DL11-C,D,E DL11-C,D,E OFLNE TST AH-8525D-MC
CZDLCDO FICHE 1 OF 1

```
                                .REM@
```

IDENTIFICATION

PRODUCT CODE: AC-8524D-MC

PRODUCT NAME: CZDLCD0 DL11-C,D,E OFLNE TST

PRODUCT DATE: MAY 1980

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

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
DSI	RING	CTS	C D	R A	S R			R D	RIEN	DIEN		S X	RTS	DTR		

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15	DATA SET INTERRUPT	1. INTERRUPT SEQUENCE INITIATED WHEN BIT05 SET. 2. SETS WHENEVER BITS 10, 11, 12 OR 14 CHANGE STATE 3. CLEARED BY INIT OR READING RCSR
BIT14	RING	1. WHEN SET, INDICATES A CONTROL SIGNAL BEING RECEIVED FROM DATASET.
BIT13	CLEAR TO SEND	1. WHEN SET INDICATES ON CONDITION; WHEN CLEAR INDICATES OFF CONDITION. 2. DEPENDENT ON STATE OF 'CTS' SIGNAL FROM DATASET
BIT12	CARRIER DETECT	1. SETS WHEN DATA CARRIER

		RECEIVED 2. WHEN CLEAR INDICATES END OF CURRENT TRANSMISSION OR AN ERROR CONDITION.
BIT11	RECEIVER ACTIVE	1. WHEN SET INDICATES RECEIVER INTERFACE IS ACTIVE. 2. CLEARED BY INIT OR RCVR DONE (BIT07).
BIT10	SECONDARY RECEIVE OR SUPERVISORY RECEIVED DATA	1. PROVIDES RECEIVE CAPABILITY, WHEN SET, FOR REVERSE CHANNEL OF REMOTE STATION. SETS WHEN BIT03 IS SET. 2. CLEARED BY INIT
BIT07	RECEIVER DONE	1. SETS WHEN CHARACTER HAS BEEN RECEIVED. WILL INITIATE AN INTERRUPT PROVIDING BIT06 IS ALSO SET 2. CLEARED WHEN RDBR IS ADDRESSED OR BIT00 IS SET. 3. ALSO, CLEARED BY INIT
BIT06	RECEIVER INTERRUPT ENABLE	1. WHEN SET, ALLOWS INTERRUPT PROVIDING BIT07 IS SET. 2. CLEARED BY INIT 3. ***READ/WRITE BIT***
BIT05	DATASET INTERRUPT ENABLE	1. WHEN SET, ALLOWS INTERRUPT PROVIDING BIT15 IS SET. 2. CLEARED BY INIT 3. ***READ/WRITE BIT***
BIT03	SECONDARY TRANSMIT OR SUPERVISORY TRANSMITTED DATA	1. PROVIDES TRANSMIT CAPABILITY. WHEN SET, FOR REVERSE CHANNEL OF REMOTE STATION. SETS WHEN BIT10 IS SET. 2. CLEARED BY INIT 3. ***READ/WRITE BIT***
BIT02	REQUEST TO SEND	1. JUMPER TIES THIS BIT TO REQ TO SEND IN DATASET. 2. REQUIRED FOR TRANSMISSION 3. CLEARED BY INIT 4. ***READ/WRITE BIT***
BIT01	DATA TERMINAL READY	1. WHEN SET, PERMITS CONNECTION TO CHANNEL. 2. WHEN CLEAR, DISCONNECTS INTERFACE FROM CHANNEL.

3. MUST BE CLEARED BY PROGRAM
4. ***READ/WRITE BIT***

SPECIAL NOTES ON RCSR REGISTER

1. ADDRESSES SHOULD FALL IN THE RANGE OF 175610 TO

   176170

2. BIT01 (DATA TERMINAL READY) STATE IS NOT DEFINED AFTER POWER-UP.

3. ON DL11-C OR -D OPTIONS BITS 15, 14, 13, 12, 10, 5, 3, 2, AND 1 ARE NOT USED.

4. ON DL11-C AND -D OPTIONS BIT<00> IS 'RDR ENB' . ON A DL11-E OPTION THIS BIT IS UNUSED.

B. THE FOLLOWING IS A PICTURE VIEW OF A DL11-E TRANSMITTER CONTROL STATUS REGISTER, WHICH WILL SHOW BIT ASSIGNMENTS AND DEFINITIONS, TO PROVIDE A HANDY REFERENCE:

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

BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT07	TRANSMITTER READY	1. SET WHEN XDBR CAN ACCEPT ANOTHER CHARACTER. WILL INITIATE AN INTERRUPT IF BIT06 ALSO SET. 2. ALSO SET BY INIT 3. CLEARED BY LOADING XDBR
BIT06	TRANSMITTER INTERRUPT ENABLE	1. WHEN SET, ALLOWS INTERRUPT PROVIDING BIT07 IS SET. 2. CLEARED BY INIT 3. ***READ/WRITE BIT***
BIT02	MAINTENANCE	1. WHEN SET, DISABLES SERIAL LINE INPUT TO RECEIVER & CONNECTS XMIT OUTPUT TO RECEIVER INPUT WHICH DISCONNECTS EXTERNAL DEVICE INPUT. THIS FORCES RECEIVER TO RUN AT XMITTER SPEED. 2. CLEARED BY INIT 3. ***READ/WRITE BIT***

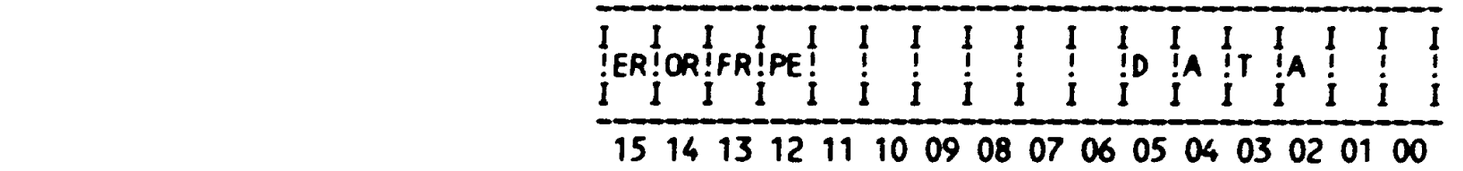
BIT00 BREAK

1. WHEN SET, TRANSMITS A CONTINUOUS SPACE TO EXTERNAL DEVICE
2. CLEARED BY INIT
3. ***READ/WRITE BIT***

!! NOTE !!

DL11-C AND -D OPTIONS ARE THE SAME.

C. THE FOLLOWING IS A PICTURE VIEW OF THE DL11-E RECEIVER AND TRANSMITTER DATA BUFFER REGISTERS, TO PROVIDE A HANDY REFERENCE.



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 PELOW:

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.

                                CHGD1 - ADDED CLEAR PROC STATUS WORD UPON ENTRY INTO TEST
                                        TO CLEAR UP INTERRUPT TEST FAILURES BEING CAUSED
                                        BY THE CPU PRIORITY BEING SET TOO HIGH.
                                CHGD2 - ADDED .=42 WITH A ZERO VALUE TO PREVENT THAT
                                        LOCATION FROM INADVERTANTLY CAUSING THE PROGRAM
                                        TO RETURN TO SOME FALSE LOCATION RATHER THAN ITS
                                        INTENDED PURPOSE OF RETURNING PROGRAM CONTROL TO
                                        THE MONITOR.

                               .ENDR 0


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

                               .TITLE  CZDLCDO DL11-C,D,E OFLNE TST
                               ;*COPYRIGHT (C) 1980
                               ;*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
                               ;*
```

```
1207                                    ;*      SWITCH          USE
1208                                    ;*      ------          ------------------
1209                                    ;*        15            HALT ON ERROR
1210                                    ;*        14            LOOP ON TEST
1211                                    ;*        13            INHIBIT ERROR TYPEOUTS
1212                                    ;*        12            /C OR /D MODEL
1213                                    ;*        11            INHIBIT ITERATIONS
1214                                    ;*        10            BELL ON ERROR
1215                                    ;*         9            LOOP ON ERROR
1216                                    ;*         8            LOOP ON TEST IN SWR<7:0>
1217
1218                                    ;*         0            CREATION OF DEVICE/S TABLE
1219                                    ;*                      OR CHANGE CHARACTER LENGTH
1220
1221            000174                  .=174
1222  000174    000000                  DISPREG:  .WORD 0       ;SOFTWARE DISPLAY REGISTER
1223  000176    000000                  SWREG:    .WORD         ;SOFTWARE SWITCH REGISTER
1224  000200    012767    001734  001242        MOV     #PRG1,STAD      ;ADDRESS OF USER PROGRAM NO. 1
1225  000206    000417                  BR      STCONT
1226  000210    012767    006424  001232        MOV     #PRG2,STAD      ;ADDRESS OF USER PROGRAM NO.2
1227  000216    000413                  BR      STCONT
1228  000220    012767    006632  001222        MOV     #PRG3,STAD      ;ADDRESS OF USER PROGRAM NO. 3
1229  000226    000407                  BR      STCONT
1230  000230    012767    007050  001212        MOV     #PRG4,STAD      ;ADDRESS OF USER PROGRAM NO. 4
1231  000236    000403                  BR      STCONT
1232  000240    012767    007410  001202        MOV     #PRG5,STAD      ;ADDRESS OF USER PROGRAM NO. 5
1233  000246                          CHGD1:
1234  000246    005067    177524        STCONT: CLR     PS              ;CLEAR PSW
1235  000252    000137    001452                JMP     @#BEGIN         ;JUMP TO COMMON START
1236
1237  000256    000042                  CHGD2:  .=42            ;STORAGE LOC FOR MONITOR ADDRESS
1238  000042    000000                          .WORD   0       ;CLEAR LOC FOR START UP
1239            000052                          .=52            ;INFORMATION LOCATION FOR ACT11
1240  000052    000000                          .WORD   0       ;NO POWER FAIL REQUIRED <BIT15=0>
1241                                                            ;IS NOT MEMORY SIZE DEPENDENT <BIT14=0>
1242                                                            ;IS SUITABLE FOR AUTOMATIC OPERATION <BIT13=0>
1243
1244                                    .SBTTL  BASIC DEFINITIONS
1245
1246                                    ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
1247            001100                  STACK=  1100
1248                                    .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
1249                                    .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL
1250
1251                                    ;*MISCELLANEOUS DEFINITIONS
1252            000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
1253            000012                  LF=     12              ;;CODE FOR LINE FEED
1254            000015                  CR=     15              ;;CODE FOR CARRIAGE RETURN
1255            000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
1256            177776                  PS=     177776          ;;PROCESSOR STATUS WORD
1257                                    .EQUIV  PS,PSW
1258            177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
1259            177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
1260            177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
1261            177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
1262
```

