2242    2261    2297
SW13    = 020000        1294#
```

```
SW14   = 040000            1293#
SW15   = 100000            1292#
SW2    = 000004            1315#
SW3    = 000010            1314#
SW4    = 000020            1313#
SW5    = 000040            1312#
SW6    = 000100            1311#
SW7    = 000200            1310#
SW8    = 000400            1309#  1648
SW9    = 001000            1308#
TABFLG  001256             1435#  1639*  1689*  1701   1706*  2982*
TBITVE= 000014             1350#
TCONT   014470             3987   3991#
TIMERX  014444             2809   2914   3983#
TIMETX  014462             2801   2906   3989#
TIMR1   001440             1579#  2346*  2403*  2460*  2517*  4079*
TIMR2   001442             1580#  2348*  2405*  2462*  2519*  4080*
TIMR3   001444             1581#  2034*  2046*
TKVEC  = 000060            1357#  3380*  3381*
TPVEC  = 000064            1358#
TRAPVE= 000034             1356#  1601*  1602*
TRPCOM  015652             4233   4240   4244#
TRTVEC= 000014             1351#
TST1    003034             1717   1923   1935#  3014
TST10   003404             2026   2032#
TST11   003544             2053   2058   2064#
TST12   003614             2075   2081#
TST13   004076             2114   2133   2139#
TST14   004226             2141   2163   2170#
TST15   004354             2172   2199   2206#
TST16   004502             2208   2234   2241#
TST17   004562             2243   2254   2260#
TST2    003102             1950#
TST20   004724             2262   2281   2290   2296#
TST21   005006             2298   2309   2317#
TST22   005040             2323   2330#
TST23   005332             2345   2356   2365   2374   2387#
TST24   005624             2402   2413   2422   2431   2444#
TST25   006116             2459   2470   2479   2488   2501#
TST3    003150             1965#
TST4    003216             1980#
TST5    003264             1995#
TST6    003306             1999   2005#
TST7    003332             2009   2015#
TYPDS  = 104405            3041   3836#
TYPE   = 104401            1668   1669   1671   1704   1726   1737   1755   1780   1802   1805   1808   1811   1836
                           1883   1894   1924   1927   2352   2409   2466   2523   2564   2566   2608   2615   2635
                           2648   2650   2691   2698   2717   2729   2731   2769   2777   2785   2803   2811   2822
                           2824   2834   2836   2876   2887   2908   2915   2927   2929   2976   3039   3042   3142
                           3150   3173   3190   3192   3195   3197   3201   3208   3273   3349   3397   3409   3463
                           3464   3467   3478   3488   3499   3518   3566   3572   3577   3581   3586   3587   3589
                           3592   3596   3663   3665   3703   3796   3798   3832#  3880
TYPOC  = 104402            3181   3205   3466   3833#
TYPON  = 104404            3835#
TYPOS  = 104403            3834#
UPMASK  015512             2093   2127   4018   4207#
```