```
1263                            ;*GENERAL PURPOSE REGISTER DEFINITIONS
1264    000000                  R0=     %0              ;;GENERAL REGISTER
1265    000001                  R1=     %1              ;;GENERAL REGISTER
1266    000002                  R2=     %2              ;;GENERAL REGISTER
1267    000003                  R3=     %3              ;;GENERAL REGISTER
1268    000004                  R4=     %4              ;;GENERAL REGISTER
1269    000005                  R5=     %5              ;;GENERAL REGISTER
1270    000006                  R6=     %6              ;;GENERAL REGISTER
1271    000007                  R7=     %7              ;;GENERAL REGISTER
1272    000006                  SP=     %6              ;;STACK POINTER
1273    000007                  PC=     %7              ;;PROGRAM COUNTER
1274
1275                            ;*PRIORITY LEVEL DEFINITIONS
1276    000000                  PR0=    0               ;;PRIORITY LEVEL 0
1277    000040                  PR1=    40              ;;PRIORITY LEVEL 1
1278    000100                  PR2=    100             ;;PRIORITY LEVEL 2
1279    000140                  PR3=    140             ;;PRIORITY LEVEL 3
1280    000200                  PR4=    200             ;;PRIORITY LEVEL 4
1281    000240                  PR5=    240             ;;PRIORITY LEVEL 5
1282    000300                  PR6=    300             ;;PRIORITY LEVEL 6
1283    000340                  PR7=    340             ;;PRIORITY LEVEL 7
1284
1285                            ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
1286    100000                  SW15=   100000
1287    040000                  SW14=   40000
1288    020000                  SW13=   20000
1289    010000                  SW12=   10000
1290    004000                  SW11=   4000
1291    002000                  SW10=   2000
1292    001000                  SW09=   1000
1293    000400                  SW08=   400
1294    000200                  SW07=   200
1295    000100                  SW06=   100
1296    000040                  SW05=   40
1297    000020                  SW04=   20
1298    000010                  SW03=   10
1299    000004                  SW02=   4
1300    000002                  SW01=   2
1301    000001                  SW00=   1
1302                            .EQUIV  SW09,SW9
1303                            .EQUIV  SW08,SW8
1304                            .EQUIV  SW07,SW7
1305                            .EQUIV  SW06,SW6
1306                            .EQUIV  SW05,SW5
1307                            .EQUIV  SW04,SW4
1308                            .EQUIV  SW03,SW3
1309                            .EQUIV  SW02,SW2
1310                            .EQUIV  SW01,SW1
1311                            .EQUIV  SW00,SW0
1312
1313                            ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
1314    100000                  BIT15=  100000
1315    040000                  BIT14=  40000
1316    020000                  BIT13=  20000
1317    010000                  BIT12=  10000
1318    004000                  BIT11=  4000
```

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

                                        ;*BASIC "CPU" TRAP VECTOR ADDRESSES
                        000004          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
                        000010          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
                        000014          TBITVEC=14              ;;"T" BIT
                        000014          TRTVEC= 14              ;;TRACE TRAP
                        000014          BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
                        000020          IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
                        000024          PWRVEC= 24              ;;POWER FAIL
                        00003C          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
```

```
1411  001224  000000                  $TMP11: .WORD   0               ;;USER DEFINED
1412  001226  000000                  $TMP12: .WORD   0               ;;USER DEFINED
1413  001230  000000                  $TMP13: .WORD   0               ;;USER DEFINED
1414  001232  000000                  $TMP14: .WORD   0               ;;USER DEFINED
1415  001234  000000                  $TMP15: .WORD   0               ;;USER DEFINED
1416  001236  000000                  $TMP16: .WORD   0               ;;USER DEFINED
1417  001240  000000                  $TMP17: .WORD   0               ;;USER DEFINED
1418  001242  000000                  $TIMES  0                       ;;MAX. NUMBER OF ITERATIONS
1419  001244  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
1420  001246  177607  000377          $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
1421  001252     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
1422  001253     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
1423  001254  000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
1424                                  ;;**************************************************************************
1425
1426                                  ;THE FOLLOWING TAG(S) ARE USER SUPPLIED BY CALLING THE MACRO
1427                                  ;'MORETAGS' AS ONE OF THE ARGUMENTS TO THE SYSMAC ROUTINE .SCMTAG
1428
1429  001256  000000                  TABFLG: .WORD   0               ;AN INDICATOR TO SHOW THAT THE
1430                                                                  ;INFORMATION FOR MULTIPLE DEVICE
1431                                                                  ;TESTING HAS ALREADY TRANSPIRED
1432                                                                  ;& 'MAINDEC' NAME HAS BEEN PRINTED
1433  001260  000000                  DLBASE: .WORD   0               ;STORAGE & WORKING LOCATION FOR A DEVICE
1434                                                                  ;RECEIVER STATUS REGISTER ADDRESS
1435  001262  000000                  KEEPAD: .WORD   0               ;STORAGE LOCATION FOR THE 1ST
1436                                                                  ;DEVICE RCSR FROM WHICH
1437                                                                  ;'BASEADD' IS RESTORED AT THE
1438                                                                  ;END OF A COMPLETE PROGRAM PASS.
1439  001264  000000                  BASEADD:.WORD   0               ;STORAGE LOCATION WHICH HOLDS
1440                                                                  ;THE RCSR ADDRESS OF THE 'NEXT'
1441                                                                  ;DEVICE DURING MULTIPLE TESTING
1442  001266  000000                  KEEPIV: .WORD   0               ;STORAGE LOCATION FOR THE 1ST
1443                                                                  ;DEVICE RECEIVER VECTOR FROM
1444                                                                  ;WHICH 'BASEIV' IS RESTORED AT THE
1445                                                                  ;END OF A COMPLETE PROGRAM PASS
1446  001270  000000                  BASEIV: .WORD   0               ;STORAGE LOCATION WHICH HOLDS
1447                                                                  ;THE VECTOR ADDRESS OF THE 'NEXT'
1448                                                                  ;DEVICE DURING MULTIPLE TESTING
1449  001272  000000                  MULTD:  .WORD   0               ;FLAG TO INDICATE TO 'END OF PASS'
1450                                                                  ;ROUTINE THAT MULTIPLE DEVICE
1451                                                                  ;TESTING IS BEING CONDUCTED
1452                                                                  ;0=NO, 1=YES
1453  001274  000000                  ACTREG: .WORD   0               ;THIS IS THE DEVICE ACTIVE REGISTER
1454                                                                  ;A BIT IS SET (STARTING AT
1455                                                                  ;BIT0)FOR EACH CONTIGUOUS DEVICE
1456                                                                  ;(A MAX. OF 16) THAT IS TO UNDERGO
1457                                                                  ;TESTING.  THIS LOCATION IS
1458                                                                  ;AUTOMATICALLY FILLED BASED ON
1459                                                                  ;USER RESPONSE TO PROGRAM QUESTIONS
1460  001276  000000                  ROTADD: .WORD   0               ;A ROTATING POINTER TO SIGNAL
1461                                                                  ;THE LAST DEVICE TESTED (IF
1462                                                                  ;MULTIPLE DEVICE TESTING WAS BEING
1463                                                                  ;DONE) IF LESS THAN A FULL COMPLE-
1464                                                                  ;MENT OF DEVICES (16)WAS SELECTED
1465  001300  000000                  LASTADD:.WORD   0               ;STORAGE LOCATION FOR THE
1466                                                                  ;RCSR ADDRESS OF THE LAST DEVICE
```

```
1467
1468
1469  001302  000000                DLPRI:  .WORD   0          ;TESTED (IF MULTIPLE DEVICE
                                                               ;TESTING WAS SELECTED BY USER)
                                                               ;STORAGE LOCATION FOR THE DEVICE
1470                                                           ;INTERRUPT PRIORITY LEVEL
1471  001304  000000                LESS1:  .WORD   0          ;THE PRIORITY LEVEL THE CPU
1472                                                           ;MUST BE AT TO ALLOW DEVICE INTERRUPTS.
1473                                                           ;THIS WILL BE 1 LEVEL LESS THAN
1474                                                           ;THE DEVICE LEVEL (BASED ON &
1475                                                           ;CALCULATED FROM USER RESPONSE TO
1476                                                           ;DEVICE PRIORITY LEVEL QUESTION)
1477  001306  177740                STLMSK: 177740             ;THIS MASK IS USED BY THE 'STALL'
1478                                                           ;ROUTINE WHICH WAITS A RANDOM NO.
1479                                                           ;OF MILLISECONDS.  ITS' USE PREVENTS
1480                                                           ;A STALL > 37 MSEC.  THIS LOCATION
1481                                                           ;HOWEVER, CAN BE PATCHED BY THE
1482                                                           ;USER TO ALLOW LARGER 'STALLS'.
1483
1484                          ;END OF USER SUPPLIED TAG(S)
```

```
1485                                    .SBTTL  ERROR POINTER TABLE
1486
1487                                    ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
1488                                    ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
1489                                    ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
1490                                    ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
1491                                    ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
1492
1493                                    ;*      EM              ;;POINTS TO THE ERROR MESSAGE
1494                                    ;*      DH              ;;POINTS TO THE DATA HEADER
1495                                    ;*      DT              ;;POINTS TO THE DATA
1496                                    ;*      DF              ;;POINTS TO THE DATA FORMAT
1497
1498
1499  001310                            $ERRTB:
1500
1501                                    ;ERROR TABLE ITEM FOR ERROR MESSAGE 1
1502
1503  001310  015714                            EM1             ;'DL11 REGISTER REFERENCE CAUSED TIMEOUT''
1504  001312  015763                            DH1             ;''  (PC)  (PS)  (SP)  TEST  DEVADR  REGADR  ''
1505  001314  016042                            DT1             ; (R7)  (PSW)  (R6)  (R0)  (R1)  (R2)
1506  001316  000000                            0               ;PRINT ALL OCTAL
1507
1508                                    ;ERROR TABLE ITEM FOR ERROR MESSAGE 2
1509
1510  001320  016060                            EM2             ;'' DL11 REGISTER ERROR ''
1511  001322  016104                            DH2             ;''  (PC)  (PS)  (SP)  TEST  DEVADR  REGADR  WAS  S/B  ''
1512  001324  016202                            DT2             ;'' (R7)  (PSW)  (R6)  (R0)  (R1)  (R2)  (R3)  (R4)  ''
1513  001326  000000                            0               ;PRINT ALL OCTAL
1514
1515                                    ;ERROR TABLE ITEM FOR ERROR MESSAGE 3
1516
1517  001330  016224                            EM3             ;'' DL11 DATA COMPARE ERROR ''
1518  001332  016254                            DH3             ;'' (PC)  (PS)  (SP)  TEST  WASADR  SHBADR  WAS  S/B  ''
1519  001334  016352                            DT3             ;''  (R7)  (PSW)  (R6)  (R0)  (R1)  (R2)  (R3)  (R4)  ''
1520  001336  000000                            0               ;PRINT ALL OCTAL
1521
1522                                    ;ERROR TABLE ITEM FOR ERROR MESSAGE 4
1523
1524  001340  016374                            EM4             ;'' UNEXPECTED TRAP TO VECTOR AT LOCATION XXX ''
1525  001342  016446                            DH4             ;''  (PC)  (PS)  (SP)  TEST  ''
1526  001344  016504                            DT4             ;'' (R7)  (PSW)  (R6)  (R0)  ''
1527  001346  000000                            0               ;PRINT ALL OCTAL
1528
1529                                    ;ERROR TABLE ITEM FOR ERROR MESSAGE 5
1530
1531  001350  016516                            EM5             ;'' DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN)  ''
1532  001352  016573                            DH5             ;'' (PC)  (PS)  (SP)  TEST  DEVADR  REGADR  (REG)  ''
1533  001354  016662                            DT5             ;''  (R7)  (PSW)  (R6)  (R0)  (R1)  (R2)  (R3)  ''
1534  001356  000000                            0
1535
1536                                    ;ERROR TABLE ITEM FOR ERROR MESSAGE 6
1537
1538  001360  015714                            EM1             ;'DL11 REGISTER REFERENCE CAUSED TIMEOUT''
1539  001362  016702                            DH6             ;''  (PC)     (PS)     (SP)     REGADR''
1540  001364  016742                            DT6             ;$ERRPC,$TMP0,$REG6,$REG2
```

```
 1541  001366  000000                    0                ;PRINT ALL OCTAL
 1542
 1543                                ;ERROR TABLE ITEM FOR ERROR MESSAGE 7
 1544
 1545  001370  016516                    EM5              ;'' DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN) ''
 1546  001372  016754                    DH7              ;'' (PC)  DEVADR  REGADR  (REG)''
 1547  001374  017014                    DT7              ;$ERRPC,$REG1,$REG2,$REG3
 1548  001376  000000                    0                ;PRINT ALL OCTAL
 1549
 1550                                ;ERROR TABLE ITEM FOR ERROR MESSAGE 10
 1551
 1552  001400  016224                    EM3              ;'' DL11 DATA COMPARE ERROR ''
 1553  001402  017026                    DH10             ;'' (PC)  DEVADR  REGADR  (REG)  S/B''
 1554  001404  017074                    DT10             ;$ERRPC,$REG1,$REG2,$REG3,$REG4
 1555  001406  000000                    0                ;PRINT ALL OCTAL
 1556
 1557                                ;;*****************************************************************
 1558                                    ;DL11 DEFINITIONS
 1559                                ;;*****************************************************************
 1560
 1561  001410  175610                    DLRCSR: 175610          ;CONTAINS ADDRESS OF RCVR CSR
 1562  001412  175612                    DLRDBR: 175612          ;CONTAINS ADDRESS OF RCVR DBR
 1563  001414  175614                    DLXCSR: 175614          ;CONTAINS ADDRESS OF XMIT CSR
 1564  001416  175616                    DLXDBR: 175616          ;CONTAINS ADDRESS OF XMIT DBR
 1565  001420  000300                    DLVECT: 300             ;CONTAINS VECTOR ADDRESS OF CURRENT DL11
 1566  001422  000000                    XFLG0:  0               ;FLAG FOR HARD XMIT ERRORS
 1567  001424  000000                    RFLG0:  0               ;FLAG FOR HARD RCVR ERRORS
 1568  001426  000000                    RFLG1:  0               ;FLAG FOR SOFT RCVR ERRORS
 1569  001430  000000                    RTRY:   0               ;COUNTS NO. OF RETRIES ON SOFT ERRORS
 1570  001432  000000                    OPTR:   0               ;CONTAINS POINTER TO OUTPUT BUFFER
 1571  001434  000000                    IPTR:   0               ;CONTAINS POINTER TO INPUT BUFFER
 1572  001436  000000                    LDOUT:  0               ;CONTAINS POINTER TO LOAD BUFFER ROUTINE
 1573  001440  000000                    TIMR1:  0               ;TIMERS FOR 256. BYTE BLOCK TRANSFERS
 1574  001442  000000                    TIMR2:  0
 1575  001444  000000                    TIMR3:  0               ;DELAY TIMER FOR TEST 10
 1576  001446  000000                    INTFLG: 0               ;SOFTWARE INTR. FLAG
 1577  001450  000000                    STAD:   0               ;TEMPORARY ADDRESS STORAGE LOC.
 1578
 1579
 1580
 1581
 1582  001452  000240          BEGIN:    NOP                     ;PROGRAM WILL START HERE
 1583                          .SBTTL    INITIALIZE THE COMMON TAGS
 1584                          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
 1585  001454  012706  001100            MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
 1586  001460  005026                    CLR     (R6)+           ;;CLEAR MEMORY LOCATION
 1587  001462  022706  001140            CMP     #SWR,R6 ;;DONE?
 1588  001466  001374                    BNE     .-6             ;;LOOP BACK IF NO
 1589  001470  012706  001100            MOV     #STACK,SP       ;;SETUP THE STACK POINTER
 1590                          ;;INITIALIZE A FEW VECTORS
 1591  001474  012737  010406  000020    MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
 1592  001502  012737  000340  000022    MOV     #340,@#IOTVEC+2 ;;LEVEL 7
 1593  001510  012737  010660  000030    MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
 1594  001516  012737  000340  000032    MOV     #340,@#EMTVEC+2 ;;LEVEL 7
 1595  001524  012737  013630  000034    MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
 1596  001532  012737  000340  000036    MOV     #340,@#TRAPVEC+2;LEVEL 7
```

```
1597  001540  012737  013714  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
1598  001546  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
1599  001554  005067  177462                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
1600  001560  005067  177460                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1601  001564  112767  000001  177323          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
1602  001572  012767  001572  177306          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1603  001600  012767  001600  177302          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
1604                                  ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1605                                  ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
1606  001606  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
1607  001612  012737  001646  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
1608  001620  012767  177570  177312          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
1609  001626  012767  177570  177306          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
1610  001634  022777  177777  177276          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
1611  001642  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1612                                                                  ;;AND  THE HARDWARE SWR IS NOT = -1
1613  001644  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
1614  001646  012716  001654          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
1615  001652  000002                          RTI
1616  001654  012767  000176  177256  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
1617  001662  012767  000174  177252          MOV     #DISPREG,DISPLAY
1618  001670  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
1619
1620                                  .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1621  001674  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
1622  001700  001006                          BNE     67$             ;;BRANCH IF YES
1623  001702  026727  177232  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
1624  001710  001005                          BNE     68$             ;;BRANCH IF NO
1625  001712  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
1626  001714  000403                          BR      68$
1627  001716  112767  000001  177210  67$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
1628  001724                          68$:
1629  001724  004767  007714                  JSR     PC,$TKINT       ;SET TTY INTERRUPT
1630  001730  000177  177514                  JMP     @STAD           ;JUMP TO SELECTED TEST
1631  001734  005067  177332          PRG1:   CLR     MULTD           ;CLEAR MULTIPLE DEVICE
1632                                                                  ;TESTING FLAG
1633  001740  005067  177312                  CLR     TABFLG          ;CLEAR TABLE CREATION FLAG
1634  001744  012767  000010  177262          MOV     #8.,$TMP15      ;SET CHARACTER LENGTH DESIGNATOR
1635                                                                  ;FOR 8 BITS -- THIS IS THE DEFAULT
1636                                                                  ;LENGTH ASSUMED BY THE PROGRAM
1637                                                                  ;UNLESS THE USER CHANGES IT THRU
1638                                                                  ;THE QUESTION AND ANSWER CYCLE
1639                                                                  ;INITIATED BY SETTING SW<0> TO A 1
1640  001752  012767  000200  177322          MOV     #200,DLPRI      ;SET STANDARD PRIORITY LEVEL
1641                                                                  ;FOR DEVICE
1642  001760  032777  000400  177152          BIT #SW8,@SWR           ;IS THE 'LOOP ON TEST' SWITCH SET?
1643  001766  001411                          BEQ     1$              ;BRANCH IF NOT
1644                                  ;
1645                                  ;IF THE 'LOOP ON TEST' SWITCH WAS SET WE WILL TAKE THE NEXT BRANCH
1646                                  ;INSTRUCTION THUS BYPASSING TABLE CREATION
1647                                  ;
1648                                  ;IF THE USER DESIRED TO LOOP ON A TEST OF OTHER THAN THE DEFAULT DEVICE
1649                                  ;THEN HE SHOULD HAVE PREVIOUSLY FILLED THE FOLLOWING PROGRAM LOCATIONS
1650                                  ;WITH THE DESIRED DEVICE REGISTER VALUES:
1651                                  ;
1652                                  ;                       ********************************
```

```
1653                                    ;               UNDER   ;DL11 DEFINITIONS                ABOVE
1654                                    ;                       **********************************
1655                                    ;
1656                                    ;       DLRCSR:         PATCH THE ADDRESS OF THE RCVR CSR
1657                                    ;       DLRDBR:         PATCH THE ADDRESS OF THE RCVR DBR
1658                                    ;       DLXCSR:         PATCH THE ADDRESS OF THE XMIT CSR
1659                                    ;       DLXDBR:         PATCH THE ADDRESS OF THE XMIT DBR
1660                                    ;       DLVECT:         PATCH THE VECTOR ADDRESS OF THIS DL11
1661                                    ;
1662  001770  104401  017417                       TYPE,   STMES           ;PRINT OUT 'MAINDEC' NAME
1663  001774  104401  021453                       TYPE,   FAILSA          ;TYPE FAILSAFE MESSAGE
1664  002000  104000                               ERROR   +0              ;TYPE OUT THE PC VALUE
1665  002002  104401  022031                       TYPE,   PCMSG           ;FOLLOWED BY =PC
1666  002006  000000                               HALT                    ;WAIT FOR USER TO RESPOND
1667  002010  000443                               BR      ONCE            ;GO TO TEST DEVICE PATCHED IN BY USER
1668  002012                                1$:
1669                                        ;ENSURE THAT IF MULTIPLE DEVICE TESTING WAS BEING DONE
1670                                        ;AND THE USER 'HALTED' THE PROGRAM BEFORE ALL DEVICES
1671                                        ;WERE COMPLETED AND WENT BACK TO 'LOAD ADDRESS 200'
1672                                        ;TO RESTART THE PROGRAM THAT AS A BARE MINIMUM
1673                                        ;HE CAN RUN THE DEFAULT DEVICE (1ST RECEIVER
1674                                        ;STATUS REGISTER ADDRESS 175610)
1675                                        ;NOTE:  IF THIS IS NOT SUITABLE THE USER WILL
1676                                        ;       HAVE TO SET SW0=1 (OR UP) IN ORDER TO
1677                                        ;       RECREATE THE TABLE HE DESTROYED FROM
1678                                        ;       ABOVE
1679  002012  012767  175610  177240                MOV     #175610,DLBASE  ;1ST POSSIBLE RECEIVER CSR
1680  002020  004767  005754                       JSR     PC,DLADDR       ;FORM DL ADDRESSES FOR
1681                                                                        ;1ST POSSIBLE DEVICE
1682  002024  012767  000300  177366                MOV     #300,DLVECT     ;1ST POSSIBLE INTERRUPT VECTOR
1683  002032  005067  177220                       CLR     TABFLG          ;CLEAR TABLE CREATION FLAG
1684
1685  002036  012706  001100        RESTRT: MOV     #STACK,SP                       ;SET UP STACK POINTER
1686  002042  012737  015602  000004        MOV     #BUSERR,@#ERRVEC                ;SET UP BUS ERROR VECTOR
1687  002050  012737  000340  000006        MOV     #340,@#ERRVEC+2
1688  002056  012737  015626  000010        MOV     #RSVERR,@#RESVEC                ;SET UP RSVD INSTR. VECTOR
1689  002064  012737  000340  000012        MOV     #340,@#RESVEC+2
1690
1691                                        ;THIS  NEXT SECTION WILL CHECK TO SEE IF MULTIPLE DEVICE TESTING
1692                                        ;WILL TAKE PLACE I.E.-
1693                                        ;       A)  HAS FREE RUNNING DEVICE TABLE ALREADY BEEN CREATED, AND/OR
1694                                        ;       B)  IF IT HAS, DOES USER WISH TO CHANGE IT, OR DO WE TEST DEFAULT DEVICE?
1695  002072  105767  177160                TSTB    TABFLG          ;HAS TABLE CREATION BEEN PERFORMED?
1696  002076  001010                       BNE     ONCE            ;BRANCH IF YES TO SKIP 'MAINDEC
1697                                                                ;TITLE' MESSAGE
1698  002100  104401  017417                TYPE    ,STMES          ;OTHERWISE, PRINT OUT 'MAINDEC'
1699                                                                ;NAME
1700  002104  105167  177146                COMB    TABFLG          ;IF TABLE CREATION HAS NOT BEEN
1701                                                                ;PERFORMED, THEN SET FLAG, AND DO SO
1702  002110  032777  000001  177022        BIT     #SW0,@SWR       ;THE PROGRAM HAS OBVIOUSLY BEEN
1703                                                                ;RESTARTED - DOES USER WISH TO
1704                                                                ;RESELECT VECTOR AND CONTROL REGISTER
1705                                                                ;ADDRESSES I.E. - CREATE A NEW TABLE?
1706  002116  001012                       BNE     GO              ;BRANCH IF YES
1707  002120  005077  177270        ONCE:   CLR     @DLXCSR         ;CLEAR OUT BOTH CSR'S
1708  002124  005077  177260                CLR     @DLRCSR
```

```
1709  002130  005777  177256          TST     @DLRDBR         ;FLUSH RCVR 'DONE'' BIT
1710  002134  005777  177252          TST     @DLRDBR
1711  002140  000167  000670          JMP     TST1            ;OTHERWISE, GO WITH EXISTING
1712                                                          ;TABLE OR NOT USE ANY TABLE AT
1713                                                          ;ALL WHICHEVER THE CASE MAY BE
1714                                                          ;(DEFAULT CASE IS 1ST POSSIBLE
1715                                                          ;DEVICE)
1716                                  ;IF WE COME THIS PATH THE USER HAS DECIDED 1 OF 2 ALTERNATIVES:
1717                                  ;       A)  TO RUN MULTIPLE DEVICES
1718                                  ;       B)  TO CREATE A NEW TABLE TO RUN FROM, OR
1719                                  ;       C)  TO CHANGE THE CHARACTER LENGTH
1720  002144  104401  020013          GO:     TYPE,   LENGTH          ;ASK USER FOR THE CHARACTER LENGTH
1721                                                          ;FOR WHICH HIS DEVICE IS SET
1722  002150  104413                  RDDEC                   ;ACCEPT THE ANSWER TYPED BY USER
1723                                  ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1724  002152  012600                  MOV     (SP)+,R0        ;GET THE ANSWER TYPED
1725  002154  020027  000010          CMP     R0,#8.          ;IS THE NUMBER TOO HIGH?
1726  002160  101114                  BHI     RETRY           ;IF YES - GO TO RETRY SITUATION
1727  002162  020027  000005          CMP     R0,#5.          ;IS THE NUMBER TOO LOW?
1728  002166  103511                  BLO     RETRY           ;IF YES - GO TO RETRY SITUATION
1729  002170  010067  177040          MOV     R0,$TMP15       ;THE VALUE TYPED IS OK
1730                                                          ;STORE FOR FUTURE USE
1731  002174  104401  020075          TYPE,   DEFAULT         ;ASK USER IF HE WISHES TO TEST OTHER
1732                                                          ;THAN THE DEFAULT DEVICE
1733  002200  104412                  RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
1734  002202  005726                  TST     (SP)+           ;LOOK AT THE ANSWER
1735  002204  001002                  BNE     1$              ;BRANCH IF REPLY WAS YES
1736  002206  000137  002770          JMP     @#FLUSH         ;OTHERWISE, SKIP REST OF INTERROGATION
1737  002212  012700  000300  1$:     MOV     #300,R0         ;START RESTORATION OF TRAPCATCHER
1738  002216  012701  000302          MOV     #302,R1         ;AREA FROM LOCATIONS 300 TO 776
1739  002222  012702  000004          MOV     #4,R2           ;SO THAT WE CREATE THE MULTIPLE
1740  002226  010110  2$:     MOV     R1,(R0)         ;DEVICES TABLE WITH A CLEAN SLATE
1741  002230  005011                  CLR     (R1)            ;       .
1742  002232  060200                  ADD     R2,R0           ;       .
1743  002234  060201                  ADD     R2,R1           ;       .
1744  002236  022701  001000          CMP     #1000,R1        ;       .
1745  002242  002771                  BLT     2$              ;       .
1746                                  ;THE TRAPCATCHER VECTOR AREA FROM 300 - 776 SHOULD NOW BE RESTORED.
1747                                  ;PROCEED TO FIND OUT THE 1ST DEVICE RECEIVER CONTROL REGISTER
1748                                  ;ADDRESS
1749  002244  104401  020202  FIRSTD: TYPE    ,#FIRSTD        ;ASK USER FOR THE RECEIVER CONTROL
1750                                                          ;REGISTER ADDRESS OF HIS FIRST
1751                                                          ;DEVICE
1752  002250  104412                  RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
1753                                                          ;AND STORE ON TOP OF STACK
1754                                  ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1755  002252  012600                  MOV     (SP)+,R0        ;GET THE ANSWER TYPED
1756  002254  020027  176170          CMP     R0,#176170      ;IS THE NUMBER TOO HIGH?
1757  002260  101060                  BHI     RETRY0          ;IF YES-GO TO RETRY SITUATION
1758  002262  020027  175610          CMP     R0,#175610      ;IS THE NUMBER TOO LOW?
1759  002266  103455                  BLO     RETRY0          ;IF YES - GO TO RETRY SITUATION
1760  002270  132700  000001          BITB    #BIT0,R0        ;NUMBER IS IN RANGE BUT IS IT
1761                                                          ;ON AN EVEN BOUNDARY?
1762  002274  001052                  BNE     RETRY0          ;IF NO - GO TO RETRY SITUATION
1763                                  ;CHECK TO SEE IF USER RESPONSE WAS TRULY A RCVR STATUS REGISTER
1764  002276  032700  000007          BIT #7,R0               ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
```

```
1765
1766  002302  001047                          BNE     RETRY0          ;USER RESPONSE EQUAL TO A ZERO?
1767  002304  010067  176750                  MOV     R0,DLBASE       ;BRANCH IF NOT
1768                                                                    ;THE 1ST ADDRESS VALUE TYPED IS OK
1769                                          ;NOW WE ARE READY TO FIND OUT THE DEVICE INTERRUPT VECTOR
1770  002310  016767  176744  176744          MOV     DLBASE,KEEPADD  ;GET 1ST ADDRESS VALUE
1771  002316  004767  005456                  JSR     PC,DLADDR       ;GO FORM DL ADDRESSES FOR
1772                                                                    ;1ST DEVICE SELECTED
1773  002322  016767  176734  176734          MOV     KEEPADD,BASEADD ;RESTORE 1ST DEVICE ADDRESS
1774  002330  104401  020270          VECT:   TYPE    ,MVECT          ;ASK USER FOR A VECTOR ADDRESS
1775  002334  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
1776                                                                    ;AND STORE ON TOP OF STACK
1777                                          ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
1778  002336  012600                          MOV     (SP)+,R0        ;GET THE ANSWER TYPED
1779  002340  020027  000776                  CMP     R0,#776         ;IS THE NUMBER TOO HIGH?
1780  002344  101032                          BHI     RETRY1          ;IF YES - GO TO RETRY SITUATION
1781  002346  020027  000300                  CMP     R0,#300         ;IS THE NUMBER TOO LOW?
1782  002352  103427                          BLO     RETRY1          ;IF YES - GO TO RETRY SITUATION
1783  002354  132700  000001                  BITB    #BIT0,R0        ;NUMBER IS IN RANGE BUT IS IT
1784                                                                    ;ON AN EVEN BOUNDARY?
1785  002360  001024                          BNE     RETRY1          ;IF NO - GO TO RETRY SITUATION
1786                                          ;CHECK TO SEE IF THE USER RESPONSE WAS TRULY A RCVR VECTOR ADDRESS
1787  002362  032700  000007                  BIT     #7,R0           ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
1788                                                                    ;USER RESPONSE EQUAL TO A ZERO?
1789  002366  001021                          BNE     RETRY1          ;BRANCH IF NOT
1790  002370  010067  177024                  MOV     R0,DLVECT       ;THE VECTOR VALUE TYPED IS OK
1791                                                                    ;STORE FOR FUTURE USE
1792  002374  016767  177020  176664          MOV     DLVECT,KEEPIV   ;GET THE FIRST VECTOR VALUE
1793  002402  016767  177012  176660          MOV     DLVECT,BASEIV   ;SAVE FIRST VECTOR VALUE
1794  002410  000414                          BR      HOWMANY         ;GO TO SEE IF USER WANTS MORE
1795                                                                    ;THAN 1 DEVICE
1796  002412  104401  001252          RETRY:  TYPE,   $QUES           ;TYPE '?' INDICATING USER TYPED
1797                                                                    ;SOMETHING WRONG FOR CHARACTER LENGTH
1798  002416  000167  177522                  JMP     GO              ;GO BACK TO REISSUE QUESTION
1799  002422  104401  001252          RETRY0: TYPE    ,$QUES          ;TYPE '?' INDICATING USER TYPE
1800                                                                    ;SOMETHING WRONG FOR 1ST ADDRESS
1801  002426  000167  177612                  JMP     FIRSTD          ;GO BACK TO REISSUE QUESTION
1802  002432  104401  001252          RETRY1: TYPE    ,$QUES          ;TYPE '?' INDICATING USER TYPED
1803                                                                    ;SOMETHING WRONG FOR VECTOR
1804  002436  000167  177666                  JMP     VECT            ;GO BACK TO REISSUE QUESTION
1805  002442  104401  020346          HOWMANY:TYPE    ,MULDEV         ;ASK USER IF HE WISHES TO RUN
1806                                                                    ;MULTIPLE DEVICES
1807  002446  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
1808                                                                    ;AND STORE ON TOP OF STACK
1809  002450  012600                          MOV     (SP)+,R0        ;GET THE ANSWER TYPED
1810  002452  005700                          TST     R0              ;WAS THE ANSWER YES?
1811  002454  001003                          BNE     1$              ;BRANCH IF IT WAS
1812  002456  005067  176610                  CLR     MULTD           ;OTHERWISE, INITIALIZE FLAG TO
1813                                                                    ;INDICATE NON-MULTIPLE DEVICES
1814  002462  000402                          BR      2$              ;SKIP NEXT INSTRUCTION
1815  002464  105167  176602          1$:     COMB    MULTD           ;INITIALIZE FLAG TO INDICATE
1816                                                                    ;RUNNING OF MULTIPLE DEVICES
1817  002470                          2$:
1818  002470  105767  176576                  TSTB    MULTD           ;ARE THERE MULTIPLE DEVICES ON
1819                                                                    ;THE SYSTEM?
1820  002474  100406                          BMI     LASTD   ;IF SO, GO TO ASK NEXT QUESTION
```

```
1821  002476  005067  176572          CLR     ACTREG          ;CLEAR DEVICE ACTIVE FLAG TO
1822                                                          ;INDICATE NO RUNNING OF MULTIPLE
1823                                                          ;DEVICES
1824  002502  005067  176570          CLR     ROTADD          ;CLEAR DEVICE ADDRESS POINTER IN
1825                                                          ;USE WHEN RUNNING MULTIPLE DEVICES
1826  002506  000167  000160          JMP     CONQUES         ;SKIP ASKING NEXT QUESTION
1827  002512                  LASTD:
1828                          ;WE WILL NOW BEGIN TO SET UP THE DEVICE ACTIVE REGISTER FOR RUNNING
1829                          ;MULTIPLE DL11 DEVICES
1830  002512  104401  020433          TYPE    ,MLASTD         ;ASK USER FOR THE RECEIVER
1831                                                          ;CONTROL REGISTER ADDRESS OF
1832                                                          ;HIS LAST DEVICE
1833  002516  104412                  RDOCT                   ;ACCEPT THE ANSWER TYPED BY
1834                                                          ;USER AND STORE ON TOP OF STACK
1835                          ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
1836  002520  012600          1$:     MOV     (SP)+,R0        ;GET THE ANSWER TYPED
1837  002522  020027  176170          CMP     R0,#176170      ;IS THE NUMBER TOO HIGH?
1838  002526  101132                  BHI     RETRY2          ;IF YES - GO TO RETRY SITUATION
1839  002530  020027  175610          CMP     R0,#175610      ;IS THE NUMBER TOO LOW?
1840  002534  103527                  BLO     RETRY2          ;IF YES - GO TO RETRY SITUATION
1841  002536  132700  000001          BITB    #BIT0,R0        ;NUMBER IS IN RANGE BUT IS IT
1842                                                          ;ON AN EVEN BOUNDARY?
1843  002542  001124                  BNE     RETRY2          ;IF NOT - GO TO RETRY SITUATION
1844                          ;CHECK TO SEE IF USER RESPONSE WAS TRULY A RCVR STATUS REGISTER
1845  002544  032700  000007          BIT     #7,R0           ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
1846                                                          ;USER RESPONSE EQUAL TO A ZERO?
1847  002550  001121                  BNE     RETRY2          ;BRANCH IF NOT
1848  002552  010067  176522          MOV     R0,LASTADD      ;THE LAST ADDRESS VALUE TYPED IS OK
1849                                                          ;STORE FOR FUTURE USE
1850                          ;NOW WE BEGIN TO ACTUALLY INITIALIZE THE DEVICE ACTIVE REGISTER
1851                          ;FROM WHICH THE PROGRAM WILL CYCLE UNTIL ALL DEVICES HAVE BEEN TESTED
1852  002556  012767  000001  176512          MOV     #1,ROTADD       ;SET UP POINTER FOR 'ACTREG'
1853  002564  005067  176504                  CLR     ACTREG          ;CLEAR DEVICE ACTIVE REGISTER
1854  002570  056767  176502  176476  2$:     BIS     ROTADD,ACTREG   ;MAKE 1ST DEVICE ACTIVE
1855  002576  000241                  CLC                     ;CLEAR CARRY BIT FOR POINTER
1856                                                          ;ROTATION
1857  002600  006167  176472          ROL     ROTADD          ;ARE WE PAST 16 LINE RANGE?
1858  002604  103422                  BCS     3$              ;BRANCH IF YES
1859  002606  062767  000010  176450          ADD     #10,BASEADD     ;STEP UP BASE ADDRESS
1860  002614  026767  176460  176442          CMP     LASTADD,BASEADD ;IS THIS THE LAST DEVICE?
1861  002622  101362                  BHI     2$              ;BRANCH IF NOT
1862                          ;NOTE:  IF THIS PATH IS TAKEN IT IS ASSUMED THAT AT LEAST 2 DEVICES
1863                          ;       EXIST AND THAT ALL ADDRESSING IS CONTIGUOUS
1864  002624  056767  176446  176442          BIS     ROTADD,ACTREG   ;INDICATE NEXT DEVICE ACTIVE
1865  002632  012767  000001  176436          MOV     #1,ROTADD       ;RESET POINTER FOR 'ACTREG' FOR
1866                                                          ;LATER USE IN END OF PASS ROUTINE
1867  002640  016767  176416  176416          MOV     KEEPAD,BASEADD  ;RESET 1ST DEVICE RECEIVER
1868                                                          ;CONTROLLER REGISTER ADDRESS FOR
1869                                                          ;LATER USE IN END OF PASS ROUTINE
1870  002646  000167  000020          JMP     CONQUES         ;GO TO CONTINUE QUESTIONING OF USER
1871  002652                  3$:
1872                          ;IF WE TAKE THIS PATH IT APPEARS THAT THERE ARE NOT AT LEAST
1873                          ;TWO DEVICES PRESENT - IN RESPONSE TO USER TYPING 'YES' TO MULTIPLE
1874                          ;DEVICES QUESTION
1875  002652  016767  176404  176404          MOV     KEEPADD,BASEADD ;RESET 1ST DEVICE RECEIVER
1876                                                          ;CONTROLLER REGISTER ADDRESS
```

```
1877  002660  104401  020531          TYPE    ,MRANGE         ;INFORM USER TO CHECK AND RETYPE
1878                                                          ;THE LAST DEVICE RCSR ADDRESS
1879  002664  104412                  RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
1880                                                          ;AND STORE ON TOP OF STACK
1881  002666  000167  177626          JMP     1$
1882  002672                  CONQUES:
1883                          ;IF WE HAVE REACHED THIS PORTION WE KNOW:
1884                          ;       A)  THE 'RXCSR' ADDRESS OF THE 1ST DEVICE
1885                          ;       B)  THE 'RXCSR' ADDRESS OF THE LAST DEVICE, SAND
1886                          ;       C)  THE INTERRUPT VECTOR OF THE 1ST DEVICE
1887                          ;NOW LET'S FIND THE PRIORITY LEVEL
1888  002672  104401  020615          TYPE    ,PLEVEL         ;ASK USER FOR PRIORITY LEVEL
1889  002676  104412                  RDOCT                   ;ACCEPT ANSWER TYPED BY USER AND
1890                                                          ;STORE ON TOP OF STACK
1891  002700  012600                  MOV     (SP)+,R0        ;GET THE ANSWER TYPED
1892  002702  020027  000007          CMP     R0,#7           ;IS THE NUMBER TOO HIGH?
1893  002706  101046                  BHI     RETRY3          ;IF YES - GO TO RETRY SITUATION
1894  002710  020027  000004          CMP     R0,#4           ;IS THE NUMBER TOO LOW?
1895  002714  103443                  BLO     RETRY3          ;IF YES GO TO RETRY SITUATION
1896  002716  010067  176360          MOV     R0,DLPRI        ;THE PRIORITY TYPED IN IS OK
1897                                                          ;STORE FOR FUTURE USE
1898
1899                          ;THIS SECTION WILL CALCULATE THE PRIORITY LEVEL FOR THE
1900                          ;PROCESSOR BASED ON THE USER RESPONSE FOR PRIORITY LEVEL OF THE
1901                          ;DEVICE
1902  002722  006367  176354          ASL     DLPRI           ;FORM BITS <7-5> OF PSW
1903  002726  006367  176350          ASL     DLPRI           ;       .
1904  002732  006367  176344          ASL     DLPRI           ;       .
1905  002736  006367  176340          ASL     DLPRI           ;       .
1906  002742  006367  176334          ASL     DLPRI           ;       .
1907  002746  016767  176330  176330  MOV     DLPRI,LESS1     ;START TO FORM LEVEL TO ALLOW
1908                                                          ;INTERRUPTS
1909  002754  162767  000001  176322  SUB     #1,LESS1        ;DROP DEVICE LEVEL PRIORITY
1910                                                          ;BY 1 LEVEL FOR PSW
1911  002762  042767  000037  176314  BIC     #37,LESS1       ;MAKE SURE THE T,N,Z,V & C
1912                                                          ;BITS FOR THE PROCESSOR ARE CLEAR
1913  002770  005077  176420  FLUSH:  CLR     @DLXCSR         ;CLEAR OUT BOTH CSR'S
1914  002774  005077  176410          CLR     @DLRCSR
1915  003000  005777  176406          TST     @DLRDBR         ;FLUSH RCVR 'DONE'' BIT
1916  003004  005777  176402          TST     @DLRDBR
1917  003010  000167  000020          JMP     TST1            ;BEGIN TESTING
1918  003014  104401  001252  RETRY2: TYPE    ,$QUES          ;TYPE '?' INDICATING USER TYPED
1919                                                          ;SOMETHING WRONG FOR LAST ADDRESS
1920  003020  000167  177466          JMP     LASTD   ;GO BACK TO REISSUE QUESTION
1921  003024  104401  001252  RETRY3: TYPE    ,$QUES          ;TYPE'?' INDICATING USER TYPED
1922                                                          ;SOMETHING WRONG FOR PRIORITY
1923  003030  000167  177636          JMP     CONQUES         ;GO BACK TO REISSUE QUESTION
1924
1925
1926                          ;;*****************************************************************
1927                          ;*TEST 1        TEST THAT REFERENCE TO RCSR DOES NOT CAUSE TIMEOUT
1928                          ;;*****************************************************************
1929  003034  000004          TST1:   SCOPE
1930  003036  016746  174742          MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
1931  003042  012767  003060  174734  MOV     #1$,ERRVEC      ;GO TO 1$ IF TIMEOUT
1932  003050  016702  176334          MOV     DLRCSR,R2       ;REGADR = RCSR ADR
```

```
 1933  003054  005712                        TST     (R2)            ;USE REGADR ON BUS
 1934  003056  000407                        BR      3$              ;;<GO TO NEXT TEST IF NO TIMEOUT>
 1935  003060  004767  011614          1$:   JSR     PC,SUERT1       ;GO SET UP ERROR INFO
 1936  003064  012767  003074  176152        MOV     #2$,$ESCAPE     ;RETURN TO 2$ AFTER ERROR PRINT
 1937  003072  104001                        ERROR+1                 ;DL REFERENCE CAUSED BUS TIMEOUT
 1938  003074  022626                  2$:   CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
 1939  003076  012667  174702          3$:   MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
 1940
 1941                                        ;;**************************************************************************
 1942                                        ;*TEST 2         TEST THAT REFERENCE TO XCSR DOES NOT CAUSE TIMEOUT
 1943                                        ;;**************************************************************************
 1944  003102  000004                  TST2: SCOPE
 1945  003104  016746  174674                MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
 1946  003110  012767  003126  174666        MOV     #1$,ERRVEC      ;GO TO 1$ IF TIMEOUT
 1947  003116  016702  176272                MOV     DLXCSR,R2       ;REGADR = XCSR ADR
 1948  003122  005712                        TST     (R2)            ;USE REGADR ON BUS
 1949  003124  000407                        BR      3$              ;;<GO TO NEXT TEST IF NO TIMEOUT>
 1950  003126  004767  011546          1$:   JSR     PC,SUERT1       ;GO SET UP ERROR INFO
 1951  003132  012767  003142  176104        MOV     #2$,$ESCAPE     ;RETURN TO 2$ AFTER ERROR PRINT
 1952  003140  104001                        ERROR+1                 ;DL REFERENCE CAUSED BUS TIMEOUT
 1953  003142  022626                  2$:   CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
 1954  003144  012667  174634          3$:   MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
 1955
 1956                                        ;;**************************************************************************
 1957                                        ;*TEST 3         TEST THAT REFERENCE TO RDBR DOES NOT CAUSE TIMEOUT
 1958                                        ;;**************************************************************************
 1959  003150  000004                  TST3: SCOPE
 1960  003152  016746  174626                MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
 1961  003156  012767  003174  174620        MOV     #1$,ERRVEC      ;GO TO 1$ IF TIMEOUT
 1962  003164  016702  176222                MOV     DLRDBR,R2       ;REGADR = RDBR ADR
 1963  003170  005712                        TST     (R2)            ;USE REGADR ON BUS
 1964  003172  000407                        BR      3$              ;;<GO TO NEXT TEST IF NO TIMEOUT>
 1965  003174  004767  011500          1$:   JSR     PC,SUERT1       ;GO SET UP ERROR INFO
 1966  003200  012767  003210  176036        MOV     #2$,$ESCAPE     ;RETURN TO 2$ AFTER ERROR PRINT
 1967  003206  104001                        ERROR+1                 ;DL REFERENCE CAUSED BUS TIMEOUT
 1968  003210  022626                  2$:   CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
 1969  003212  012667  174566          3$:   MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
 1970
 1971                                        ;;**************************************************************************
 1972                                        ;*TEST 4         TEST THAT REFERENCE TO XDBR DOES NOT CAUSE TIMEOUT
 1973                                        ;;**************************************************************************
 1974  003216  000004                  TST4: SCOPE
 1975  003220  016746  174560                MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
 1976  003224  012767  003242  174552        MOV     #1$,ERRVEC      ;GO TO 1$ IF TIMEOUT
 1977  003232  016702  176160                MOV     DLXDBR,R2       ;REGADR = XDBR ADR
 1978  003236  005712                        TST     (R2)            ;USE REGADR ON BUS
 1979  003240  000407                        BR      3$              ;;<GO TO NEXT TEST IF NO TIMEOUT>
 1980  003242  004767  011432          1$:   JSR     PC,SUERT1       ;GO SET UP ERROR INFO
 1981  003246  012767  003256  175770        MOV     #2$,$ESCAPE     ;RETURN TO 2$ AFTER ERROR PRINT
 1982  003254  104001                        ERROR+1                 ;DL REFERENCE CAUSED BUS TIMEOUT
 1983  003256  022626                  2$:   CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
 1984  003260  012667  174520          3$:   MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
 1985
 1986                                        ;;**************************************************************************
 1987                                        ;*TEST 5         TEST THAT RCSR IS ALL ZEROES ON ENTRY
 1988                                        ;;**************************************************************************
```

```
                        ;T5      TEST THAT RCSR IS ALL ZEROES ON ENTRY                                  SEQ 0038

1989  003264  000004     TST5:   SCOPE
1990  003266  005004             CLR     R4                      ;RESULT IN RCSR S/B = 0
1991  003270  016702  176114     MOV     DLRCSR,R2               ;REGADR = RCSR ADR
1992  003274  020412             CMP     R4,(R2)                 ;[RCSR]=000000 ??
1993  003276  001403             BEQ     TST6                    ;;<BR IF YES>
1994  003300  004767  011314     JSR     PC,SUER2                ;GO SET UP ERROR INFO
1995  003304  104002             ERROR+2                         ;RCSR NOT CLEAR ON START UP
1996                     ;;*****************************************************************
1997                     ;*TEST 6        TEST THAT 'READY' BIT IS ONLY BIT SET IN XCSR
1998                     ;;*****************************************************************
1999  003306  000004     TST6:   SCOPE
2000  003310  012704  000200     MOV     #200,R4                 ;RESULT IN XCSR S/B = 000200
2001  003314  016702  176074     MOV     DLXCSR,R2               ;REGADR = XCSR ADR
2002  003320  020412             CMP     R4,(R2)                 ;[XCSR]=000200 ??
2003  003322  001403             BEQ     TST7                    ;;<BR IF YES>
2004  003324  004767  011270     JSR     PC,SUER2                ;GO SETUP ERROR INFO
2005  003330  104002             ERROR+2                         ;[XCSR] INCORRECT ON START UP
2006                     ;;*****************************************************************
2007                     ;*TEST 7        TEST THAT 'MAINT' BIT CAN BE SET AND CLEARED
2008                     ;;*****************************************************************
2009  003332  000004     TST7:   SCOPE
2010  003334  012704  000204     MOV     #204,R4                 ;RESULT IN XCSR S/B = 000204
2011  003340  016702  176050     MOV     DLXCSR,R2               ;REGADR = XCSR ADR
2012  003344  052712  000004     BIS     #BIT2,(R2)              ;SET THE 'MAINT' BIT
2013  003350  020412             CMP     R4,(R2) ;RESULT IN XCSR OK ??
2014  003352  001403             BEQ     1$                      ;;<BR IF YES>
2015  003354  004767  011240     JSR     PC,SUER2                ;GO SET UP ERROR INFO
2016  003360  104002             ERROR+2                         ;MAINT. BIT FAILED TO SET PROPERLY
2017  003362  012704  000200 1$:  MOV     #200,R4                 ;RESULT IN XCSR S/B = 000200
2018  003366  042712  000004     BIC     #BIT2,(R2)              ;NOW CLEAR THE 'MAINT' BIT
2019  003372  020412             CMP     R4,(R2)                 ;RESULT IN XCSR OK ??
2020  003374  001403             BEQ     TST10                   ;;<BR IF YES>
2021  003376  004767  011216     JSR     PC,SUER2                ;GO SET UP ERROR INFO
2022  003402  104002             ERROR+2                         ;MAINT BIT FAILED TO CLEAR PROPERLY
2023                     ;;*****************************************************************
2024                     ;*TEST 10       TEST THAT XMIT I.E. CAN CAUSE AN INTR
2025                     ;;*****************************************************************
2026  003404  000004     TST10:  SCOPE
2027  003406  005067  176034     CLR     INTFLG                  ;INIT SOFTWARE INTR FLAG
2028  003412  012767  000001 176024  MOV  #1,TIMR3                ;SET TIMER FOR DELAY                   (CHGC6)
2029  003420  016705  175774     MOV     DLVECT,R5               ;GET VECTOR ADDRESS
2030  003424  012765  003520 000004  MOV  #2$,4(R5)               ;GO TO 4$ ON INTR
2031  003432  016765  175644 000006  MOV  DLPRI,6(R5)            ;PRIORITY LEVEL 4
2032  003440  005005             CLR     R5                      ;INIT INTR. TIMER
2033  003442  012704  000200     MOV     #200,R4                 ;RESULT IN XCSR S/B = 000200
2034  003446  016702  175742     MOV     DLXCSR,R2               ;REGADR = XCSR ADR