```
VECT    002330          1780#   1810
XDB     017700          2803    2908    4473#
XFLGO   001422          1572#   2336    2393    2450    2507    3900*   4072*
XINT    014076          3896#   4064
XMSG1   017110          2352    4406#
XMSG2   017167          2409    4414#
XMSG3   017250          2466    4423#
XMSG4   017333          2523    4432#
$AUTOB  001134          1385#   1633*   3456    3613
$BDADR  001122          1380#
$BDDAT  001126          1382#
$BELL   001246          1426#   3142    3164    3409    3606
$CHARC  013446          3705*   3715*   3722    3731*   3736#
$CKSWR  012132          3442#   3840
$CMTAG  001100          1368#   1590    1591    1599    1605    1606
$CM1    = 000010        1400#   1401#   1402#   1403#   1404#   1405#   1406#   1407#   1408#
$CM2    = 000020        1400#   1401#   1402#   1403#   1404#   1405#   1406#   1407#   1408#
$CM3    = 000010        1398#   1400
$CM4    = 000020        1408#   1409#   1410#   1411#   1412#   1413#   1414#   1415#   1416#   1417#   1418#   1419#   1420#
                        1421#   1422#   1423#   1424#
$CNTLC  013030          3397    3478    3606#
$CNTLG  013042          3463    3608#
$CNTLU  013035          3488    3581    3607#
$CRLF   001253          1428#   3150    3164    3173    3192    3197    3201    3499    3586    3606    3668    3704    3739
                        3800
$DBLK   011616          3315    3349    3357#
$DOAGN  010362          3035    3044    3050#
$DTBL   011606          3318    3353#
$ENDAD  010352          3046#
$ENDCT  010320          3037#
$ENDMG  010371          3039    3054#
$ENULL  010366          3042    3053#
$EOP    010054          2516    2526    2534    2542    2552    2965#
$EOPCT  010312          3034#   3038
$ERFLG  001103          1371#   3062    3091    3093    3099*   3120    3137*   3164
$ERMAX  001115          1377#   1607*   3093    3115*   3120
$ERROR  010660          1599    3135#
$ERRPC  001116          1378#   3144*   3145*   3146    3164    3179    4277    4299    4321    4364    4377    4389    4402
$ERRTB  001310          1505#   3187
$ERRTY  011020          3149    3172#
$ERTTL  001112          1375#   3143*   3164
$ESCAP  001244          1425#   1606*   1942*   1957*   1972*   1987*   2050*   2109*   2278*   2353*   2362*   2371*   2379*
                        2410*   2419*   2428*   2436*   2467*   2476*   2485*   2493*   2524*   2532*   2540*   2547*   2598*
                        2681*   2762*   2867*   3114*   3159    3161    3164    4161*   4248*
$FILLC  001156          1396#   3708    3739
$FILLS  001155          1395#   3739
$GDADR  001120          1379#
$GDDAT  001124          1381#
$GET42  010342          3043#
$GTSWR  012222          3464#   3838
$HD     = 000001        1211    1212
$HIOCT  013230          3656*   3667#
$ICNT   001104          1372#   3106*   3107    3109*   3119
$ILLUP  014060          3850    3866    3885#
$INTAG  001135          1386#   3461*   3480    3500    3613
$ITEMB  001114          1376#   3146*   3164    3176
```