2035  003452  052712  000100     BIS     #100,(R2)               ;SET INTR. ENABLE BIT 06
2036  003456  005767  175764 1$:  TST     INTFLG                  ;DID INTR OCCUR YET ??
2037  003462  001023             BNE     3$                      ;BR IF IT DID
2038  003464  005305             DEC     R5                      ;COUNT THE TIMER
2039  003466  001373             BNE     1$                      ;BR IF NO TIMEOUT
2040  003470  005367  175750     DEC     TIMR3                   ;REDUCE ADDED DELAY                    (CHGC6)
2041  003474  001770             BEQ     1$                      ;BRANCH BACK ON FIRST PASS             (CHGC6)
2042  003476  012704  000300     MOV     #300,R4                 ;RESULT IN XCSR S/B = 000300
2043  003502  004767  011112     JSR     PC,SUER2                ;GO SETUP ERROR INFO
2044  003506  012767  003516 175530  MOV  #4$,$ESCAPE           ;RETURN TO 4$ AFTER ERROR PRINT
```

```
2045  003514  104002                        ERROR+2                 ;INTR.  FAILED
2046  003516                          4$:
2047  003516  000412                        BR      TST11           ;;<GO TO NEXT TEST>
2048  003520  005167  175722          2$:   COM     INTFLG          ;SET THE SOFTWARE FLAG
2049  003524  042712  000100                BIC     #100,(R2)       ;TURN OFF I.E. BIT
2050  003530  000002                        RTI                     ;RETURN CONTROL TO INTR. ROUTINE
2051  003532  020412                  3$:   CMP     R4,(R2)         ;RESULT IN XCSR OK ??
2052  003534  001403                        BEQ     TST11           ;;<BR IF YES>
2053  003536  004767  011056                JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2054  003542  104002                        ERROR+2                 ;XMIT INTR. NOT SERVICED PROPERLY
2055                                  ;;**************************************************************
2056                                  ;*TEST 11        TEST THAT RCVR I.E. BIT CAN BE SET AND CLEARED
2057                                  ;;**************************************************************
2058  003544  000004                  TST11:  SCOPE
2059  003546  012704  000100                MOV     #100,R4         ;RESULT IN RCSR S/B = 000100
2060  003552  016702  175632                MOV     DLRCSR,R2       ;REGADR = RCSR ADR
2061  003556  052712  000100                BIS     #BIT6,(R2)      ;SET I.E. BIT
2062  003562  020412                        CMP     R4,(R2)         ;DID IT SET PROPERLY ??
2063  003564  001403                        BEQ     1$              ;;<BR IF YES>
2064  003566  004767  011026                JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2065  003572  104002                        ERROR+2                 ;RCVR I.E. BIT FAILED TO SET PROPERLY
2066  003574  005004                  1$:   CLR     R4              ;RESULT IN RCSR S/B = 000000
2067  003576  042712  000100                BIC     #BIT6,(R2)      ;CLEAR THE I.E. BIT
2068  003602  020412                        CMP     R4,(R2)         ;DID IT CLEAR PROPERLY ??
2069  003604  001403                        BEQ     TST12           ;;<BR IF YES>
2070  003606  004767  011006                JSR     PC,SUER2        ;GO SET UP ERROR INFO
2071  003612  104002                        ERROR+2                 ;RCVR I.E. BIT FAILED TO CLEAR PROPERLY
2072                                  ;;**************************************************************
2073                                  ;*TEST 12        TEST THAT RCVR 'DONE'' CAN GENERATE AN INTR.
2074                                  ;;**************************************************************
2075  003614  000004                  TST12:  SCOPE
2076  003616  016705  175576                MOV     DLVECT,R5       ;GET THE VECTOR ADDRESS
2077  003622  012725  004000                MOV     #3$,(R5)+       ;GO TO 3$ ON RCVR INTR.
2078  003626  016715  175450                MOV     DLPRI,(R5)      ;AT LEVEL 4
2079  003632  005067  175610                CLR     INTFLG          ;INIT THE SOFTWARE FLAG
2080  003636  005005                        CLR     R5              ;INIT INTR. TIMER
2081  003640  105067  175340                CLRB    $TMP1           ;INIT WHERE DATA WILL BE STORED
2082  003644  016702  175540                MOV     DLRCSR,R2       ;REGADR = RCSR ADR
2083  003650  005012                        CLR     (R2)            ;INIT THE RCSR TO 000000
2084  003652  052712  000100                BIS     #BIT6,(R2)      ;ENABLE RCVR INTERRUPTS
2085  003656  052762  000004  000004        BIS     #BIT2,4(R2)     ;NOW TURN ON MAINT MODE
2086  003664  112767  000252  175312        MOVB    #252,$TMP1      ;GET DATA PATTERN AND
2087  003672  004767  011614                JSR     PC,UPMASK       ;GO MASK OFF BITS AS A FUNCTION OF
2088                                                                  ;CHARACTER LENGTH ( 5, 6, 7, OR 8 BITS)
2089  003676  116700  175330                MOVB    $TMP14,R0       ;SAVE DATA PATTERN FOR FURTHER USE
2090  003702  116762  175324  000006        MOVB    $TMP14,6(R2)    ;LOAD XMIT BUFFER REG.
2091  003710  005767  175532          1$:   TST     INTFLG          ;DID RCVR INTR. YET ??
2092  003714  001044                        BNE     4$              ;BR IF IT DID
2093  003716  005305                        DEC     R5              ;COUNT THE TIMER
2094  003720  001373                        BNE     1$              ;BR IF NO TIMEOUT
2095  003722  013767  177776  175252        MOV     @#PSW,$TMP0     ;SAVE ERROR PSW
2096  003730  042762  000004  000004        BIC     #BIT2,4(R2)     ;DISABLE MAINT MODE
2097  003736  042712  000100                BIC     #100,(R2)       ;DISABLE RCVR INTR.
2098  003742  010667  175230                MOV     SP,$REG6        ;SAVE THE ERROR SP
2099  003746  010201                        MOV     R2,R1           ;DEVADR = RCSR ADR
2100  003750  011203                        MOV     (R2),R3         ;GET THE WAS DATA
```

```
2101  003752  012704  000200              MOV     #200,R4         ;[RCSR] S/B = 000200
2102  003756  004767  010664              JSR     PC,SUERR1       ;GO SET UP ERROR INFO.
2103  003762  012767  003772  175254      MOV     #2$,$ESCAPE     ;RETURN TO 2$ AFTER ERROR ALWAYS
2104  003770  104002                      ERROR+2                 ;RCVR INTERRUPT FAILED
2105  003772  005762  000002      2$:     TST     2(R2)           ;REFERENCE RCVR DATA BUFFER
2106                                                              ;TO CLEAR RCSR IN CASE RCVR
2107                                                              ;INTERRUPTS COULD NOT BE ENABLED
2108  003776  000437                      BR      TST13           ;;<GO TO NEXT TEST>
2109  004000  042762  000004  000004  3$: BIC     #BIT2,4(R2)     ;DISABLE THE MAINT MODE
2110  004006  116267  000002  175170      MOVB    2(R2),$TMP1     ;GET THE RECEIVED DATA
2111  004014  042712  000100              BIC     #BIT6,(R2)      ;TURN OFF RCVR INTR. ENAB
2112  004020  005167  175422              COM     INTFLG          ;SET THE SOFTWARE FLAG
2113  004024  000002                      RTI                     ;RETURN TO MAINLINE
2114  004026  005004              4$:     CLR     R4              ;[RCSR] S/B=0
2115  004030  005712                      TST     (R2)            ;IS IT ALL ZEROES ??
2116  004032  001403                      BEQ     5$              ;;<BR IF YES>
2117  004034  004767  010560              JSR     PC,SUER2        ;GO SET UP ERROR INFO
2118  004040  104002                      ERROR+2                 ;RCVR INTR NOT SERVICED PROPERLY
2119  004042  016701  175344      5$:     MOV     DLRDBR,R1       ;SAVE WAS ADDRESS
2120  004046  016702  175344              MOV     DLXDBR,R2       ;SAVE THE S/B ADDRESS
2121  004052  004767  011434              JSR     PC,UPMASK       ;GET THE WAS DATA AND
2122                                                              ;GO MASK OFF BITS AS A FUNCTION OF
2123                                                              ;CHARACTER LENGTH ( 5, 6, 7, OR 8 BITS)
2124  004056  116703  175150              MOVB    $TMP14,R3       ;SET UP FOR ERROR CHECKING
2125  004062  110004                      MOVB    R0,R4           ;GET THE S/B DATA
2126  004064  020403                      CMP     R4,R3           ;WAS = S/B ??
2127  004066  001403                      BEQ     TST13           ;;<BR IF YES>
2128  004070  004767  010552              JSR     PC,SUERR1       ;GO SET UP THE ERROR INFO
2129  004074  104003                      ERROR+3                 ;DATA COMPARE ERROR
2130                          ;;**********************************************************************
2131                          ;*TEST 13        TEST THAT 'REQ TO SEND" ASSERTS 'RING"
2132                          ;;**********************************************************************
2133  004076  000004      TST13:  SCOPE
2134  004100  032777  010000  175032      BIT     #SW12,@SWR      ;ARE WE TESTING /C OR /D MODEL?
2135  004106  001047                      BNE     TST14           ;;<BRANCH IF YES>
2136  004110  012704  140004              MOV     #140004,R4      ;RESULT IN RCSR S/B = 140004
2137  004114  016702  175270              MOV     DLRCSR,R2       ;REGADR = RCSR ADR
2138  004120  005012                      CLR     (R2)            ;INIT THE RCSR TO 000000
2139  004122  052712  000004              BIS     #BIT2,(R2)      ;SET 'REQ TO SEND"
2140  004126  032777  100000  175254      BIT     #BIT15,@DLRCSR  ;DID 'RING" SET 'DATA SET INT" ?
2141  004134  001003                      BNE     1$              ;;<BR IF YES>
2142  004136  004767  010456              JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2143  004142  104002                      ERROR+2                 ;'RING" TRANSITION FAILED TO SET 'DATA SET INT"
2144                                                              ;NOTE: 'BIT  #BIT15,(R2)" RESETS BIT15
2145  004144  012704  040004      1$:     MOV     #40004,R4       ;RESULT IN RCSR S/B = 40004
2146  004150  020412                      CMP     R4,(R2)         ;BOTH 'RING" AND 'REQ TO SEND" ASSERTED ?
2147  004152  001403                      BEQ     2$              ;;<BR IF YES>
2148  004154  004767  010440              JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2149  004160  104002                      ERROR+2                 ;'RING" OR 'REQ TO SEND" FAILED TO SET
2150  004162  005004              2$:     CLR     R4              ;RESULT IN RCSR S/B = 000000
2151  004164  042712  000004              BIC     #BIT2,(R2)      ;CLEAR 'REQ TO SEND"
2152  004170  032777  100000  175212      BIT     #BIT15,@DLRCSR  ;DID 'DATA SET INT" GET SET ??
2153  004176  001403                      BEQ     3$              ;;<BR IF NOT>
2154  004200  004767  010414              JSR     PC,SUER2        ;GO SET UP ERROR INFO
2155  004204  104002                      ERROR+2                 ;CLEARING 'RING" SET 'DATA SET INT"
2156  004206  020412              3$:     CMP     R4,(R2)         ;RCSR CONTAIN ALL ZEROES ??
```

```
2157  004210  001406                        BEQ     TST14           ;;<BR IF YES>
2158  004212  004767  010402                JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2159  004216  016767  000002  174754        MOV     .+6,$REG7       ;SAVE THE ERROR PC
2160  004224  104002                        ERROR+2                 ;CLEARING 'REQ TO SEND' FAILED TO CLEAR 'RING'
2161                            ;;**********************************************************
2162                            ;*TEST 14        TEST THAT 'SEC XMIT' ASSERTS 'SEC REC' AND 'DATA SET INT'
2163                            ;;**********************************************************
2164  004226  000004            TST14:  SCOPE
2165  004230  032777  010000  174702        BIT     #SW12,@SWR      ;ARE WE TESTING /C OR /D MODEL?
2166  004236  001046                        BNE     TST15           ;;<BRANCH IF YES>
2167  004240  016702  175144                MOV     DLRCSR,R2       ;REGADR = RCSR ADR
2168  004244  005012                        CLR     (R2)            ;INIT RCSR TO 000000
2169  004246  012704  102010                MOV     #102010,R4      ;CONTENTS OF RCSR S/B = 102010
2170  004252  052712  000010                BIS     #BIT3,(R2)      ;SET 'SEC XMIT' BIT
2171  004256  032777  100000  175124        BIT     #BIT15,@DLRCSR  ;DID 'DATA SET INT' SET ??
2172  004264  001003                        BNE     1$              ;;<BR IF YES>
2173  004266  004767  010326                JSR     PC,SUER2        ;GO SET UP ERROR INFO
2174  004272  104002                        ERROR+2                 ;'DATA SET INT' FAILED TO SET-NOTE THAT
2175                                                                ;'BIT  #BIT15,(R2)' RESETS BIT15
2176  004274  012704  002010        1$:     MOV     #2010,R4        ;RESULT IN RCSR S/B = 2010
2177  004300  020412                        CMP     R4,(R2)         ;ARE 'SEC XMIT' AND 'SEC REC' BOTH SET ?
2178  004302  001403                        BEQ     2$              ;;<BR IF YES>
2179  004304  004767  010310                JSR     PC,SUER2        ;GO SET UP ERROR INFO
2180  004310  104002                        ERROR+?                 ;'SEC XMIT' OR 'SEC REC' FAILED TO SET
2181                                                                ;OR 'DATA SET INT' FAILED TO BE CLEARED
2182                                                                ;WHEN REFERENCING RCSR
2183  004312  012704  100000        2$:     MOV     #BIT15,R4       ;RESULT IN RCSR S/B = 100000
2184  004316  042712  000010                BIC     #BIT3,(R2)      ;CLEAR 'SEC XMIT' BIT
2185  004322  032777  100000  175060        BIT     #BIT15,@DLRCSR  ;DID CLEARING IT SET 'DATA SET INT'??
2186  004330  001003                        BNE     3$              ;;<BR IF YES>
2187  004332  004767  010262                JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2188  004336  104002                        ERROR+2                 ;CLEARING 'SEC XMIT' FAILED TO SET 'DATA
2189                                                                ;SET INT. (NOTE THAT REFERENCING RCSR
2190                                                                ; CLEARS 'DATA SET INT'
2191  004340  005004        3$:     CLR     R4                      ;RESULT IN RCSR S/B = 000000
2192  004342  020412                        CMP     R4,(R2)         ;'SEC XMIT' AND 'SEC REC' CLEAR ?
2193  004344  001403                        BEQ     TST15           ;;<BR IF YES>
2194  004346  004767  010246                JSR     PC,SUER2        ;GO SETUP ERROR INFO
2195  004352  104002                        ERROR+2                 ;'SEC XMIT' OR 'SEC REC' FAILED TO CLEAR
2196                                                                ;OR REFERENCING RCSR  FAILED TO CLEAR 'DATA SET INT'
2197                            ;;**********************************************************
2198                            ;*TEST 15        TEST THAT 'DTR' CAN ASSERT 'CLR TO SEND' AND 'CAR DET'
2199                            ;;**********************************************************
2200  004354  000004            TST15:  SCOPE
2201  004356  032777  010000  174554        BIT     #SW12,@SWR      ;ARE WE TESTING /C OR /D MODEL?
2202  004364  001046                        BNE     TST16           ;;<BRANCH IF YES>
2203  004366  016702  175016                MOV     DLRCSR,R2       ;REGADR = RCSR ADR
2204  004372  005012                        CLR     (R2)            ;INIT RCSR TO 000000
```

```
2205  004374  012704  130002          MOV     #130002,R4      ;RESULT IN RCSR S/B = 130002
2206  004400  052712  000002          BIS     #BIT1,(R2)      ;SET 'DTR'' BIT
2207  004404  032777  100000  174776  BIT     #BIT15,@DLRCSR  ;DID 'DATA SET INT'' SET ??
2208  004412  001003                  BNE     1$              ;;<BR IF YES>
2209  004414  004767  010200          JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2210  004420  104002                  ERROR+2                 ;'DATA SET INT'' FAILED TO SET -
2211                                                          ;NOTE: THE REFERENCE TO RCSR  ABOVE WILL
2212                                                          ;WILL UNCONDITIONALLY CLEAR RCSR BIT 15.
2213  004422  012704  030002  1$:     MOV     #30002,R4       ;RESULT IN RCSR S/B = 30002
2214  004426  020412                  CMP     R4,(R2)         ;'DTR','CLR TO SEND'', AND ''CAR DET'' ALLSET
2215  004430  001403                  BEQ     2$              ;;<BR IF ALL SET>
2216  004432  004767  010162          JSR     PC,SUER2        ;GO SET UP ERROR INFO
2217  004436  104002                  ERROR+2                 ;'DTR','CLR TO SEND'' OR ''CAR DET'' FAILED
2218                                                          ;TO SET OR 'DATA SET INT'' FAILED TO CLEAR
2219
2220  004440  012704  100000  2$:     MOV     #BIT15,R4       ;RESULT IN RCSR S/B = 100000
2221
2222  004444  042712  000002          BIC     #BIT1,(R2)      ;NOW CLEAR 'DTR''
2223  004450  032777  100000  174732  BIT     #BIT15,@DLRCSR  ;DID 'DATA SET INT'' SET ??
2224  004456  001003                  BNE     3$              ;;<BR IF YES>
2225  004460  004767  010134          JSR     PC,SUER2        ;GO SETUP ERROR INFO
2226
2227  004464  104002                  ERROR+2                 ;'DATA SET INT'' FAILED TO SET WHEN 'DTR''
2228                                                          ;WENT TO A ZERO.
2229  004466  005004          3$:     CLR     R4              ;RESULT IN RCSR S/B = 000000
2230  004470  020412                  CMP     R4,(R2)         ;DID ALL BITS CLEAR??
2231  004472  001403                  BEQ     TST16           ;;<BR IF YES>
2232  004474  004767  010120          JSR     PC,SUER2        ;GO SET UP ERROR INFO
2233  004500  104002                  ERROR+2                 ;'DTR','CLR TO SEND'' OR ''CAR DET'' FAILED
2234                                                          ;TO CLEAR PROPERLY
2235                          ;;**********************************************************************
2236                          ;*TEST 16        TEST THAT 'DATA SET INT ENAB'' CAN SET AND CLEAR
2237                          ;;**********************************************************************
2238  004502  000004          TST16:  SCOPE
2239  004504  032777  010000  174426          BIT     #SW12,@SWR      ;ARE WE TESTING /C OR /D MODEL?
2240  004512  001023                  BNE     TST17           ;;<BRANCH IF YES>
2241  004514  016702  174670          MOV     DLRCSR,R2       ;REGADR = RCSR ADR
2242  004520  012704  000040          MOV     #40,R4          ;RESULT IN RCSR S/B = 000040
2243  004524  052712  000040          BIS     #BIT5,(R2)      ;SET THE 'DATA SET I.E.'' BIT
2244  004530  020412                  CMP     R4,(R2)         ;DID IT SET OK ??
2245  004532  001403                  BEQ     1$              ;;<BR IF YES>
2246  004534  004767  010060          JSR     PC,SUER2        ;GO SET UP ERROR INFO
2247  004540  104002                  ERROR+2                 ;'DAT SET I. E.'' FAILED TO SET
2248  004542  005004          1$:     CLR     R4              ;MAKE S/B DATA = 000000
2249  004544  042712  000040          BIC     #BIT5,(R2)      ;NOW CLEAR THE 'DATA SET I.E.'' BIT
2250  004550  020412                  CMP     R4,(R2)         ;DID IT CLEAR OK ??
2251  004552  001403                  BEQ     TST17           ;;<BR IF YES>
2252  004554  004767  010040          JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2253  004560  104002                  ERROR+2                 ;'DATA SET I.E.'' FAILED TO CLEAR
2254                          ;;**********************************************************************
2255                          ;*TEST 17        TEST THE 'DATA SET I.E.'' CAN CAUSE A RCVR INTR
2256                          ;;**********************************************************************
2257  004562  000004          TST17:  SCOPE
2258  004564  032777  010000  174346          BIT     #SW12,@SWR      ;ARE WE TESTING A /C OR /D MODEL?
2259  004572  001054                  BNE     TST20           ;;<BRANCH IF YES>
2260  004574  016705  174620          MOV     DLVECT,R5       ;GET THE VECTOR ADDR
```

```
2261  004600  012725  004666          MOV     #3$,(R5)+       ;GO TO 3$ ON RCVR INTR.
2262  004604  016715  174472          MOV     DLPRI,(R5)      ;AT LEVEL 4
2263  004610  005005                  CLR     R5              ;INIT INTR. TIMER
2264  004612  005067  174630          CLR     INTFLG          ;INIT SOFTWARE FLAG
2265  004616  005004                  CLR     R4              ;RESULT IN RCSR S/B = 0 AFTER INTR.
2266  004620  016702  174564          MOV     DLRCSR,R2       ;REGADR = RCSR ADR
2267  004624  052712  000040          BIS     #BIT5,(R2)      ;SET THE 'DATA SET I.E.' BIT
2268  004630  052712  000002          BIS     #BIT1,(R2)      ;NOW SET 'DTR' TO GEN INTR.
2269  004634  005767  174606  1$:     TST     INTFLG          ;DID INTR OCCUR YET ??
2270  004640  001016                  BNE     4$              ;BR IF YES
2271  004642  005305                  DEC     R5              ;COUNT THE TIMER
2272  004644  001373                  BNE     1$              ;BR IF NO TIMEOUT
2273  004646  004767  007746          JSR     PC,SUER2        ;GO SET UP ERROR INFO
2274  004652  005012                  CLR     (R2)            ;TURN IT ALL OFF
2275  004654  012767  004664  174362  MOV     #2$,$ESCAPE     ;COME BACK TO 2$ IN ALL CASES
2276  004662  104002                  ERROR+2                 ;'DATA SET' INTR FAILED TO OCCUR
2277  004664                  2$:
2278  004664  000417                  BR      TST20           ;;<GO TO NEXT TEST>
2279  004666  005012          3$:     CLR     (R2)            ;ZERO THE RCSR
2280  004670  005167  174552          COM     INTFLG          ;SET THE SOFTWARE FLAG
2281  004674  000002                  RTI                     ;RETURN TO SENDER
2282  004676  032712  100000  4$:     BIT     #BIT15,(R2)     ;DID 'DATA SET INT' GET SET BY INTR. SERVICE ??
2283  004702  001003                  BNE     5$              ;;<BR IF YES>
2284  004704  004767  007710          JSR     PC,SUER2        ;GO SET UP ERROR INFO
2285  004710  104002                  ERROR+2                 ;DATA SET INTR. NOT SERVICED PROPERLY
2286  004712  020412          5$:     CMP     R4,(R2)         ;ALL BITS IN RCSR CLEAR ??
2287  004714  001403                  BEQ     TST20           ;;<BR IF YES>
2288  004716  004767  007676          JSR     PC,SUER2        ;GO SET UP ERROR INFO
2289  004722  104002                  ERROR+2                 ;INTR. SERVICE FAILED TO CLEAR RCSR
2290                          ;;**********************************************************************
2291                          ;*TEST 20        TEST THAT THE 'BREAK' BIT CAN BE SET AND CLEARED
2292                          ;;**********************************************************************
2293  004724  000004  TST20:  SCOPE
2294  004726  032777  010000  174204          BIT     #SW12,@SWR      ;ARE WE TESTING /C OR /D MODEL?
2295  004734  001024                  BNE     TST21           ;;<BRANCH IF YES>
2296  004736  012704  000201          MOV     #201,R4         ;RESULT S/B = 201 IN XCSR
2297  004742  016702  174446          MOV     DLXCSR,R2       ;SET UP REGADR
2298  004746  052712  000001          BIS     #BIT0,(R2)      ;SET THE 'BREAK' BIT
2299  004752  020412                  CMP     R4,(R2)         ;DID IT SET PROPERLY ??
2300  004754  001403                  BEQ     1$              ;;<BR IF YES>
2301  004756  004767  007636          JSR     PC,SUER2        ;GO SET UP ERROR INFO.
2302  004762  104002                  ERROR+2                 ;'BREAK' BIT FAILED TO SET PROPERLY
2303  004764  012704  000200  1$:     MOV     #200,R4         ;RESULT S/B = 200 IN XCSR
2304  004770  042712  000001          BIC     #BIT0,(R2)      ;CLEAR THE 'BREAK' BIT
2305  004774  020412                  CMP     R4,(R2)         ;DID IT CLEAR PROPERLY ??
2306  004776  001403                  BEQ     TST21           ;;<BR IF YES>
2307  005000  004767  007614          JSR     PC,SUER2        ;GO SET UP ERROR INFO
2308  005004  104002                  ERROR+2                 ;'BREAK' FAILED TO CLEAR PROPERLY
2309
2310
2311                          ;;**********************************************************************
2312                          ;*TEST 21        TEST THAT A 'RESET' CLEARS THE 'BREAK' BIT
2313                          ;;**********************************************************************
2314  005006  000004  TST21:  SCOPE
2315  005010  012704  000200          MOV     #200,R4         ;RESULT S/B = 200
2316  005014  016702  174374          MOV     DLXCSR,R2       ;SET UP REGADR
```

```
 2317  005020  052712  000001            BIS     #BIT0,(R2)      ;SET THE 'BREAK'' BIT
 2318  005024  000005                    RESET                   ;CLEAR IT WITH A 'RESET''
 2319  005026  020412                    CMP     R4,(R2)         ;DID IT CLEAR ??
 2320  005030  001403                    BEQ     TST22           ;;<BR IF YES>
 2321  005032  004767  007562            JSR     PC,SUER2        ;GO SET UP ERROR INFO.
 2322  005036  104002                    ERROR+2                 ;RESET INSTR. FAILED TO CLEAR 'BREAK''
 2323
```

```
                        ;;*****************************************************************
                        ;*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$:     TST     XFLG0           ;ANY HARD XMIT ERRORS ??
005076  001040                  BNE     5$              ;BR IF YES
005100  005767  174320          TST     RFLG0           ;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
```

```
2380  005330  001253                          BNE     1$              ;BR IF NOT
2381
2382
2383                                  ;;*****************************************************************
2384                                  ;*TEST 23        TEST TO TURN AROUND BINARY UP COUNT PATTERN
2385                                  ;;*****************************************************************
2386  005332  000004                  TST23:  SCOPE
2387  005334  012767  000001  173700          MOV     #1,$TIMES       ;;DO 1 ITERATION
2388  005342  004767  007406                  JSR     PC,SUVEC        ;GO SET UP VECTORS
2389  005346  005067  174056                  CLR     RTRY            ;INITIALIZE RETRY FLAG
2390  005352  012767  015152  174056  1$:     MOV     #LDOUT2,LDOUT   ;SET POINTER TO LOAD ROUTINE
2391  005360  004767  007416                  JSR     PC,PRIME        ;GO SET UP BUFFERS AND DEVICE
2392  005364  005767  174032          2$:     TST     XFLG0           ;ANY HARD XMIT ERRORS ??
2393  005370  001040                          BNE     5$              ;BR IF YES
2394  005372  005767  174026                  TST     RFLG0           ;ANY HARD RECEIVER ERROR ??
2395  005376  001053                          BNE     7$              ;BR IF YES
2396  005400  005767  174022                  TST     RFLG1           ;ANY SOFT RECEIVER ERRORS ??
2397  005404  001065                          BNE     9$              ;BR IF YES
2398  005406  022767  023040  174020          CMP     #BUFEND,IPTR    ;RECEIVED 256. BYTES ??
2399  005414  001003                          BNE     3$              ;BR IF NOT
2400  005416  004767  007652                  JSR     PC,CHKDAT       ;GO CHECK THE DATA BUFFERS
2401  005422  000500                          BR      TST24           ;;<GO TO NEXT TEST>
2402  005424  005367  174010          3$:     DEC     TIMR1           ;DEC TIMEOUT COUNTER 1
2403  005430  001355                          BNE     2$              ;BR IF NO TIMEOUT
2404  005432  005367  174004                  DEC     TIMR2           ;DEC TIMEOUT COUNTER 2
2405  005436  001352                          BNE     2$              ;BR IF NO TIMEOUT
2406  005440  042777  000100  173742          BIC     #100,@DLRCSR    ;TURN OFF THE INTRS.
2407  005446  042777  000104  173740          BIC     #104,@DLXCSR
2408  005454  104401  017167                  TYPE    ,XMSG2          ;GO TYPE TIMEOUT MESSAGE
2409  005460  012767  005470  173556          MOV     #4$,$ESCAPE     ;GO TO 4$ AFTER ERROR PRINT
2410  005466  104000                          ERROR                   ;PRINT ERROR PC
2411  005470                          4$:
2412  005470  000455                          BR      TST24           ;;<GO TO NEXT TEST>
2413  005472  016701  173712          5$:     MOV     DLRCSR,R1       ;PUT DEVADR IN R1
2414  005476  016702  173712                  MOV     DLXCSR,R2       ;PUT REGADR IN R2
2415  005502  011203                          MOV     (R2),R3         ;GET THE WAS DATA
2416  005504  012704  000204                  MOV     #204,R4         ;PUT S/B DATA IN R4
2417  005510  004767  007132                  JSR     PC,SUERR1       ;GO SET UP ERROR INFO
2418  005514  012767  005524  173522          MOV     #6$,$ESCAPE     ;GO TO 6$ AFTER PRINTING ERROR
2419  005522  104002                          ERROR+2                 ;TRANSMITTER FALSE INTERRUPT
2420  005524                          6$:
2421  005524  000437                          BR      TST24           ;;<GO TO NEXT TEST>
2422  005526  016701  173656          7$:     MOV     DLRCSR,R1       ;SAVE THE DEVADR
2423  005532  010102                          MOV     R1,R2           ;SAVE THE REGADR
2424  005534  011203                          MOV     (R2),R3         ;GET THE WAS DATA
2425  005536  012704  000200                  MOV     #200,R4         ;RESULT S/B = 200
2426  005542  004767  007100                  JSR     PC,SUERR1       ;GO SET UP ERROR INFO
2427  005546  012767  005556  173470          MOV     #8$,$ESCAPE     ;GO TO 8$ AFTER ERROR PRINT
2428  005554  104002                          ERROR+2                 ;RECEIVER FALSE INTERRUPT
2429  005556                          8$:
2430  005556  000422                          BR      TST24           ;;<GO TO NEXT TEST>
2431  005560  016701  173624          9$:     MOV     DLRCSR,R1       ;SAVE THE DEVADR
2432  005564  016702  173622                  MOV     DLRDBR,R2       ;SAVE REGADR
2433  005570  016703  173410                  MOV     $TMP1,R3        ;GET CONTENTS OF ERROR RDBR
2434  005574  004767  007046                  JSR     PC,SUERR1       ;GO SETUP ERROR INFO
2435  005600  012767  005610  173436          MOV     #10$,$ESCAPE    ;GO TO 10$ AFTER ERROR PRINT
```

```
2436  005606  104005                        ERROR+5                 ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN
2437  005610  005267  173614          10$:   INC     RTRY            ;COUNT ONE TRY
2438  005614  022767  000003  173606        CMP     #3,RTRY         ;TRIED THREE TIMES
2439  005622  001253                        BNE     1$              ;BR IF NOT
2440                                ;;*********************************************************************
2441                                ;*TEST 24        TEST TO TURN AROUND BINARY DOWN COUNT PATTERN
2442                                ;;*********************************************************************
2443  005624  000004                TST24:  SCOPE
2444  005626  012767  000001  173406        MOV     #1,$TIMES       ;;DO 1 ITERATION
2445  005634  004767  007114                JSR     PC,SUVEC        ;GO SET UP VECTORS
2446  005640  005067  173564                CLR     RTRY            ;INITIALIZE RETRY FLAG
2447  005644  012767  015172  173564  1$:   MOV     #LDOUT3,LDOUT   ;SET POINTER TO LOAD ROUTINE
2448  005652  004767  007124                JSR     PC,PRIME        ;GO SET UP BUFFERS AND DEVICE
2449  005656  005767  173540          2$:   TST     XFLG0           ;ANY HARD XMIT ERRORS ??
2450  005662  001040                        BNE     5$              ;BR IF YES
2451  005664  005767  173534                TST     RFLG0           ;ANY HARD RECEIVER ERROR ??
2452  005670  001053                        BNE     7$              ;BR IF YES
2453  005672  005767  173530                TST     RFLG1           ;ANY SOFT RECEIVER ERRORS ??
2454  005676  001065                        BNE     9$              ;BR IF YES
2455  005700  022767  023040  173526        CMP     #BUFEND,IPTR    ;RECEIVED 256. BYTES ??
2456  005706  001003                        BNE     3$              ;BR IF NOT
2457  005710  004767  007360                JSR     PC,CHKDAT       ;GO CHECK THE DATA BUFFERS
2458  005714  000500                        BR      TST25           ;;<GO TO NEXT TEST>
2459  005716  005367  173516          3$:   DEC     TIMR1           ;DEC TIMEOUT COUNTER 1
2460  005722  001355                        BNE     2$              ;BR IF NO TIMEOUT
2461  005724  005367  173512                DEC     TIMR2           ;DEC TIMEOUT COUNTER 2
2462  005730  001352                        BNE     2$              ;BR IF NO TIMEOUT
2463  005732  042777  000100  173450        BIC     #100,@DLRCSR    ;TURN OFF THE INTRS.
2464  005740  042777  000104  173446        BIC     #104,@DLXCSR
2465  005746  104401  017250                TYPE    ,XMSG3          ;GO TYPE TIMEOUT MESSAGE
2466  005752  012767  005762  173264        MOV     #4$,$ESCAPE     ;GO TO 4$ AFTER ERROR PRINT
2467  005760  104000                        ERROR                   ;PRINT ERROR PC
2468  005762                          4$:
2469  005762  000455                        BR      TST25           ;;<GO TO NEXT TEST>
2470  005764  016701  173420          5$:   MOV     DLRCSR,R1       ;PUT DEVADR IN R1
2471  005770  016702  173420                MOV     DLXCSR,R2       ;PUT REGADR IN R2
2472  005774  011203                        MOV     (R2),R3         ;GET THE WAS DATA
2473  005776  012704  000204                MOV     #204,R4         ;PUT S/B DATA IN R4
2474  006002  004767  006640                JSR     PC,SUERR1       ;GO SET UP ERROR INFO
2475  006006  012767  006016  173230        MOV     #6$,$ESCAPE     ;GO TO 6$ AFTER PRINTING ERROR
2476  006014  104002                        ERROR+2                 ;TRANSMITTER FALSE INTERRUPT
2477  006016                          6$:
2478  006016  000437                        BR      TST25           ;;<GO TO NEXT TEST>
2479  006020  016701  173364          7$:   MOV     DLRCSR,R1       ;SAVE THE DEVADR
2480  006024  010102                        MOV     R1,R2           ;SAVE THE REGADR
2481  006026  011203                        MOV     (R2),R3         ;GET THE WAS DATA
2482  006030  012704  000200                MOV     #200,R4         ;RESULT S/B = 200
2483  006034  004767  006606                JSR     PC,SUERR1       ;GO SET UP ERROR INFO
2484  006040  012767  006050  173176        MOV     #8$,$ESCAPE     ;GO TO 8$ AFTER ERROR PRINT
2485  006046  104002                        ERROR+2                 ;RECEIVER FALSE INTERRUPT
2486  006050                          8$:
2487  006050  000422                        BR      TST25           ;;<GO TO NEXT TEST>
2488  006052  016701  173332          9$:   MOV     DLRCSR,R1       ;SAVE THE DEVADR
2489  006056  016702  173330                MOV     DLRDBR,R2       ;SAVE REGADR
2490  006062  016703  173116                MOV     $TMP1,R3        ;GET CONTENTS OF ERROR RDBR
2491  006066  004767  006554                JSR     PC,SUERR1       ;GO SETUP ERROR INFO
```

```
2492  006072  012767  006102  173144          MOV     #10$,$ESCAPE    ;GO TO 10$ AFTER ERROR PRINT
2493  006100  104005                          ERROR+5                 ;REPORT SOFT ERROR (PARITY,FRAMING  OR OVERRUN
2494  006102  005267  173322          10$:    INC     RTRY            ;COUNT ONE TRY
2495  006106  022767  000003  173314          CMP     #3,RTRY         ;TRIED THREE TIMES
2496  006114  001253                          BNE     1$              ;BR IF NOT
2497                                          ;;*****************************************************************
2498                                          ;*TEST 25       TEST TO TURN AROUND WORST CASE PATTERN
2499                                          ;;*****************************************************************
2500  006116  000004                  TST25:  SCOPE
2501  006120  012767  000001  173114          MOV     #1,$TIMES       ;;DO 1 ITERATION
2502  006126  004767  006622                  JSR     PC,SUVEC        ;GO SET UP VECTORS
2503  006132  005067  173272                  CLR     RTRY            ;INITIALIZE RETRY FLAG
2504  006136  012767  015226  173272  1$:     MOV     #LDOUT4,LDOUT   ;SET POINTER TO LOAD ROUTINE
2505  006144  004767  006632                  JSR     PC,PRIME        ;GO SET UP BUFFERS AND DEVICE
2506  006150  005767  173246          2$:     TST     XFLGO           ;ANY HARD XMIT ERRORS ??
2507  006154  001042                          BNE     5$              ;BR IF YES
2508  006156  005767  173242                  TST     RFLGO           ;ANY HARD RECEIVER ERROR ??
2509  006162  001056                          BNE     7$              ;BR IF YES
2510  006164  005767  173236                  TST     RFLG1           ;ANY SOFT RECEIVER ERRORS ??
2511  006170  001071                          BNE     9$              ;BR IF YES
2512  006172  022767  023040  173234          CMP     #BUFEND,IPTR    ;RECEIVED 256. BYTES ??
2513  006200  001004                          BNE     3$              ;BR IF NOT
2514  006202  004767  007066                  JSR     PC,CHKDAT       ;GO CHECK THE DATA BUFFERS
2515  006206  000167  001642                  JMP             $EOP            ;GO TO NEXT TEST
2516  006212  005367  173222          3$:     DEC     TIMR1           ;DEC TIMEOUT COUNTER 1
2517  006216  001354                          BNE     2$              ;BR IF NO TIMEOUT
2518  006220  005367  173216                  DEC     TIMR2           ;DEC TIMEOUT COUNTER 2
2519  006224  001351                          BNE     2$              ;BR IF NO TIMEOUT
2520  006226  042777  000100  173154          BIC     #100,@DLRCSR    ;TURN OFF THE INTRS.
2521  006234  042777  000104  173152          BIC     #104,@DLXCSR
2522  006242  104401  017333                  TYPE    ,XMSG4          ;GO TYPE TIMEOUT MESSAGE
2523  006246  012767  006256  172770          MOV     #4$,$ESCAPE     ;GO TO 4$ AFTER ERROR PRINT
2524  006254  104000                          ERROR                   ;PRINT ERROR PC
2525  006256  000167  001572          4$:     JMP     $EOP            ;GO TO NEXT TEST
2526  006262  016701  173122          5$:     MOV     DLRCSR,R1       ;PUT DEVADR IN R1
2527  006266  016702  173122                  MOV     DLXCSR,R2       ;PUT REGADR IN R2
2528  006272  011203                          MOV     (R2),R3         ;GET THE WAS DATA
2529  006274  012704  000204                  MOV     #204,R4         ;PUT S/B DATA IN R4
2530  006300  004767  006342                  JSR     PC,SUERR1       ;GO SET UP ERROR INFO
2531  006304  012767  006314  172732          MOV     #6$,$ESCAPE     ;GO TO 6$ AFTER PRINTING ERROR
2532  006312  104002                          ERROR+2                 ;TRANSMITTER FALSE INTERRUPT
2533  006314  000167  001534          6$:     JMP     $EOP            ;GO TO NEXT TEST
2534  006320  016701  173064          7$:     MOV     DLRCSR,R1       ;SAVE THE DEVADR
2535  006324  010102                          MOV     R1,R2           ;SAVE THE REGADR
2536  006326  011203                          MOV     (R2),R3         ;GET THE WAS DATA
2537  006330  012704  000200                  MOV     #200,R4         ;RESULT S/B = 200
2538  006334  004767  006306                  JSR     PC,SUERR1       ;GO SET UP ERROR INFO
2539  006340  012767  006350  172676          MOV     #8$,$ESCAPE     ;GO TO 8$ AFTER ERROR PRINT
2540  006346  104002                          ERROR+2                 ;RECEIVER FALSE INTERRUPT
2541  006350  000167  001500          8$:     JMP     $EOP            ;GO TO NEXT TEST
2542  006354  016701  173030          9$:     MOV     DLRCSR,R1       ;SAVE THE DEVADR
2543  006360  016702  173026                  MOV     DLRDBR,R2       ;SAVE REGADR
2544  006364  016703  172614                  MOV     $TMP1,R3        ;GET CONTENTS OF ERROR RDBR
2545  006370  004767  006252                  JSR     PC,SUERR1       ;GO SETUP ERROR INFO
2546  006374  012767  006404  172642          MOV     #10$,$ESCAPE    ;GO TO 10$ AFTER ERROR PRINT
2547  006402  104005                          ERROR+5                 ;REPORT SOFT ERROR (PARITY,FRAMING, OR OVERRUN
```

```
2548  006404  005267  173020          10$:    INC     RTRY            ;COUNT ONE TRY
2549  006410  022767  000003  173012          CMP     #3,RTRY         ;TRIED THREE TIMES
2550  006416  001247                          BNE     1$              ;BR IF NOT
2551  006420  000167  001430                  JMP     $EOP            ;GO TO END OF PASS ROUTINE
2552
2553                                  ;THIS IS PROGRAM #2
2554                                  ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
2555                                  ;       A)  SELECTION OF A TRANSMITTER DATA BUFFER
2556                                  ;       B)  SELECTION OF A CHARACTER FOR CONTINUOUS TRANSFER
2557                                  ;       C)  SELECTION OF AN EXPIRATION TIME IN MILLISECONDS
2558                                  ;           BETWEEN EACH TRANSMITTER DATA BUFFER CHARACTER TRANSFER
2559                                  ;       D)  A TIGHT SCOPE LOOP LOCK ON A SPECIFIC CHARACTER
2560                                  ;
2561
2562  006424  000240                  PRG2:   NOP
2563  006426  104401  017460                  TYPE    ,PROG2M         ;INDICATE THAT USER SELECTED
2564                                                                  ;PROGRAM #2
2565  006432  104401  021102          PRG2A:  TYPE    ,LINTAD         ;ASK USER FOR THE TRANSMITTER
2566                                                                  ;DATA BUFFER ADDRESS OF THE DEVICE
2567                                                                  ;HE WISHES TO TEST
2568  006436  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
2569                                                                  ;AND STORE ON TOP OF STACK
2570                                  ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
2571  006440  012602                          MOV     (SP)+,R2        ;GET THE ANSWER TYPED
2572  006442  020227  176176                  CMP     R2,#176176      ;IS THE NUMBER TOO HIGH?
2573  006446  101065                          BHI     REDO1           ;IF YES - GO TO RETRY SITUATION
2574  006450  020227  175616                  CMP     R2,#175616      ;IS THE NUMBER TOO LOW?
2575  006454  103462                          BLO     REDO1           ;IF YES - GO TO RETRY SITUATION
2576  006456  132702  000001                  BITB    #BIT0,R2        ;NUMBER IS IN RANGE BUT IS IT
2577                                                                  ;ON AN EVEN BOUNDARY?
2578  006462  001057                          BNE     REDO1           ;IF NOT GO TO RETRY SITUATION
2579                                  ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
2580  006464  010203                          MOV     R2,R3           ;GET THE USER RESPONSE
2581  006466  142703  000370                  BICB    #370,R3         ;MASK OFF LOWER BYTE EXCEPT FOR
2582                                                                  ;LEAST SIGNIFICANT DIGIT
2583  006472  122703  000006                  CMPB    #6,R3           ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
2584                                                                  ;USER RESPONSE EQUAL TO A SIX?
2585  006476  001051                          BNE     REDO1           ;BRANCH IF NOT
2586  006500  010267  172476                  MOV     R2,$TMP0        ;THE TRANSMITTER ADDRESS
2587                                                                  ;TYPED IS OK - STORE FOR
2588                                                                  ;FUTURE USE
2589                                  ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
2590  006504  016746  171274                  MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
2591  006510  012767  006522  171266          MOV     #2$,ERRVEC      ;SET UP TIMEOUT SERVICE ADDRESS
2592  006516  005712                          TST     (R2)            ;IF PRESENT WE WILL EXECUTE THE
2593                                                                  ;NEXT INSTRUCTION - IF NOT
2594                                                                  ;WE GO TO 2$:
2595  006520  000412                          BR      4$              ;BRANCH IF PRESENT
2596  006522  004767  006200          2$:     JSR     PC,SUERT2       ;GO SET UP FOR ERROR INFORMATION
2597  006526  012767  006536  172510          MOV     #3$,$ESCAPE     ;POINT OF RETURN AFTER ERROR REPORT
2598  006534  104006                          ERROR   +6              ;XDBR REFERENCE CAUSED TIMEOUT
2599  006536  022626                  3$:     CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
2600  006540  012667  171240                  MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
2601  006544  000426                          BR      REDO1           ;GO TO RETRY SITUATION
2602  006546  012667  171232          4$:     MOV     (SP)+,ERRVEC    ;DEVICE REGISTER IS PRESENT!
2603                                                                  ;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
006556  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY
                                                                ;USER AND STORE ON TOP OF STACK
006560  012667  172420                  MOV     (SP)+,$TMP1     ;GET THE ANSWER TYPED
                                ;NOTE:  THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE
                                ;       OCTAL ASCII EQUIVALENT OF THE CHARACTER E.G. A=101
006564  104401  021271                  TYPE    ,SELDLY         ;ASK THE USER FOR THE DELAY
                                                                ;IN MSEC (OCTAL NO.) BETWEEN
                                                                ;CHARACTER TRANSFERS
006570  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY
                                                                ;USER AND STORE ON TOP OF STACK
006572  012667  172410                  MOV     (SP)+,$TMP2     ;GET THE ANSWER TYPED
006576  116767  172404  000012  1$:     MOVB    $TMP2,2$        ;SET THE DELAY COUNT ARGUMENT
                                                                ;FOR TIMER ROUTINE
006604  116777  172374  172370          MOVB    $TMP1,@$TMP0    ;LOAD THE TRANSMITTER DATA
                                                                ;BUFFER WITH THE CHARACTER
006612  004767  005470                  JSR     PC,DELAY        ;GO OFF TO WAIT THE SPECIFIED
                                                                ;NO. OF MSEC. BEFORE ISSUING
                                                                ;ANOTHER CHARACTER
006616  000000          2$:     .WORD   0                       ;THIS IS WHERE THE DELAY COUNT RESIDES
006620  000766                  BR      1$                      ;GO BACK TO ISSUE ANOTHER CHARACTER
006622  104401  001252  REDO1:  TYPE    ,$QUES                  ;TYPE A QUESTION MARK(?)
006626  000167  177600          JMP     PRG2A                   ;REITERATE THE XDBR QUESTION TO USER
                                ;THIS IS PROGRAM #3
                                ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW:
                                ;       A)  SELECTION OF A TRANSMITTER DATA BUFFER
                                ;       B)  SELECTION OF A CHARACTER FOR CONTINUOUS TRANSFER
                                ;           IN MAINTENANCE MODE
                                ;       C)  SELECTION OF AN EXPIRATION TIME IN MILLISECONDS
                                ;           BETWEEN EACH TRANSMITTER DATA BUFFER CHARACTER TRANSFER
                                ;       D)  A TIGHT SCOPE LOOP LOCK ON A SPECIFIC CHARACTER
                                ;
006632  000240          PRG3:   NOP
006634  104401  017524          TYPE    ,PROG3M                 ;INDICATE THAT USER SELECTED
                                                                ;PROGRAM #3
006640  104401  021102  PRG3A:  TYPE    ,LINTAD                 ;ASK USER FOR THE TRANSMITTER DATA
                                                                ;BUFFER ADDRESS OF THE DEVICE
                                                                ;HE WISHES TO TEST
006644  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY
                                                                ;USER AND STORE ON TOP OF STACK
                                ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
006646  012602                  MOV     (SP)+,R2                ;GET THE ANSWER TYPED
006650  020227  176176          CMP     R2,#176176              ;IS THE NUMBER TOO HIGH?
006654  101071                  BHI     REDO2                   ;IF YES - GO TO RETRY SITUATION
006656  020227  175616          CMP     R2,#175616              ;IS THE NUMBER TOO LOW?
006662  103466                  BLO     REDO2                   ;IF YES - GO TO RETRY SITUATION
```

```
2660  006664  132702  000001          BITB    #BIT0,R2        ;NUMBER IS IN RANGE BUT IS IT
2661                                                          ;ON AN EVEN BOUNDARY?
2662  006670  001063                  BNE     REDO2           ;IF NOT - GO TO RETRY SITUATION
2663                                ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XDBR DBR ADDRESS
2664  006672  010203                  MOV     R2,R3           ;GET THE USER RESPONSE
2665  006674  142703  000370          BICB    #370,R3         ;MASK OFF LOWER BYTE EXCEPT FOR
2666                                                          ;LEAST SIGNIFICANT DIGIT
2667  006700  122703  000006          CMPB    #6,R3           ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
2668                                                          ;USER RESPONSE EQUAL TO A TWO?
2669  006704  001055                  BNE     REDO2           ;BRANCH IF NOT
2670  006706  010267  172270          MOV     R2,$TMP0        ;THE TRANSMITTER ADDRESS TYPED IS
2671                                                          ;OK - STORE FOR FUTURE USE
2672                                ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
2673  006712  016746  171066          MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
2674  006716  012767  006730  171060  MOV     #2$,ERRVEC      ;SET UP TIMEOUT SERVICE ADDRESS
2675  006724  005712                  TST     (R2)            ;IF PRESENT WE WILL EXECUTE THE
2676                                                          ;NEXT INSTRUCTION - IF NOT WE
2677                                                          ;GO TO 2$:
2678  006726  000412                  BR      4$              ;BRANCH IF PRESENT
2679  006730  004767  005772  2$:     JSR     PC,SUERT2       ;GO SET UP FOR ERROR INFORMATION
2680  006734  012767  006744  172302  MOV     #3$,$ESCAPE     ;POINT OF RETURN AFTER ERROR REPORT
2681  006742  104006                  ERROR   +6              ;XDBR REFERENCE CAUSED TIMEOUT
2682  006744  022626          3$:     CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
2683  006746  012667  171032          MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
2684  006752  000432                  BR      REDO2           ;GO TO RETRY SITUATION
2685  006754  012667  171024  4$:     MOV     (SP)+,ERRVEC    ;DEVICE REGISTER IS PRESENT!
2686                                                          ;RESTORE TIMEOUT VECTOR
2687                                ;WE ARE NOW READY FOR THE CHARACTER TO BE TRANSMITTED, AND THE
2688                                ;DELAY TIME (IN MILLISECONDS) THAT IS TO TRANSPIRE BETWEEN SUCCESSIVE
2689                                ;CHARACTER TRANSFERS
2690  006760  104401  021163  PRG38: TYPE    ,SELCAR         ;ASK USER FOR THE CHARACTER
2691                                                          ;HE WISHES TO TRANSFER
2692  006764  104412                  RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
2693                                                          ;AND STORE ON TOP OF STACK
2694  006766  012667  172212          MOV     (SP)+,$TMP1     ;GET THE ANSWER TYPED
2695                                ;NOTE:  THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE
2696                                ;       OCTAL ASCII EQUIVALENT OF THE CHARACTER E.G. B=102
2697  006772  104401  021271          TYPE    ,SELDLY         ;ASK THE USER FOR THE DELAY
2698                                                          ;IN MSEC (OCTAL NO.) BETWEEN
2699                                                          ;CHARACTER TRANSFERS
2700  006776  104412                  RDOCT                   ;ACCEPT THE ANSWER TYPED BY
2701                                                          ;USER AND STORE ON TOP OF STACK
2702  007000  012667  172202          MOV     (SP)+,$TMP2     ;GET THE ANSWER TYPED
2703  007004  162702  000002          SUB     #2,R2           ;GET THE CORRESPONDING XCSR
2704                                                          ;ADDRESS FOR TRANSMITTER UNDER-
2705                                                          ;GOING TEST
2706  007010  052712  000004  1$:     BIS     #BIT2,(R2)      ;SET MAINTENANCE BIT IN XCSR
2707  007014  116767  172166  000012  MOVB    $TMP2,2$        ;SET THE DELAY COUNT ARGUMENT
2708                                                          ;FOR TIMER ROUTINE
2709  007022  116777  172156  172152  MOVB    $TMP1,@$TMP0    ;LOAD THE TRANSMITTER DATA BUFFER
2710                                                          ;WITH THE CHARACTER
2711  007030  004767  005252          JSR     PC,DELAY        ;GO OFF TO WAIT THE SPECIFIED
2712                                                          ;NO. OF MSEC. BEFORE ISSUING
2713                                                          ;ANOTHER CHARACTER
2714  007034  000000  2$:     .WORD   0               ;THIS IS WHERE THE DELAY COUNT RESIDES
2715  007036  000764          BR      1$              ;GO BACK TO ISSUE ANOTHER CHARACTER
```

```
                                REDO2:  TYPE    ,$QUES          ;TYPE A QUESTION MARK(?)
                                        JMP     PRG3A           ;REITERATE THE XDBR QUESTION TO
                                                                ;USER


                                ;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
                                ;


                                PRG4:   NOP
                                        TYPE    ,PROG4M         ;INDICATE THAT USER SELECTED
                                                                ;PROGRAM #4
                                PRG4A:  TYPE    ,LINTAD         ;ASK USER FOR THE TRANSMITTER
                                                                ;DATA BUFFER ADDRESS OF THE
                                                                ;DEVICE HE WISHES TO TEST
                                        RDOCT                   ;ACCEPT THE ANSWER TYPED BY
                                                                ;USER AND STORE ON TOP OF STACK
                                ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
                                        MOV     (SP)+,R2        ;GET THE ANSWER TYPED
                                        CMP     R2,#176176      ;IS THE NUMBER TOO HIGH?
                                        BHI     REDO3           ;IF YES - GO TO RETRY SITUATION
                                        CMP     R2,#175616      ;IS THE NUMBER TOO LOW?
                                        BLO     REDO3           ;IF YES - GO TO RETRY SITUATION
                                        BITB    #BIT0,R2        ;NUMBER IS IN RANGE BUT IS IT
                                                                ;ON AN EVEN BOUNDARY?
                                        BNE     REDO3           ;IF NO - GO TO RETRY SITUATION
                                ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
                                        MOV     R2,R3           ;GET THE USER RESPONSE
                                        BICB    #370,R3         ;MASK OFF LOWER BYTE EXCEPT FOR
                                                                ;LEAST SIGNIFICANT DIGIT
                                        CMPB    #6,R3           ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
                                                                ;USER RESPONSE EQUAL TO A SIX?
                                        BNE     REDO3           ;BRANCH IF NOT
                                        MOV     R2,$TMP0        ;THE TRANSMITTER ADDRESS TYPED