```
$LF     001254          1429#  3164   3596   3606   3668   3739   3800
$LPADR  001106          1373#  1608*  3097*  3112*  3117   3119
$LPERR  001110          1374#  1609*  3097   3113*  3119   3158
$MAIL = ****** U        1626   1629   3112   3152   3692
$MNEW   013060          3467   3611#
$MSWR   013047          3464   3609#
$MXCNT  010656          3110   3119#
$NULL   001154          1394#  3710   3739
$NWTST= 000001          1932#  1947#  1962#  1977#  1992#  2002#  2012#  2029#  2061#  2078#  2136#  2167#  2203#
                        2238#  2257#  2293#  2314#  2327#  2384#  2441#  2498#
$OCNT   011376          3245*  3274*  3287#
$OMODE  011400          3240*  3244*  3249   3252*  3263*  3289#
$OVER   010642          3074   3090   3098   3108   3116#
$PASS   001100          1369#  3031*  3032*  3040   3053   3104   3120
$POWER  014066          3881   3888#
$PWRAD  014054          3883#
$PWRDN  013714          1603   3850#  3878
$PWRMG  014050          3881#
$PWRUP  013766          3860   3866#
$QUES   001252          1427#  1802   1805   1808   1924   1927   2635   2717   2822   2824   2927   2929   3164
                        3518   3589   3606   3665   3668   3739   3798   3800
$RDCHR  012474          3531#  3841
$RDDEC  013452          3754#  3844
$RDLIN  012564          3554#  3842
$RDOCT  013072          3629#  3843
$RDSZ = 000010          3547#
$REGAD  001160          1398#
$REG0   001162          1400#  4041*  4051*  4246*  4277   4299   4321   4340   4364
$REG1   001164          1401#  4010*  4012*  4042*  4052*  4277   4299   4321   4364   4389   4402
$REG2   001166          1402#  4008*  4010   4043*  4056*  4277   4299   4321   4364   4377   4389   4402
$REG3   001170          1403#  4014*  4044*  4299   4321   4364   4389   4402
$REG4   001172          1404#  4023*  4045*  4299   4321   4402
$REG5   001174          1405#
$REG6   001176          1406#  2104*  3899*  3917*  3923*  4038*  4039*  4054*  4055*  4155*  4244*  4277   4299
                        4321   4340   4364   4377
$REG7   001200          1407#  2165*  4249*  4340
$RTNAD  010364          3052#
$R2A  = ****** U        3845
$SAVRE= ****** U        3845
$SAVR6  014064          3859*  3867   3868*  3869*  3887#
$SCOPE  010406          1597   3071#
$SETUP= 000117          1587#  1596   1597   1599   1601   1603   1605   1606   1608   1626   3029   3072   3136
                        3155   3163   3401   3406   3407   3437   3613
$STUP = 177777          1587#
$SVLAD  010614          3082   3111#
$SWR  = 167400          1198#  1211   1219   1220   1221   1222   1223   1224   1225   1226   1424   1425   1426
                        1605   1606   1608   1609   1936   1951   1966   1981   1996   2006   2016   2033   2065
                        2082   2140   2171   2207   2242   2261   2297   2318   2331   2388   2445   2502   2962
                        3030   3045   3051   3053   3063   3064   3065   3066   3067   3073   3085   3087   3088
                        3091   3092   3093   3100   3101   3102   3113   3116   3119   3127   3128   3129   3130
                        3131   3140   3147   3152   3156   3164   3884
$SWRMK= 000000          1226   1227   3067   3068   3089
$TIMES  001242          1424#  1605*  2331*  2388*  2445*  2502*  3030*  3100*  3107   3110*  3119
$TKB    001146          1391#  3362   3382   3393   3418   3446   3473
$TKCNT  011626          3363#  3377*  3407   3424*  3538   3540*
$TKINT  011644          1635   3377#  3398   3459
```

```
$TKQEN= 011644          3367#  3432   3543
$TKQIN  011630          3364#  3378*  3379   3430*  3431*  3432   3434*
$TKQOU  011632          3365#  3379*  3541   3542*  3543   3545*
$TKQSR  011634          3366#  3378   3434   3545
$TKS    001144          1390#  3362   3383*  3414*  3416   3422*  3444   3460*  3470   3482*  3502*
$TKSRV  011714          3380   3393#
$TMP0   001202          1408#  2101*  2587*  2623*  2671*  2710*  2752*  2796   2807*  2857*  2901   2912*  3898*
                        3916*  3922*  4035*  4048*  4154*  4247*  4277   4299   4321   4340   4364
$TMP1   001204          1409#  2087*  2092*  2116*  2377   2434   2491   2545   2612*  2623   2695*  2710   2781*
                        2807   2899*  2912   3924*  4053*  4207   4377
$TMP10  001222          1416#
$TMP11  001224          1417#  2894*  2895   2898*
$TMP12  001226          1418#  2895*  2896*  2897*  2898   2899
$TMP13  001230          1419#
$TMP14  001232          1420#  2095   2096   2130   4020   4023   4207*  4225*
$TMP15  001234          1421#  1640*  1735*  2794*  2885*  4175   4179   4183   4211   4215   4219
$TMP16  001236          1422#
$TMP17  001240          1423#
$TMP2   001206          1410#  2620*  2621   2703*  2708   2775*  2798   2890*  2903
$TMP3   001210          1411#  2796*  2797*  2805*  2813   2901*  2902*  2910*  2917   3983   3989
$TMP4   001212          1412#  2813*  2815*  2817   2917*  2919*  2921   4008
$TMP5   001214          1413#  2817*  2921*  4006   4014   4020
$TMP6   001216          1414#  3991*  3992*  4006*  4016
$TMP7   001220          1415#  4118*  4120   4123*  4131*  4132   4133*  4134   4135*  4136*
$TN   = 000026          1211#  1932   1936#  1947   1951#  1962   1966#  1977   1981#  1992   1996#  1999   2002
                        2006#  2009   2012   2016#  2026   2029   2033#  2053   2058   2061   2065#  2075   2078
                        2082#  2114   2133   2136   2140#  2141   2163   2167   2171#  2172   2199   2203   2207#
                        2208   2234   2238   2242#  2243   2254   2257   2261#  2262   2281   2290   2293   2297#
                        2298   2309   2314   2318#  2323   2327   2331#  2345   2356   2365   2374   2384   2388#
                        2402   2413   2422   2431   2441   2445#  2459   2470   2479   2488   2498   2502#
$TPB    001152          1393#  3728*  3739
$TPFLG  001157          1397#  3686   3739
$TPS    001150          1392#  3726   3739
$TRAP   013630          1601   3809#
$TRAP2  013652          3820#  3831
$TRP  = 000014          3824#  3833#  3834#  3835#  3836#  3837#  3838   3839#  3840   3841#  3842#  3843#  3844#
                        3845#
$TRPAD  013664          3814   3831#
$TSTNM  001102          1370#  3008*  3029*  3062   3089   3111*  3116   3120   3139   3164   4040   4049   4245
$TTYIN  013020          3556   3557   3569   3587   3601   3605#
$TYPBN= ****** U        3837
$TYPDS  011402          3303#  3836
$TYPE   013232          3686#  3824   3832
$TYPEC  013402          3504   3707   3714   3721   3726#  3727
$TYPEX  013450          3732   3734   3737#
$TYPOC  011200          3243#  3833
$TYPON  011214          3242   3245#  3835
$TYPOS  011154          3238#  3834
$XTSTR  010420          3076#
$$GET4= 000000          3045#
$OFILL  011377          3239*  3243*  3253   3288#
$40CAT= ****** U        3073   3149
.       = 023042        1231#  1245#  1367#  1430   1594   1608   1609   2165   3053   3057   3119   3120   3164
                        3211#  3357#  3362   3366#  3367   3368   3605#  3606   3613#  3668   3739   3800   3862
                        3886   4298#  4376#  4388#  4681#  4684#
```