                                                                ;IS OK - STORE FOR FUTURE USE
                                ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
                                        MOV     ERRVEC,-(SP)    ;SAVE THE TIMEOUT VECTOR
                                        MOV     #2$,ERRVEC      ;SET UP TIMEOUT SERVICE ADDRESS
                                        TST     (R2)            ;IF PRESENT WE WILL EXECUTE THE
                                                                ;NEXT INSTRUCTION - IF NOT WE
                                                                ;GO TO 2$:
                                        BR      4$              ;BRANCH IF PRESENT
                                2$:     JSR     PC,SUERT2       ;GO SET UP FOR ERROR INFORMATION
                                        MOV     #3$,$ESCAPE     ;POINT OF RETURN AFTER ERROR REPORT
                                        ERROR   +6              ;XDBR REFERENCE CAUSED TIMEOUT
                                3$:     CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
                                        MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
                                        BR      REDO3           ;GO TO RETRY SITUATION
                                4$:     MOV     (SP)+,ERRVEC    ;DEVICE REGISTER IS PRESENT!
                                                                ;RESTORE TIMEOUT VECTOR
                                        TYPE    ,RSTALL         ;ASK THE USER IF HE DESIRES SOME
                                                                ;RANDOM NO. OF MSEC. WAIT TIME
                                                                ;BEFORE CHECKING FOR XCSR DONE
                                                                ;FLAG
```

```
                                        RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
                                                                ;AND STORE ON TOP OF STACK
                                        MOV     (SP)+,$TMP2     ;GET THE ANSWER TYPED
                                ;WE ARE NOW READY FOR THE CHARACTER TO BE TRANSMITTED
                                PRG4B:  TYPE    ,SELCAR         ;ASK USER FOR THE CHARACTER HE
                                                                ;WISHES TO TRANSFER
                                        RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
                                                                ;AND STORE ON TOP OF STACK
                                        MOV     (SP)+,$TMP1     ;GET THE ANSWER TYPED
                                ;NOTE:  THE USER RESPONSE FOR THE CHARACTER WAS TO BE THE OCTAL
                                ;       ASCII EQUIVALENT OF THE CHARACTER E.G. C=103
                                ;
                                PRG4C:  TYPE,   LENGTH  ;ASK USER FOR THE CHARACTER LENGTH
                                                        ;FOR WHICH HIS DEVICE IS SET
                                        RDDEC           ;ACCEPT THE ANSWER TYPED BY USER
                                ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
                                        MOV     (SP)+,R0        ;GET THE ANSWER TYPED
                                        CMP     R0,#8.          ;IS THE NUMBER TOO HIGH?
                                        BHI     REDO3A          ;IF YES - GO TO RETRY SITUATION
                                        CMP     R0,#5.          ;IS THE NUMBER TOO LOW?
                                        BLO     REDO3A          ;IF YES - GO TO RETRY SITUATION
                                        MOV     R0,$TMP15       ;THE VALUE TYPED IS OK
                                                                ;STORE 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              ;RCSR 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,$TMP5    ;STORE THE CHARACTER FROM THE
                                                                ;RECEIVER BUFFER & REST OF CONTENTS
                                        JSR     PC,DATCHK       ;GO TO COMPARE EXPECTED & RECEIVED
                                                                ;DATA
                                        BR      1$              ;GO BACK TO ISSUE ANOTHER CHARACTER
                                REDO3:  TYPE    ,$QUES          ;TYPE A QUESTION MARK(?)
                                        JMP     PRG4A           ;REITERATE THE XDBR QUESTION TO USER
                                REDO3A: TYPE,   $QUES           ;TYPE '?' INDICATING USER TYPED
                                                                ;SOMETHING WRONG FOR CHARACTER LENGTH
                                        JMP     PRG4C           ;GO BACK TO REISSUE QUESTION

                                ;THIS IS PROGRAM #5
```

```
                        ;THE FOLLOWING USER UTILITY PROGRAM WILL ALLOW USER PARAMETERS
                        ;FOR RUNNING A BINARY COUNT IN MAINTENANCE MODE
                        ;
007410  000240          PRG5:   NOP
007412  104401  017634          TYPE    ,PROG5M         ;INDICATE THAT USER SELECTED
                                                        ;PROGRAM #5
007416  104401  021102  PRG5A:  TYPE    ,LINTAD         ;ASK USER FOR THE TRANSMITTER DATA
                                                        ;BUFFER ADDRESS OF THE DEVICE
                                                        ;HE WISHES TO TEST
007422  104412                  RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
                                                        ;AND STORE ON TOP OF STACK
                        ;CHECK TO SEE IF THE USER RESPONSE WAS WITHIN LIMITS
007424  012602                  MOV     (SP)+,R2        ;GET THE ANSWER TYPED
007426  020227  176176          CMP     R2,#176176      ;IS THE NUMBER TOO HIGH?
007432  101152                  BHI     REDO4           ;IF YES - GO TO RETRY SITUATION
007434  020227  175616          CMP     R2,#175616      ;IS THE NUMBER TOO LOW?
007440  103547                  BLO     REDO4           ;IF YES - GO TO RETRY SITUATION
007442  132702  000001          BITB    #BIT0,R2        ;NUMBER IS IN RANGE BUT IS IT
                                                        ;ON AN EVEN BOUNDARY?
007446  001144                  BNE     REDO4           ;IF NOT - GO TO RETRY SITUATION
                        ;CHECK TO SEE IF USER RESPONSE WAS TRULY A XMIT BUFFER REGISTER
007450  010203                  MOV     R2,R3           ;GET THE USER RESPONSE
007452  142703  000370          BICB    #370,R3 ;MASK OFF LOWER BYTE EXCEPT FOR
                                                        ;LEAST SIGNIFICANT DIGIT
007456  122703  000006          CMPB    #6,R3           ;WAS THE LEAST SIGNIFICANT DIGIT OF THE
                                                        ;USER RESPONSE EQUAL TO A SIX?
007462  001136                  BNE     REDO4           ;BRANCH IF NOT
007464  010267  171512          MOV     R2,$TMP0        ;THE TRANSMITTER ADDRESS TYPED
                                                        ;IS OK - STORE FOR FUTURE USE
                        ;NOW CHECK TO MAKE SURE THE DEVICE IS PRESENT
007470  016746  170310          MOV     ERRVEC,-(SP)    ;SAVE THE TIMOUT VECTOR
007474  012767  007506  170302  MOV     #2$,ERRVEC      ;SET UP TIMEOUT SERVICE ADDRESS
007502  005712                  TST     (R2)            ;IF PRESENT WE WILL EXECUTE THE
                                                        ;NEXT INSTRUCTION - IF NOT WE
                                                        ;GO TO 2$:
007504  000412                  BR      4$              ;BRANCH IF PRESENT
007506  004767  005214  2$:     JSR     PC,SUERT2       ;GO SETUP FOR ERROR INFORMATION
007512  012767  007522  171524  MOV     #3$,$ESCAPE     ;POINT OF RETURN AFTER ERROR REPORT
007520  104006                  ERROR   +6              ;XDBR REFERENCE CAUSED TIMEOUT
007522  022626          3$:     CMP     (SP)+,(SP)+     ;CLEAN STACK FROM TIMEOUT
007524  012667  170254          MOV     (SP)+,ERRVEC    ;RESTORE TIMEOUT VECTOR
007530  000513                  BR      REDO4           ;GO TO RETRY SITUATION
007532  012667  170246  4$:     MOV     (SP)+,ERRVEC    ;DEVICE REGISTER IS PRESENT!
                                                        ;ASK THE USER IF HE DESIRES SOME
                                                        ;RANDOM NO. OF MSEC. WAIT TIME
                                                        ;BEFORE CHECKING XCSR DONE FLAG
007536  104401  020013  PRG5C:  TYPE,   LENGTH  ;ASK USER FOR THE CHARACTER LENGTH
                                                        ;FOR WHICH HIS DEVICE IS SET
007542  104413                  RDDEC                   ;ACCEPT THE ANSWER TYPED BY USER
                        ;CHECK TO SEE IF USER RESPONSE WAS WITHIN LIMITS
007544  012600                  MOV     (SP)+,R0        ;GET THE ANSWER TYPED
007546  020027  000010          CMP     R0,#8.          ;IS THE NUMBER TOO HIGH?
007552  101106                  BHI     REDO4A          ;IF YES - GO TO RETRY SITUATION
007554  020027  000005          CMP     R0,#5.          ;IS THE NUMBER TOO LOW?
007560  103503                  BLO     REDO4A          ;IF YES - GO TO RETRY SITUATION
```

```
2884  007562  010067  171446                  MOV     R0,$TMP15       ;THE VALUE TYPED IS OK
2885                                                                    ;STORE FOR FUTURE USE
2886  007566  104401  021364                  TYPE    ,RSTALL         ;RANDOM NO. OF MSEC. WAIT TIME
2887  007572  104412                          RDOCT                   ;ACCEPT THE ANSWER TYPED BY USER
2888                                                                    ;AND STORE ON TOP OF STACK
2889  007574  012667  171406                  MOV     (SP)+,$TMP2     ;GET THE ANSWER TYPED
2890                        ;WE ARE NOW READY TO INITIALIZE THE BINARY COUNT AND GET
2891                        ;THE BINARY CHARACTER
2892                        ;
2893  007600  012767  177777  171416          MOV     #-1,$TMP11      ;SET LEAD IN VARIABLE TO -1
2894  007606  016767  171412  171412  PRG5B:  MOV     $TMP11,$TMP12   ;STORE PREVIOUS BINARY CHARACTER
2895  007614  005267  171406                  INC     $TMP12          ;FLIP BINARY CHARACTER AGAIN
2896  007620  042767  177400  171400          BIC     #177400,$TMP12  ;MASK TO 8 BITS
2897  007626  016767  171374  171370          MOV     $TMP12,$TMP11   ;STORE BINARY CHARACTER
2898  007634  016767  171366  171342          MOV     $TMP12,$TMP1    ;STORE BINARY CHARACTER
2899                                                                    ;FOR FUTURE USE
2900  007642  016767  171334  171340          MOV     $TMP0,$TMP3     ;GET THE XDBR ADDRESS
2901  007650  162767  000002  171332          SUB     #2,$TMP3        ;FORM THE XCSR ADDRESS
2902  007656  005767  171324          1$:     TST     $TMP2           ;DO WE RANDOM STALL?
2903  007662  001402                          BEQ     2$              ;BRANCH IF IT WASN'T DESIRED
2904  007664  004767  004462                  JSR     PC,STALL        ;GO STALL RANDOM VALUE OF MSEC.
2905  007670  004767  004566          2$:     JSR     PC,TIMETX       ;GO WAIT FOR TRANSMITTER DONE
2906                                                                    ;BIT TO SET
2907  007674  104401  017700                  TYPE    ,XDB            ;TYPE TRANSMITTER DONE BIT MESSAGE
2908  007700  104000                          ERROR   +0              ;XCSR DONE BIT NEVER SET
2909  007702  052777  000004  171300          BIS     #BIT2,@$TMP3    ;SET THE MAINTENANCE BIT IN THE
2910                                                                    ;TRANSMITTER CONTROL STATUS REGISTER
2911  007710  016777  171270  171264          MOV     $TMP1,@$TMP0    ;LOAD TRANSMITTER DATA BUFFER
2912                                                                    ;WITH SELECTED CHARACTER
2913  007716  004767  004522                  JSR     PC,TIMERX       ;GO WAIT FOR RECEIVER DONE BIT TO SET
2914  007722  104401  017747                  TYPE    ,RDB            ;TYPE RECEIVER DONE BIT MESSAGE
2915  007726  104000                          ERROR   +0              ;RCSR DONE BIT NEVER SET
2916  007730  016767  171254  171254          MOV     $TMP3,$TMP4     ;GET THE TRANSMITTER CONTROL
2917                                                                    ;STATUS REGISTER ADDRESS
2918  007736  162767  000002  171246          SUB     #2,$TMP4        ;FORM THE RECEIVER DATA BUFFER
2919                                                                    ;ADDRESS
2920  007744  017767  171242  171242          MOV     @$TMP4,$TMP5    ;STORE THE CHARACTER FROM THE
2921                                                                    ;RECEIVER BUFFER + REST OF CONTENTS
2922  007752  004767  004542                  JSR     PC,DATCHK       ;GO TO COMPARE EXPECTED &  RECEIVED
2923                                                                    ;DATA
2924  007756  000713                          BR      PRG5B           ;GO BACK TO ISSUE ANOTHER BINARY
2925                                                                    ;CHARACTER
2926  007760  104401  001252          RED04:  TYPE    ,$QUES          ;TYPE A QUESTION MARK(?)
2927  007764  000167  177426                  JMP     PRG5A           ;GO BACK TO REITERATE XDBR QUESTION
2928  007770  104401  001252          RED04A: TYPE,   $QUES           ;TYPE '?' INDICATING USER TYPED
2929                                                                    ;SOMETHING WRONG FOR CHARACTER LENGTH
2930  007774  000167  177536                  JMP     PRG5C           ;GO BACK TO REISSUE QUESTION
2931
2932                        ;THIS ROUTINE WILL SET UP:
2933                        ;       RCSR - RECEVER STATUS REGISTER
2934                        ;       RBUF - RECEIVER BUFFER REGISTER
2935                        ;       XCSR - TRANSMITTER STATUS REGISTER
2936                        ;       XBUF - TRANSMITTER BUFFER REGISTER
2937                        ;INITIALLY, IN RESPONSE TO USER REPLY TO 1ST DEVICE HE WANTS
2938                        ;TESTED, AND THEREAFTER, AT THE END OF A PROGRAM TO CYCLE THRU
2939                        ;ALL DEVICES FOR MULTIPLE DEVICE TESTING (IF REQUIRED)
```

```
2940  010000  016767  171254  171402  DLADDR: MOV     DLBASE,DLRCSR   ;STORE RECEIVER STATUS REGISTER
2941                                                                  ;OF CURRENT DEVICE
2942  010006  062767  000002  171244          ADD     #2,DLBASE       ;FORM RECEIVER BUFFER REGISTER
2943                                                                  ;OF CURRENT DEVICE
2944  010014  016767  171240  171370          MOV     DLBASE,DLRDBR   ;STORE RECEIVER BUFFER REGISTER
2945                                                                  ;OF CURRENT DEVICE
2946  010022  062767  000002  171230          ADD     #2,DLBASE       ;FORM TRANSMITTER STATUS REGISTER
2947                                                                  ;OF OF CURRENT DEVICE
2948  010030  016767  171224  171356          MOV     DLBASE,DLXCSR   ;STORE TRANSMITTER STATUS REGISTER
2949                                                                  ;OF CURRENT DEVICE
2950  010036  062767  000002  171214          ADD     #2,DLBASE       ;FOR TRANSMITTER BUFFER REGISTER
2951                                                                  ;OF CURRENT DEVICE
2952  010044  016767  171210  171344          MOV     DLBASE,DLXDBR   ;STORE TRANSMITTER BUFFER REGISTER
2953                                                                  ;OF CURRENT DEVICE
2954  010052  000207                          RTS     PC              ;RETURN
2955
2956                                  .SBTTL  END OF PASS ROUTINE
2957
2958                                  ;;*******************************************************************
2959                                  ;*INCREMENT THE PASS NUMBER ($PASS)
2960                                  ;*TYPE 'END PASS #XXXXX'' (WHERE XXXXX IS A DECIMAL NUMBER)
2961                                  ;*IF THERES A MONITOR GO TO IT
2962                                  ;*IF THERE ISN'T JUMP TO RESTRT
2963
2964  010054                          $EOP:
2965                                  ;THIS NEXT SECTION UP TO THE NEXT LINE OF ASTERISKS WAS
2966                                  ;SUPPLIED BY THE MACRO 'EOPBEG'.  THE MACRO NAME APPEARS IN
2967                                  ;THE SOURCE PROGRAM AS ONE OF THE ARGUMENTS TO THE .SEOP
2968                                  ;SYSMAC UTILITY ROUTINE CALL.
2969  010054  000004                          SCOPE
2970  010056  105767  171210                  TSTB    MULTD           ;ARE WE RUNNING MULTIPLE DEVICES?
2971  010062  001501                          BEQ     3$              ;BRANCH IF NOT FOR NORMAL
2972                                                                  ;'END PASS # XX' TYPEOUT
2973  010064  005767  171204                  TST     ACTREG          ;ARE ANY DEVICES ACTIVE?
2974  010070  001011                          BNE     1$              ;BRANCH IF YES
2975  010072  104401  020670                  TYPE    ,FOULUP         ;INDICATE SOMETHING WRONG!
2976                                                                  ;MULTIPLE DEVICES ARE BEING
2977                                                                  ;RUN SUPPOSEDLY, BUT NONE ARE
2978                                                                  ;SHOWN ACTIVE
2979  010076  000000                          HALT                    ;WAIT FOR A USER RESPONSE
2980  010100  005067  171166                  CLR     MULTD           ;CLEAR MULTIPLE DEVICE FLAG
2981  010104  005067  171146                  CLR     TABFLG          ;CLEAR TABLE CREATION FLAG
2982  010110  000167  171722                  JMP     RESTRT          ;GET READY TO START ALL OVER
2983                                                                  ;AGAIN - ALL DEVICES WERE
2984                                                                  ;DESELECTED SOMEHOW!
2985  010114  062767  000010  171142  1$:     ADD     #10,BASEADD     ;FORM A NEW BASE
2986                                                                  ;ADDRESS FOR START OF NEXT BLOCK
2987                                                                  ;OF REGISTERS FOR NEXT DEVICE
2988  010122  062767  000010  171140          ADD     #10,BASEIV      ;FORM A NEW BASE ADDRESS FOR
2989                                                                  ;START OF NEXT BLOCK OF VECTORS
2990                                                                  ;FOR NEXT DEVICE
2991  010130  000241                          CLC                     ;CLEAR LAST DEVICE INDICATOR
2992  010132  006167  171140                  ROL     ROTADD          ;UPDATE NEXT POSSIBLE DEVICE ACTIVE
2993                                                                  ;POINTER
2994  010136  103431                          BCS     2$              ;BRANCH IF THIS WAS THE
2995                                                                  ;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
010150  016767  171110  171102          MOV     BASEADD,DLBASE  ;FORM THE RECEIVER STATUS REGISTER
                                                        ;ADDRESS OF NEXT DEVICE
010156  016767  171106  171234          MOV     BASEIV,DLVECT   ;GET NEXT DEVICE RCVR VECTOR
010164  000240                          NOP
010166  004767  177606                  JSR     PC,DLADDR       ;GO FORM DL ADDRESSES FOR NEXT
                                                        ;DEVICE SELECTED
010172  005067  170704                  CLR     $TSTNM          ;INITIALIZE TEST NO. FOR A PROGRAM
                                                        ;PASS OVER THE NEXT DEVICE ACTIVE
010176  005077  171212                  CLR     @DLXCSR         ;CLEAR OUT BOTH CSR'S
010202  005077  171202                  CLR     @DLRCSR
010206  005777  171200                  TST     @DLRDBR         ;FLUSH RCVR 'DONE'' BIT
010212  005777  171174                  TST     @DLRDBR
010216  000167  172612                  JMP     TST1            ;START TESTING THIS DEVICE
010222                          2$:
                                ;IF WE TAKE THIS PATH WE HAVE MADE A CYCLE THRU THE PROGRAM ONCE
                                ;PER DEVICE I.E. - WE HAVE MADE A COMPLETE PASS
                                ;NOW WE NEED TO RESTORE EVERYTHING FOR THE NEXT COMPLETE PASS
010222  012767  000001  171046          MOV     #1,ROTADD       ;SET UP ROTATING POINTER FOR NEXT
                                                        ;MULTIPLE PASS
010230  016767  171026  171026          MOV     KEEPADD,BASEADD ;RESTORE BASE ADDRESS
010236  016767  171024  171024          MOV     KEEPIV,BASEIV   ;RESTORE BASE INTERRUPT VECTOR
010244  016767  171014  171006          MOV     BASEADD,DLBASE  ;RESTORE 1ST DEVICE BASE ADDRESS
010252  016767  171012  171140          MOV     BASEIV,DLVECT   ;RESTORE 1ST DEVICE VECTOR ADDRESS
010260  000240                          NOP
010262  004767  177512                  JSR     PC,DLADDR       ;FORM ADDRESSES FOR 1ST DEVICE
010266                          3$:
                                ;;**************************************************************************
010266  005067  170610          $TSTNM: CLR     $TSTNM          ;;ZERO THE TEST NUMBER
010272  005067  170744          $TIMES: CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
010276  005267  170576          $PASS:  INC     $PASS           ;;INCREMENT THE PASS NUMBER
010302  042767  100000  170570          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
010310  005327                          DEC     (PC)+           ;;LOOP?
010312  000001          $EOPCT: .WORD   1
010314  003022                          BGT     $DOAGN          ;;YES
010316  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
010320  000001          $ENDCT: .WORD   1
010322  010312                          $EOPCT
010324  104401  010371                  TYPE    ,$ENDMG         ;;TYPE 'END PASS #'
010330  016746  170544                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
010334  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
010336  104401  010366                  TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
010342  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
010346  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
010350  000005                          RESET                   ;;CLEAR THE WORLD
010352  004710          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
010354  000240                          NOP                     ;;SAVE ROOM
010356  000240                          NOP                     ;;FOR
010360  000240                          NOP                     ;;ACT11
010362                  $DOAGN:
010362  000137                          JMP     @(PC)+          ;;RETURN
010364  002036          $RTNAD: .WORD   RESTRT
```

```
 3052  010366     377     377     000  $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
 3053  010371     015  042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS #/
 3054  010376  050040  051501  020123
 3055  010404  000043
 3056                                   .SBTTL  SCOPE HANDLER ROUTINE
 3057
 3058                                   ;;*********************************************************************
 3059                                   ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
 3060                                   ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
 3061                                   ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
 3062                                   ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 3063                                   ;*SW14=1         LOOP ON TEST
 3064                                   ;*SW11=1         INHIBIT ITERATIONS
 3065                                   ;*SW09=1         LOOP ON ERROR
 3066                                   ;*SW08=1         LOOP ON TEST IN SWR<7:0>
 3067                                   ;*CALL
 3068                                   ;*      SCOPE            ;;SCOPE=IOT
 3069
 3070  010406                           $SCOPE:
 3071  010406  104407                          CKSWR                          ;;TEST  FOR CHANGE IN SOFT-SWR
 3072  010410  032777  040000  170522  1$:     BIT     #BIT14,@SWR            ;;LOOP ON PRESENT TEST?
 3073  010416  001111                          BNE     $OVER                  ;;YES IF SW14=1
 3074                                   ;#####START OF CODE FOR THE XOR TESTER#####
 3075  010420  000416                  $XTSTR: BR      6$                     ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
 3076                                                                         ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
 3077  010422  013746  000004                  MOV     @#ERRVEC,-(SP)         ;;SAVE THE CONTENTS OF THE ERROR VECTOR
 3078  010426  012737  010446  000004          MOV     #5$,@#ERRVEC           ;;SET FOR TIMEOUT
 3079  010434  005737  177060                  TST     @#177060               ;;TIME OUT ON XOR?
 3080  010440  012637  000004                  MOV     (SP)+,@#ERRVEC         ;;RESTORE THE ERROR VECTOR
 3081  010444  000463                          BR      $SVLAD                 ;;GO TO THE NEXT TEST
 3082  010446  022626                  5$:     CMP     (SP)+,(SP)+            ;;CLEAR THE STACK AFTER A TIME OUT
 3083  010450  012637  000004                  MOV     (SP)+,@#ERRVEC         ;;RESTORE THE ERROR VECTOR
 3084  010454  000423                          BR      7$                     ;;LOOP ON THE PRESENT TEST
 3085  010456                          6$:;#####END OF CODE FOR THE XOR TESTER#####
 3086  010456  032777  000400  170454          BIT     #BIT08,@SWR            ;;LOOP ON SPEC. TEST?
 3087  010464  001404                          BEQ     2$                     ;;BR IF NO
 3088  010466  127767  170446  170406          CMPB    @SWR,$TSTNM            ;;ON THE RIGHT TEST?    SWR<7:0>
 3089  010474  001462                          BEQ     $OVER                  ;;BR IF YES
 3090  010476  105767  170401          2$:     TSTB    $ERFLG                 ;;HAS AN ERROR OCCURRED?
 3091  010502  001421                          BEQ     3$                     ;;BR IF NO
 3092  010504  126767  170405  170371          CMPB    $ERMAX,$ERFLG          ;;MAX. ERRORS FOR THIS TEST OCCURRED?
 3093  010512  101015                          BHI     3$                     ;;BR IF NO
 3094  010514  032777  001000  170416          BIT     #BIT09,@SWR            ;;LOOP ON ERROR?
 3095  010522  001404                          BEQ     4$                     ;;BR IF NO
 3096  010524  016767  170360  170354  7$:     MOV     $LPERR,$LPADR          ;;SET LOOP ADDRESS TO LAST SCOPE
 3097  010532  000443                          BR      $OVER
 3098  010534  105067  170343          4$:     CLRB    $ERFLG                 ;;ZERO THE ERROR FLAG
 3099  010540  005067  170476                  CLR     $TIMES                 ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
 3100  010544  000415                          BR      1$                     ;;ESCAPE TO THE NEXT TEST
 3101  010546  032777  004000  170364  3$:     BIT     #BIT11,@SWR            ;;INHIBIT ITERATIONS?
 3102  010554  001011                          BNE     1$                     ;;BR IF YES
 3103  010556  005767  170316                  TST     $PASS                  ;;IF FIRST PASS OF PROGRAM
 3104  010562  001406                          BEQ     1$                     ;;       INHIBIT ITERATIONS
 3105  010564  005267  170314                  INC     $ICNT                  ;;INCREMENT ITERATION COUNT
 3106  010570  026767  170446  170306          CMP     $TIMES,$ICNT           ;;CHECK THE NUMBER OF ITERATIONS MADE
 3107  010576  002021                          BGE     $OVER                  ;;BR IF MORE ITERATION REQUIRED
```

```
3108  010600  012767  000001  170276  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
3109  010606  016767  000044  170426          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
3110  010614  105267  170262          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
3111  010620  011667  170262                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
3112  010624  011667  170260                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
3113  010630  005067  170410                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
3114  010634  112767  000001  170253          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
3115  010642  016777  170234  170272  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
3116  010650  016716  170232                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
3117  010654  000002                          RTI                     ;;FIXES PS
3118  010656  000100          $MXCNT: 100                             ;;MAX. NUMBER OF ITERATIONS
3119
3120                           .SBTTL  ERROR HANDLER ROUTINE
3121
3122                           ;;**********************************************************************
3123                           ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
3124                           ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
3125                           ;*AND GO TO $ERRTYP ON ERROR
3126                           ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
3127                           ;*SW15=1        HALT ON ERROR
3128                           ;*SW13=1        INHIBIT ERROR TYPEOUTS
3129                           ;*SW10=1        BELL ON ERROR
3130                           ;*SW09=1        LOOP ON ERROR
3131                           ;*CALL
3132                           ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
3133
3134  010660                   $ERROR:
3135  010660  104407                  CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
3136  010662  105267  170215  7$:     INCB    $ERFLG                  ;;SET THE ERROR FLAG
3137  010666  001775                  BEQ     7$                      ;;DON'T LET THE FLAG GO TO ZERO
3138  010670  016777  170206  170244  MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER AND ERROR FLAG
3139  010676  032777  002000  170234  BIT     #BIT10,@SWR             ;;BELL ON ERROR?
3140  010704  001402                  BEQ     1$                      ;;NO - SKIP
3141  010706  104401  001246          TYPE    ,$BELL                  ;;RING BELL
3142  010712  005267  170174  1$:     INC     $ERTTL                  ;;COUNT THE NUMBER OF ERRORS
3143  010716  011667  170174          MOV     (SP),$ERRPC             ;;GET ADDRESS OF ERROR INSTRUCTION
3144  010722  162767  000002  170166  SUB     #2,$ERRPC
3145  010730  117767  170162  170156  MOVB    @$ERRPC,$ITEMB          ;;STRIP AND SAVE THE ERROR ITEM CODE
3146  010736  032777  020000  170174  BIT     #BIT13,@SWR             ;;SKIP TYPEOUT IF SET
3147  010744  001004                  BNE     20$                     ;;SKIP TYPEOUTS
3148  010746  004767  000046          JSR     PC,$ERRTYP              ;;GO TO USER ERROR ROUTINE
3149  010752  104401  001253          TYPE    ,$CRLF
3150  010756                   20$:
3151  010756  005777  170156  2$:     TST     @SWR                    ;;HALT ON ERROR
3152  010762  100002                  BPL     3$                      ;;SKIP IF CONTINUE
3153  010764  000000                  HALT                            ;;HALT ON ERROR!
3154  010766  104407                  CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
3155  010770  032777  001000  170142  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
3156  010776  001402                  BEQ     4$                      ;;BR IF NO
3157  011000  016716  170104          MOV     $LPERR,(SP)             ;;FUDGE RETURN FOR LOOPING
3158  011004  005767  170234  4$:     TST     $ESCAPE                 ;;CHECK FOR AN ESCAPE ADDRESS
3159  011010  001402                  BEQ     5$                      ;;BR IF NONE
3160  011012  016716  170226          MOV     $ESCAPE,(SP)            ;;FUDGE RETURN ADDRESS FOR ESCAPE
3161  011016                   5$:
3162  011016  000002                  RTI                             ;;RETURN
3163
```

```
                                .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

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

011020                          $ERRTYP:
011020  104401  001253                TYPE    ,$CRLF          ;;''CARRIAGE RETURN'' & 'LINE FEED''
011024  010046                        MOV     R0,-(SP)        ;;SAVE R0
011026  005000                        CLR     R0              ;;PICKUP THE ITEM INDEX
011030  153700  001114                BISB    @#$ITEMB,R0
011034  001004                        BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                                ;;TYPE THE PC OF THE ERROR
011036  016746  170054                MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                ;;ERROR ADDRESS
011042  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
011044  000426                        BR      6$              ;;GET OUT
011046  005300                1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
011050  006300                        ASL     R0              ;;      WORK FOR THE ERROR TABLE
011052  006300                        ASL     R0
011054  006300                        ASL     R0
011056  062700  001310                ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
011062  012067  000004                MOV     (R0)+,2$        ;;PICKUP 'ERROR MESSAGE'' POINTER
011066  001404                        BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
011070  104401                        TYPE                    ;;TYPE THE 'ERROR MESSAGE''
011072  000000                2$:     .WORD   0               ;;'ERROR MESSAGE'' POINTER GOES HERE
011074  104401  001253                TYPE    ,$CRLF          ;;''CARRIAGE RETURN'' & 'LINE FEED''
011100  012067  000004                MOV     (R0)+,4$        ;;PICKUP 'DATA HEADER'' POINTER
011104  001404                        BEQ     5$              ;;SKIP TYPEOUT IF 0
011106  104401                        TYPE                    ;;TYPE THE 'DATA HEADER''
011110  000000                4$:     .WORD   0               ;;'DATA HEADER'' POINTER GOES HERE
011112  104401  001253                TYPE    ,$CRLF          ;;''CARRIAGE RETURN'' & 'LINE FEED''
011116  011000                5$:     MOV     (R0),R0         ;;PICKUP 'DATA TABLE'' POINTER
011120  001004                        BNE     7$              ;;GO TYPE THE DATA
011122  012600                6$:     MOV     (SP)+,R0        ;;RESTORE R0
011124  104401  001253                TYPE    ,$CRLF          ;;''CARRIAGE RETURN'' & 'LINE FEED''
011130  000207                        RTS     PC              ;;RETURN
011132                          7$:
011132  013046                        MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
011134  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
011136  005710                        TST     (R0)            ;;IS THERE ANOTHER NUMBER?
011140  001770                        BEQ     6$              ;;BR IF NO
011142  104401  011150                TYPE    ,8$             ;;TYPE TWO(2) SPACES
011146  000771                        BR      7$              ;;LOOP
011150  020040      000       8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
        011154                        .EVEN

                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

                                ;;**********************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                ;*$TYPOS--ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
```

```
3220                                    ;*      TYPOS                   ;;CALL FOR TYPEOUT
3221                                    ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3222                                    ;*      .BYTE   M               ;;M=1 OR 0
3223                                    ;*                                      ;1=TYPE LEADING ZEROS
3224                                    ;*                                     ;;0=SUPPRESS LEADING ZEROS
3225                                    ;*
3226                                    ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3227                                    ;*$TYPOS OR $TYPOC
3228                                    ;*CALL:
3229                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
3230                                    ;*      TYPON                   ;;CALL FOR TYPEOUT
3231                                    ;*
3232                                    ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3233                                    ;*CALL:
3234                                    ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
3235                                    ;*      TYPOC                   ;;CALL FOR TYPEOUT
3236
3237  011154  017646  000000            $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
3238  011160  116667  000001  000211            MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
3239  011166  112667  000207                    MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
3240  011172  062716  000002                    ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
3241  011176  000406                            BR      $TYPON
3242  011200  112767  000001  000171  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
3243  011206  112767  000006  000165            MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
3244  011214  112767  000005  000154  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
3245  011222  010346                            MOV     R3,-(SP)        ;;SAVE R3
3246  011224  010446                            MOV     R4,-(SP)        ;;SAVE R4
3247  011226  010546                            MOV     R5,-(SP)        ;;SAVE R5
3248  011230  116704  000145                    MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
3249  011234  005404                            NEG     R4
3250  011236  062704  000006                    ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
3251  011242  110467  000132                    MOVB    R4,$OMODE       ;;SAVE IT FOR USE
3252  011246  116704  000125                    MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
3253  011252  016605  000012                    MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
3254  011256  005003                            CLR     R3              ;;CLEAR THE OUTPUT WORD
3255  011260  006105                    1$:     ROL     R5              ;;ROTATE MSB INTO 'C'
3256  011262  000404                            BR      3$              ;;GO DO MSB
3257  011264  006105                    2$:     ROL     R5              ;;FORM THIS DIGIT
3258  011266  006105                            ROL     R5
3259  011270  006105                            ROL     R5
3260  011272  010503                            MOV     R5,R3
3261  011274  006103                    3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
3262  011276  105367  000076                    DECB    $OMODE          ;;TYPE THIS DIGIT?
3263  011302  100016                            BPL     7$              ;;BR IF NO
3264  011304  042703  177770                    BIC     #177770,R3      ;;GET RID OF JUNK
3265  011310  001002                            BNE     4$              ;;TEST FOR 0
3266  011312  005704                            TST     R4              ;;SUPPRESS THIS 0?
3267  011314  001403                            BEQ     5$              ;;BR IF YES
3268  011316  005204                    4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
3269  011320  052703  000060                    BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
3270  011324  052703  000040            5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
3271  011330  110367  000040                    MOVB    R3,8$           ;;SAVE FOR TYPING
3272  011334  104401  011374                    TYPE    ,8$             ;;GO TYPE THIS DIGIT
3273  011340  105367  000032            7$:     DECB    $OCNT           ;;COUNT BY 1
3274  011344  003347                            BGT     2$              ;;BR IF MORE TO DO
3275  011346  002402                            BLT     6$              ;;BR IF DONE
```

```
3276  011350  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
3277  011352  000744                          BR      2$              ;;GO DO THE LAST DIGIT
3278  011354  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
3279  011356  012604                          MOV     (SP)+,R4        ;;RESTORE R4
3280  011360  012603                          MOV     (SP)+,R3        ;;RESTORE R3
3281  011362  016666  000002  000004         MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
3282  011370  012616                          MOV     (SP)+,(SP)
3283  011372  000002                          RTI                     ;;RETURN
3284  011374     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
3285  011375     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
3286  011376     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
3287  011377     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
3288  011400  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
3289
3290                                  .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
3291
3292                                  ;;**************************************************************************
3293                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
3294                                  ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
3295                                  ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
3296                                  ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
3297                                  ;*REPLACED WITH SPACES.
3298                                  ;*CALL:
3299                                  ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
3300                                  ;*      TYPDS                   ;;GO TO THE ROUTINE
3301
3302  011402                          $TYPDS:
3303  011402  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3304  011404  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3305  011406  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3306  011410  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3307  011412  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
3308  011414  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
3309  011420  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
3310  011424  100004                          BPL     1$              ;;BR IF INPUT IS POS.
3311  011426  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
3312  011430  112766  000055  000001         MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
3313  011436  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
3314  011440  012703  011616                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
3315  011444  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
3316  011450  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
3317  011452  016001  011606                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
3318  011456  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
3319  011460  002402                          BLT     4$              ;;BR IF DONE
3320  011462  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
3321  011464  000774                          BR      3$
3322  011466  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
3323  011470  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
3324  011472  001002                          BNE     5$              ;;FALL THROUGH IF 0
3325  011474  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
3326  011476  100407                          BMI     7$              ;;BR IF YES
3327  011500  106316                  5$:     ASLB    (SP)            ;;MSD?
3328  011502  103003                          BCC     6$              ;;BR IF NO
3329  011504  116663  000001  177777         MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
3330  011512  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
3331  011516  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
```

```
3332  011522  110223                        MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
3333  011524  005720                        TST     (R0)+           ;;JUST INCREMENTING
3334  011526  020027  000010                CMP     R0,#10          ;;CHECK THE TABLE INDEX
3335  011532  002746                        BLT     2$              ;;GO DO THE NEXT DIGIT
3336  011534  003002                        BGT     8$              ;;GO TO EXIT
3337  011536  010502                        MOV     R5,R2           ;;GET THE LSD
3338  011540  000764                        BR      6$              ;;GO CHANGE TO ASCII
3339  011542  105726                8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
3340  011544  100003                        BPL     9$              ;;BR IF NO
3341  011546  116663  177777  177776        MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
3342  011554  105013                9$:     CLRB    (R3)            ;;SET THE TERMINATOR
3343  011556  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
3344  011560  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
3345  011562  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
3346  011564  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
3347  011566  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
3348  011570  104401  011616                TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
3349  011574  016666  000002  000004        MOV     2(SP),4(SP)     ;;ADJUST THE STACK
3350  011602  012616                        MOV     (SP)+,(SP)
3351  011604  000002                        RTI                     ;;RETURN TO USER
3352  011606  023420                $DTBL:  10000.
3353  011610  001750                        1000.
3354  011612  000144                        100.
3355  011614  000012                        10.
3356  011616  000004                $DBLK:  .BLKW   4
3357
3358                                        .SBTTL  TTY INPUT ROUTINE
3359
3360                                        ;;**************************************************************************
3361                                        .ENABL  LSB
3362  011626  000000                $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
3363  011630  000000                $TKQIN: .WORD   0               ;;INPUT POINTER
3364  011632  000000                $TKQOUT: .WORD  0               ;;OUTPUT POINTER
3365  011634  000010                $TKQSRT: .BLKB  8.              ;;TTY KEYBOARD QUEUE
3366          011644                $TKQEND=.
3367
3368                                        ;*TK INITIALIZE ROUTINE
3369                                        ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
3370                                        ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
3371                                        ;
3372                                        ;*CALL:
3373                                        ;*      JSR     PC,$TKINT
3374                                        ;*      RETURN
3375                                        ;
3376  011644  005067  177756        $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
3377  011650  012767  011634  177752        MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
3378  011656  016767  177746  177746        MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
3379  011664  012737  011714  000060        MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
3380  011672  012737  000200  000062        MOV     #200,@#TKVEC+2  ;;'BR' LEVEL 4
3381  011700  005777  167242                TST     @$TKB           ;;CLEAR DONE FLAG
3382  011704  012777  000100  167232        MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
3383  011712  000207                        RTS     PC              ;;RETURN TO CALLER
3384
3385                                        ;*TK SERVICE ROUTINE
3386                                        ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
3387                                        ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
```

```
3388                                    ;*IT IN THE QUEUE.
3389                                    ;*IF THE CHARACTER IS A ''CONTROL-C'' (^C) $TKINT IS CALLED AND
3390                                    ;*UPON RETURN EXIT IS MADE TO THE ''CONTROL-C'' RESTART ADDRESS (200)
3391                                    ;
3392  011714  117746  167226           $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
3393  011720  042716  177600                   BIC     #^C177,(SP)     ;;STRIP THE JUNK
3394  011724  021627  000003                   CMP     (SP),#3         ;;IS IT A CONTROL C?
3395  011730  001007                           BNE     1$              ;;BRANCH IF NO
3396  011732  104401  013030                   TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
3397  011736  004767  177702                   JSR     PC,$TKINT       ;;INIT THE KEYBOARD
3398  011742  005726                           TST     (SP)+           ;;CLEAN UP STACK
3399  011744  000167  166230                   JMP     200             ;;CONTROL C RESTART
3400  011750  021627  000007           1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
3401  011754  001004                           BNE     2$              ;;BRANCH IF NO
3402  011756  022767  000176  167154           CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
3403  011764  001500                           BEQ     6$              ;;GO TO SWR CHANGE
3404
3405  011766                           2$:
3406  011766  022767  000010  177632           CMP     #8.,$TKCNT      ;;IS THE QUEUE FULL?
3407  011774  001004                           BNE     3$              ;;BRANCH IF NO
3408  011776  104401  001246                   TYPE    ,$BELL          ;;RING THE TTY BELL
3409  012002  005726                           TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
3410  012004  000451                           BR      5$              ;;EXIT
3411  012006  021627  000023           3$:     CMP     (SP),#23        ;;IS IT A CONTROL-S?
3412  012012  001021                           BNE     32$             ;;BRANCH IF NO
3413  012014  005077  167124                   CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
3414  012020  005726                           TST     (SP)+           ;;CLEAN CHAR OFF STACK
3415  012022  105777  167116           31$:    TSTB    @$TKS           ;;WAIT FOR A CHAR
3416  012026  100375                           BPL     31$             ;;LOOP UNTIL ITS THERE
3417  012030  117746  167112                   MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
3418  012034  042716  177600                   BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
3419  012040  022627  000021                   CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
3420  012044  001366                           BNE     31$             ;;BRANCH IF NO
3421  012046  012777  000100  167070           MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
3422  012054  000002                           RTI                     ;;RETURN
3423  012056  005267  177544           32$:    INC     $TKCNT          ;;COUNT THIS CHARACTER
3424  012062  021627  000140                   CMP     (SP),#140       ;;IS IT UPPER CASE?
3425  012066  002405                           BLT     4$              ;;BRANCH IF YES
3426  012070  021627  000175                   CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
3427  012074  003002                           BGT     4$              ;;BRANCH IF YES
3428  012076  042716  000040                   BIC     #40,(SP)        ;;MAKE IT UPPER CASE
3429  012102  112677  177522           4$:     MOVB    (SP)+,@$TKQIN   ;;AND PUT IT IN QUEUE
3430  012106  005267  177516                   INC     $TKQIN          ;;UPDATE THE POINTER
3431  012112  026727  177512  011644           CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
3432  012120  001003                           BNE     5$              ;;BRANCH IF NO
3433  012122  012767  011634  177500           MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
3434  012130  000002                   5$:     RTI                     ;;RETURN
3435
3436                                    ;;**************************************************************************
3437                                    ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
3438                                    ;*ROUTINE IS ENTERED  FROM THE TRAP HANDLER, AND WILL
3439                                    ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
3440                                    ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
3441  012132  022767  000176  167000   $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
3442  012140  001124                           BNE     15$             ;;EXIT IF NOT
3443  012142  105777  166776                   TSTB    @$TKS           ;;IS A CHAR WAITING?
```

```
3444  012146  100121                        BPL     15$             ;;IF NOT, EXIT
3445  012150  117746  166772                MOVB    @$TKB,-(SP)     ;;YES
3446  012154  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
3447  012160  021627  000007                CMP     (SP),#7         ;;IS IT A CONTROL-G?
3448  012164  001300                        BNE     2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
3449                                                                ;;AND EXIT
3450
3451                                ;;**************************************************************************
3452                                ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
3453                                ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
3454                                ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
3455  012166  126727  166742  000001 6$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
3456  012174  001674                        BEQ     2$              ;;BRANCH IF YES
3457  012176  005726                        TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
3458  012200  004767  177440                JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
3459  012204  005077  166734                CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
3460  012210  112767  000001  166717        MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR
3461
3462  012216  104401  013042                TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
3463  012222  104401  013047  $GTSWR: TYPE  ,$MSWR          ;;TYPE CURRENT CONTENTS
3464  012226  016746  165744                MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
3465  012232  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3466  012234  104401  013060                TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
3467  012240  005046                19$:    CLR     -(SP)           ;;CLEAR COUNTER
3468  012242  005046                        CLR     -(SP)           ;;THE NEW SWR
3469  012244  105777  166674        7$:     TSTB    @$TKS           ;;CHAR THERE?
3470  012250  100375                        BPL     7$              ;;IF NOT TRY AGAIN
3471
3472  012252  117746  166670                MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
3473  012256  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
3474
3475  012262  021627  000003                CMP     (SP),#3         ;;IS IT A CONTROL-C?
3476  012266  001015                        BNE     9$              ;;BRANCH IF NOT
3477  012270  104401  013030                TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
3478  012274  062706  000006                ADD     #6,SP           ;;CLEAN UP STACK
3479  012300  126727  166631  000001        CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
3480  012306  001003                        BNE     8$              ;;BRANCH IF NO
3481  012310  012777  000100  166626        MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
3482  012316  000167  165656        8$:     JMP     200             ;;CONTROL-C RESTART
3483
3484
3485  012322  021627  000025        9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