```
COMMEN     1#  1360#
ENDCOM     1#  1360#
EOPBEG  1214#  2966
ERROR   1254#  1670  1943  1958  1973  1988  2001  2011  2022  2028  2051  2060  2071  2077  2110
        2124   2135  2149  2155  2161  2166  2180  2186  2194  2201  2216  2223  2230  2236  2250
        2256   2279  2288  2292  2305  2311  2325  2354  2363  2372  2380  2411  2420  2429  2437
        2468   2477  2486  2494  2525  2533  2541  2548  2599  2682  2763  2804  2812  2868  2909
        2916   4025  4027  4162  4250
ESCAPE     1#  1360#
GETPRI     1#  1360#
GETSWR     1#  1360#  1626
MORETA  1213#  1431
MULT       1#  1360#
NEWTST     1#  1360#  1932  1947  1962  1977  1992  2002  2012  2029  2061  2078  2136  2167  2203
        2238   2257  2293  2314  2327  2384  2441  2498
POP        1#  1199#  1360#  3344  3657  3789  3871  3872
PUSH       1#  1199#  1360#  3303  3631  3756  3852  3858
REPORT     1#  1360#
SCOPE   1255#  1935  1950  1965  1980  1995  2005  2015  2032  2064  2081  2139  2170  2206  2241
        2260   2296  2317  2330  2387  2444  2501  2970
SETPRI     1#  1360#  3534
SETTRA  3824#  3833  3834  3835  3836  3838  3840  3841  3842  3843  3844
SETUP      1#  1199#  1360#  1589
SKIP       1#  1360#  1940  1955  1970  1985  1999  2009  2020  2026  2052  2058  2069  2075  2114
        2122   2133  2141  2147  2153  2159  2163  2172  2178  2184  2192  2199  2208  2214  2221
        2228   2234  2243  2248  2254  2262  2280  2286  2290  2298  2303  2309  2323  2345  2355
        2364   2373  2402  2412  2421  2430  2459  2469  2478  2487
SLASH      1#  1360#
SPACE   1360#
STARS      1#  1199#  1360#  1363  1430  1563  1565  1932  1934  1947  1949  1962  1964  1977  1979
        1992   1994  2002  2004  2012  2014  2029  2031  2061  2063  2078  2080  2136  2138  2167
        2169   2203  2205  2238  2240  2257  2259  2293  2295  2314  2316  2327  2329  2384  2386
        2441   2443  2498  2500  2959  3028  3059  3123  3167  3215  3293  3361  3437  3452  3523
        3547   3617  3671  3742  3803  3848  3864  3892  3894  3910  3912  4030  4032  4255  4257
SWRSU      1#  1360#  1610#
TRMTRP  3824#
TYPBIN     1#  1360#
TYPDEC     1#  1360#  3040
TYPNAM     1#  1360#
TYPNUM     1#  1360#
TYPOCS     1#  1360#
TYPOCT     1#  1360#  3179  3203  3465
TYPTXT     1#  1360#
$$CMRE  1361#  1400  1401  1402  1403  1404  1405  1406  1407
$$CMTM  1361#  1408  1409  1410  1411  1412  1413  1414  1415  1416  1417  1418  1419  1420  1421
        1422   1423
$$ESCA     1#  1360#
$$NEWT     1#  1360#  1932  1947  1962  1977  1992  2002  2012  2029  2061  2078  2136  2167  2203
        2238   2257  2293  2314  2327  2384  2441  2498
$$SET   3824#  3833  3834  3835  3836  3838  3840  3841  3842  3843  3844
$$SKIP     1#  1360#  1999  2009  2026  2053  2058  2075  2114  2133  2141  2163  2172  2199  2208
        2234   2243  2254  2262  2281  2290  2298  2309  2323  2345  2356  2365  2374  2402  2413
        2422   2431  2459  2470  2479  2488
.EQUAT     1#  1199#  1250
.HEADE     1#  1199#  1201
.KT11      1#
```

```
.SETUP      1#   1199#   1587
.SWRHI      1#   1199#   1275
.SWRLO   1199#   1227#   1228
.$ACT1      1#
.$APTB      1#
.$APTH      1#
.$APTY      1#
.$ASTA      1#
.$CATC      1#   1199#
.$CMTA      1#   1199#   1361
.$DB2D      1#
.$DB2O      1#
.$DIV       1#
.$EOP       1#   1199#   2957
.$ERRO      1#   1199#   3121
.$ERRT      1#   1199#   3165
.$MULT      1#
.$POWE      1#   1199#   3846
.$RAND      1#
.$RDDE      1#   1199#   3740
.$RDOC      1#   1199#   3615
.$READ      1#   1199#   3359
.$R2AZ      1#
.$SAVE      1#
.$SB2D      1#
.$SB2O      1#
.$SCOP      1#   1199#   3057
.$SIZE      1#
.$SUPR      1#
.$TRAP      1#   1199#   3801
.$TYPB      1#
.$TYPD      1#   1199#   3291
.$TYPE      1#   1199#   3669
.$TYPO      1#   1199#   3213
.$40CA      1#
.1170       1#


. ABS.  023042     000


 ERRORS DETECTED:  0

 CZDLCC.BIN,CZDLCC.LST/CRF/SOL/NL:TOC=CZDLCC.SML,CZDLCC.P11
 RUN-TIME: 44 59 3 SECONDS
 RUN-TIME RATIO: 383/107=3.5
 CORE USED:  34K  (67 PAGES)
```