3486  012326  001005                        BNE     10$             ;;BRANCH IF NOT
3487  012330  104401  013035                TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
3488  012334  062706  000006        20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
3489  012340  000737                        BR      19$             ;;LET'S TRY IT AGAIN
3490
3491
3492  012342  021627  000015        10$:    CMP     (SP),#15        ;;IS IT A <CR>?
3493  012346  001022                        BNE     16$             ;;BRANCH IF NO
3494  012350  005766  000004                TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
3495  012354  001403                        BEQ     11$             ;;BRANCH IF YES
3496  012356  016677  000002  166554        MOV     2(SP),@SWR      ;;SAVE NEW SWR
3497  012364  062706  000006        11$:    ADD     #6,SP           ;;CLEAR UP STACK
3498  012370  104401  001253        14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
3499  012374  126727  166535  000001        CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
```

```
3500  012402  001003                        BNE     15$             ;;BRANCH IF NOT
3501  012404  012777  000100  166532        MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
3502  012412  000002                15$:    RTI                     ;;RETURN
3503  012414  004767  000762        16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
3504  012420  021627  000060                CMP     (SP),#60        ;;CHAR < 0?
3505  012424  002420                        BLT     18$             ;;BRANCH IF YES
3506  012426  021627  000067                CMP     (SP),#67        ;;CHAR > 7?
3507  012432  003015                        BGT     18$             ;;BRANCH IF YES
3508  012434  042726  000060                BIC     #60,(SP)+       ;;STRIP-OFF ASCII
3509  012440  005766  000002                TST     2(SP)           ;;IS THIS THE FIRST CHAR
3510  012444  001403                        BEQ     17$             ;;BRANCH IF YES
3511  012446  006316                        ASL     (SP)            ;;NO, SHIFT PRESENT
3512  012450  006316                        ASL     (SP)            ;;    CHAR OVER TO MAKE
3513  012452  006316                        ASL     (SP)            ;;    ROOM FOR NEW ONE.
3514  012454  005266  000002        17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
3515  012460  056616  177776                BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
3516  012464  000667                        BR      7$              ;;GET THE NEXT ONE
3517  012466  104401  001252        18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
3518  012472  000720                        BR      20$             ;;SIMULATE CONTROL-U
3519                                .DSABL  LSB
3520
3521
3522                                ;;*******************************************************************
3523                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
3524                                ;*CALL:
3525                                ;*      RDCHR                   ;;GET A CHARACTER FROM THE QUEUE
3526                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
3527                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
3528                                ;
3529
3530  012474  011646                $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
3531  012476  016666  000004  000002        MOV     4(SP),2(SP)     ;;THE PS
3532  012504  005066  000004                CLR     4(SP)           ;;GET READY FOR A CHARACTER
3533  012510  005046                        CLR     -(SP)           ;;PUT NEW PS ON STACK
3534  012512  012746  012520                MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
3535  012516  000002                        RTI                     ;;POP NEW PC AND PS
3536  012520                        64$:
3537  012520  005767  177102        1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
3538  012524  001775                        BEQ     1$
3539  012526  005367  177074                DEC     $TKCNT          ;;DECREMENT THE COUNTER
3540  012532  117766  177074  000004        MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
3541  012540  005267  177066                INC     $TKQOUT         ;;UPDATE THE POINTER
3542  012544  026727  177062  011644        CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
3543  012552  001003                        BNE     2$              ;;BRANCH IF NO
3544  012554  012767  011634  177050        MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
3545  012562  000002                2$:     RTI                     ;;RETURN
3546                                ;;*******************************************************************
3547                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
3548                                ;*CALL:
3549                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
3550                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3551                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
3552
3553  012564  010346                $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
3554  012566  005046                        CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
3555  012570  012703  013020        1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
```

```
3556  012574  022703  013030          2$:   CMP   #$TTYIN+8.,R3   ;;BUFFER FULL?
3557  012600  101456                        BLOS  4$               ;;BR IF YES
3558  012602  104410                        RDCHR                  ;;GO READ ONE CHARACTER FROM THE TTY
3559  012604  112613                        MOVB  (SP)+,(R3)       ;;GET CHARACTER
3560  012606  122713  000177          10$:  CMPB  #177,(R3)        ;;IS IT A RUBOUT
3561  012612  001022                        BNE   5$               ;;BR IF NO
3562  012614  005716                        TST   (SP)             ;;IS THIS THE FIRST RUBOUT?
3563  012616  001007                        BNE   6$               ;;BR IF NO
3564  012620  112767  000134  000170        MOVB  #'\,9$           ;;TYPE A BACK SLASH
3565  012626  104401  013016                TYPE  ,9$
3566  012632  012716  177777                MOV   #-1,(SP)         ;;SET THE RUBOUT KEY
3567  012636  005303                  6$:   DEC   R3               ;;BACKUP BY ONE
3568  012640  020327  013020                CMP   R3,#$TTYIN       ;;STACK EMPTY?
3569  012644  103434                        BLO   4$               ;;BR IF YES
3570  012646  111367  000144                MOVB  (R3),9$          ;;SETUP TO TYPEOUT THE DELETED CHAR.
3571  012652  104401  013016                TYPE  ,9$              ;;GO TYPE
3572  012656  000746                        BR    2$               ;;GO READ ANOTHER CHAR.
3573  012660  005716                  5$:   TST   (SP)             ;;RUBOUT KEY SET?
3574  012662  001406                        BEQ   7$               ;;BR IF NO
3575  012664  112767  000134  000124        MOVB  #'\,9$           ;;TYPE A BACK SLASH
3576  012672  104401  013016                TYPE  ,9$
3577  012676  005016                        CLR   (SP)             ;;CLEAR THE RUBOUT KEY
3578  012700  122713  000025          7$:   CMPB  #25,(R3)         ;;IS CHARACTER A CTRL U?
3579  012704  001003                        BNE   8$               ;;BR IF NO
3580  012706  104401  013035                TYPE  ,$CNTLU          ;;TYPE A CONTROL 'U'
3581  012712  000726                        BR    1$               ;;GO START OVER
3582  012714  122713  000022          8$:   CMPB  #22,(R3)         ;;IS CHARACTER A '^R'?
3583  012720  001011                        BNE   3$               ;;BRANCH IF NO
3584  012722  105013                        CLRB  (R3)             ;;CLEAR THE CHARACTER
3585  012724  104401  001253                TYPE  ,$CRLF           ;;TYPE A 'CR' & 'LF'
3586  012730  104401  013020                TYPE  ,$TTYIN          ;;TYPE THE INPUT STRING
3587  012734  000717                        BR    2$               ;;GO PICKUP ANOTHER CHACTER
3588  012736  104401  001252          4$:   TYPE  ,$QUES           ;;TYPE A '?'
3589  012742  000712                        BR    1$               ;;CLEAR THE BUFFER AND LOOP
3590  012744  111367  000046          3$:   MOVB  (R3),9$          ;;ECHO THE CHARACTER
3591  012750  104401  013016                TYPE  ,9$
3592  012754  122723  000015                CMPB  #15,(R3)+        ;;CHECK FOR RETURN
3593  012760  001305                        BNE   2$               ;;LOOP IF NOT RETURN
3594  012762  105063  177777                CLRB  -1(R3)           ;;CLEAR RETURN (THE 15)
3595  012766  104401  001254                TYPE  ,$LF             ;;TYPE A LINE FEED
3596  012772  005726                        TST   (SP)+            ;;CLEAN RUBOUT KEY FROM THE STACK
3597  012774  012603                        MOV   (SP)+,R3         ;;RESTORE R3
3598  012776  011646                        MOV   (SP),-(SP)       ;;ADJUST THE STACK AND PUT ADDRESS OF THE
3599  013000  016666  000004  000002        MOV   4(SP),2(SP)      ;;      FIRST ASCII CHARACTER ON IT
3600  013006  012766  013020  000004        MOV   #$TTYIN,4(SP)
3601  013014  000002                        RTI                    ;;RETURN
3602  013016     000                  9$:   .BYTE 0                ;;STORAGE FOR ASCII CHAR. TO TYPE
3603  013017     000                        .BYTE 0                ;;TERMINATOR
3604  013020  000010          $TTYIN:   .BLKB 8.                   ;;RESERVE 8 BYTES FOR TTY INPUT
3605  013030  041536  005015     000  $CNTLC:   .ASCIZ /^C/<15><12>    ;;CONTROL 'C'
3606  013035     136  006525  000012  $CNTLU:   .ASCIZ /^U/<15><12>    ;;CONTROL 'U'
3607  013042  043536  005015     000  $CNTLG:   .ASCIZ /^G/<15><12>    ;;CONTROL 'G'
3608  013047     015  051412  051127  $MSWR:    .ASCIZ <15><12>/SWR = /
3609  013054  036440  000040
3610  013060  020040  042516  020127  $MNEW:    .ASCIZ /  NEW = /
3611  013066  020075     000
```

```
 3612            013072                  .EVEN
 3613
 3614                                    .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
 3615
 3616                                    ;;*******************************************************************
 3617                                    ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
 3618                                    ;*CHANGE IT TO BINARY.
 3619                                    ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
 3620                                    ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A '?' WILL BE TYPED
 3621                                    ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
 3622                                    ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
 3623                                    ;*CALL:
 3624                                    ;*      RDOCT                   ;;READ AN OCTAL NUMBER
 3625                                    ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
 3626                                    ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
 3627
 3628  013072  011646                    $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
 3629  013074  016666  000004  000002            MOV     4(SP),2(SP)     ;;INPUT NUMBER
 3630  013102  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 3631  013104  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 3632  013106  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 3633  013110  104411                    1$:     RDLIN                   ;;READ AN ASCIZ LINE
 3634  013112  012600                            MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
 3635  013114  010067  000100                    MOV     R0,5$           ;;AND SAVE IT
 3636  013120  005001                            CLR     R1              ;;CLEAR DATA WORD
 3637  013122  005002                            CLR     R2
 3638  013124  112046                    2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
 3639  013126  001420                            BEQ     3$              ;;IF ZERO GET OUT
 3640  013130  122716  000060                    CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
 3641  013134  003026                            BGT     4$              ;;IS AN OCTAL DIGIT
 3642  013136  122716  000067                    CMPB    #'7,(SP)
 3643  013142  002423                            BLT     4$
 3644  013144  006301                            ASL     R1              ;;*2
 3645  013146  006102                            ROL     R2
 3646  013150  006301                            ASL     R1              ;;*4
 3647  013152  006102                            ROL     R2
 3648  013154  006301                            ASL     R1              ;;*8
 3649  013156  006102                            ROL     R2
 3650  013160  042716  177770                    BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
 3651  013164  062601                            ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
 3652  013166  00C756                            BR      2$              ;;LOOP
 3653  013170  005726                    3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
 3654  013172  010166  000012                    MOV     R1,12(SP)       ;;SAVE THE RESULT
 3655  013176  010267  000026                    MOV     R2,$HIOCT
 3656  013202  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
 3657  013204  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
 3658  013206  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
 3659  013210  000002                            RTI                     ;;RETURN
 3660  013212  005726                    4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
 3661  013214  105010                            CLRB    (R0)            ;;SET A TERMINATOR
 3662  013216  104401                            TYPE                    ;;TYPE UP THRU THE BAD CHAR.
 3663  013220  000000                    5$:     .WORD   0
 3664  013222  104401  001252                    TYPE    ,$QUES          ;;'?' 'CR' & 'LF'
 3665  013226  000730                            BR      1$              ;;TRY AGAIN
 3666  013230  000000                    $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
 3667
```

```
                                .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
```

```
3724  013400  000724                          BR      2$              ;;GET NEXT CHARACTER
3725  013402  105777  165542          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
3726  013406  100375                          BPL     $TYPEC
3727  013410  116677  000002  165534          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
3728  013416  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
3729  013424  001003                          BNE     1$              ;;BRANCH IF NO
3730  013426  105067  000014                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
3731  013432  000406                          BR      $TYPEX          ;;EXIT
3732  013434  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
3733  013442  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
3734  013444  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
3735  013446  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
3736  013450  000207          $TYPEX: RTS     PC
3737
3738
3739                          .SBTTL  READ A DECIMAL NUMBER FROM THE TTY
3740
3741                          ;;**************************************************************************
3742                          ;*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
3743                          ;*CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
3744                          ;*ARE READ A "?" FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
3745                          ;*THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
3746                          ;*USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
3747                          ;*POSITIVE 32767 TO NEGATIVE 32768.
3748                          ;*CALL:
3749                          ;*      RDDEC                   ;;READ A DECIMAL NUMBER
3750                          ;*      RETURN HERE             ;;NUMBER IS ON TOP OF THE STACK
3751                          ;
3752
3753  013452  011646          $RDDEC: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR
3754  013454  016663  000004  000002          MOV     4(SP),2(SP)     ;;THE INPUT NUMBER
3755  013462  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3756  013464  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3757  013466  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3758  013470  104411                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
3759  013472  012600                          MOV     (SP)+,R0        ;;ADDRESS OF 1ST CHAR.
3760  013474  010067  000120                  MOV     R0,6$           ;;SAVE INCASE OF BAD INPUT
3761  013500  005046                          CLR     -(SP)           ;;CLEAR DATA WORD
3762  013502  005002                          CLR     R2              ;;SIGN SET POSITIVE
3763  013504  122710  000055                  CMPB    #'-,(R0)        ;;SEE IF A MINUS SIGN WAS TYPED
3764  013510  001001                          BNE     2$              ;;BR IF NO MINUS SIGN
3765  013512  112002                          MOVB    (R0)+,R2        ;;SAVE FOR LATER USE
3766  013514  112001                  2$:     MOVB    (R0)+,R1        ;;PICKUP THIS CHARACTER
3767  013516  001424                          BEQ     3$              ;;GET OUT IF ZERO
3768  013520  122701  000060                  CMPB    #'0,R1          ;;MAKE SURE THIS CHARACTER
3769  013524  003032                          BGT     5$              ;;IS A DIGIT BETWEEN 0 & 9
3770  013526  122701  000071                  CMPB    #'9,R1
3771  013532  002427                          BLT     5$
3772  013534  032716  170000                  BIT     #^C7777,(SP)    ;;DON'T LET NUMBER GET TO BIG
3773  013540  001024                          BNE     5$              ;;BR IF NUMBER WOULD OVERFLOW
3774  013542  006316                          ASL     (SP)            ;;*2
3775  013544  011646                          MOV     (SP),-(SP)      ;;SAVE FOR LATER
3776  013546  006316                          ASL     (SP)            ;;*4
3777  013550  006316                          ASL     (SP)            ;;*8.
3778  013552  062616                          ADD     (SP)+,(SP)      ;;*10.
3779  013554  102416                          BVS     5$              ;;OVERFLOW ISN'T ALLOWED
```

```
3780  013556  162701  000060          SUB     #'0,R1          ;;STRIP AWAY THE ASCII JUNK
3781  013562  060116                  ADD     R1,(SP)         ;;ADD IN THIS DIGIT
3782  013564  102412                  BVS     5$              ;;OVERFLOW ISN'T ALLOWED
3783  013566  000752                  BR      2$              ;;LOOP
3784  013570  005702          3$:     TST     R2              ;;CHECK IF NUMBER IS NEG
3785  013572  001401                  BEQ     4$              ;;BR IF NO
3786  013574  005416                  NEG     (SP)            ;;YES--NEGATE THE NUMBER
3787  013576  012666  000012  4$:     MOV     (SP)+,12(SP)    ;;SAVE THE RESULT
3788  013602  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
3789  013604  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
3790  013606  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
3791  013610  000002                  RTI                     ;;RETURN
3792
3793  013612  005726          5$:     TST     (SP)+           ;;CLEAN PARTIAL NUMBER FROM STACK
3794  013614  105010                  CLRB    (R0)            ;;SET A TERMINATOR
3795  013616  104401                  TYPE                    ;;TYPE THE INPUT UP TO BAD CHAR.
3796  013620  000000          6$:     .WORD   0               ;;POINTER GOES HERE
3797  013622  104401  001252          TYPE    ,$QUES          ;;'?''CR'&'LF'
3798  013626  000720                  BR      1$              ;;TRY AGAIN
3799
3800                          .SBTTL  TRAP DECODER
3801
3802                          ;;**************************************************************************
3803                          ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
3804                          ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3805                          ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3806                          ;*GO TO THAT ROUTINE.
3807
3808  013630  010046          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
3809  013632  016600  000002          MOV     2(SP),R0        ;;GET TRAP ADDRESS
3810  013636  005740                  TST     -(R0)           ;;BACKUP BY 2
3811  013640  111000                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
3812  013642  006300                  ASL     R0              ;;POSITION FOR INDEXING
3813  013644  016000  013664          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
3814  013650  000200                  RTS     R0              ;;GO TO ROUTINE
3815
3816
3817                          ;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO
3818
3819  013652  011646          $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
3820  013654  016666  000004  000002  MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
3821  013662  000002                  RTI                     ;;RESTORE THE PSW
3822
3823                          .SBTTL  TRAP TABLE
3824
3825                          ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3826                          ;*BY THE 'TRAP' INSTRUCTION.
3827
3828                          ;       ROUTINE
3829                          ;       -------
3830  013664  013652          $TRPAD: .WORD   $TRAP2
3831  013666  013232                  $TYPE   ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
3832  013670  011200                  $TYPOC  ;;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3833  013672  011154                  $TYPOS  ;;CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
3834  013674  011214                  $TYPON  ;;CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
3835  013676  011402                  $TYPDS  ;;CALL=TYPDS     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
```

```
3836
3837  013700  012222                        $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
3838
3839  013702  012132                        $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
3840  013704  012474                        $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
3841  013706  012564                        $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
3842  013710  013072                        $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
3843  013712  013452                        $RDDEC  ;;CALL=RDDEC    TRAP+13(104413) READ A DECIMAL NUMBER FROM TTY
3844
3845                              .SBTTL  POWER DOWN AND UP ROUTINES
3846
3847                              ;;**********************************************************************
3848                              ;POWER DOWN ROUTINE
3849  013714  012737  014060  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
3850  013722  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
3851  013730  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3852  013732  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3853  013734  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3854  013736  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3855  013740  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3856  013742  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
3857  013744  017746  165170                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
3858  013750  010667  000110                  MOV     SP,$SAVR6       ;;SAVE SP
3859  013754  012737  013766  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
3860  013762  000000                          HALT
3861  013764  000776                          BR      .-2             ;;HANG UP
3862
3863                              ;;**********************************************************************
3864                              ;POWER UP ROUTINE
3865  013766  012737  014060  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
3866  013774  016706  000064                  MOV     $SAVR6,SP       ;;GET SP
3867  014000  005067  000060                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
3868  014004  005267  000054          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
3869  014010  001375                          BNE     1$              ;;OF  WORD
3870  014012  012677  165122                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
3871  014016  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
3872  014020  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
3873  014022  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
3874  014024  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
3875  014026  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
3876  014030  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
3877  014032  012737  013714  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
3878  014040  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
3879  014046  104401                          TYPE                    ;REPORT THE POWER FAILURE
3880  014050  014066                  $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
3881  014052  012716                          MOV     (PC)+,(SP)      ;;RESTART AT RESTRT
3882  014054  002036                  $PWRAD: .WORD   RESTRT          ;;RESTART ADDRESS
3883  014056  000002                          RTI
3884  014060  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
3885  014062  000776                          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
3886  014064  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
3887  014066  005015  047520  042527  $POWER: .ASCIZ  <15><12>'POWER'
3888  014074  000122
3889                                          .EVEN
3890
3891                              ;;**********************************************************************
```

```
3892                                    ;TRANSMIT INTERRUPT SERVICE ROUTINE FOR 256. BYTE BLOCK TRANSFERS
3893                                    ;;*****************************************************************
3894
3895  014076  105777  165312          XINT:   TSTB    @DLXCSR         ;'READY'' SET ??
3896  014102  100416                          BMI     1$              ;BR IF YES
3897  014104  013767  177776  165070          MOV     @#PSW,$TMP0     ;SAVE THE ERROR PSW
3898  014112  010667  165060                  MOV     SP,$REG6        ;SAVE THE ERROR STACK POINTER
3899  014116  005167  165300                  COM     XFLG0           ;SET XMIT SOFTWARE ERROR FLAG
3900  014122  042777  000100  165260          BIC     #100,@DLRCSR    ;TURN OFF THE INTERRUPT ENABLES
3901  014130  042777  000100  165256          BIC     #100,@DLXCSR
3902  014136  000411                          BR      2$              ;GO TO EXIT
3903  014140  022767  022440  165264  1$:     CMP     #DLBUFI,OPTR    ;XMITTED 256. BYTES YET ??
3904  014146  001405                          BEQ     2$              ;BR IF YES
3905  014150  117777  165256  165240          MOVB    @OPTR,@DLXDBR   ;OUTPUT A BYTE
3906  014156  005267  165250                  INC     OPTR            ;UPDATE BUFFER POINTER
3907  014162  000002                  2$:     RTI                     ;RETURN TO MAINLINE TEST
3908
3909                                    ;;*****************************************************************
3910                                    ;RECEIVER INTERRUPT SERVICE ROUTINE FOR 256. BYTE BLOCK TRANSFERS
3911                                    ;;*****************************************************************
3912
3913  014164  105777  165220          RINT:   TSTB    @DLRCSR ;'DONE'' SET ??
3914  014170  100410                          BMI     1$              ;BR IF YES
3915  014172  013767  177776  165002          MOV     @#PSW,$TMP0     ;SAVE THE ERROR PSW
3916  014200  010667  164772                  MOV     SP,$REG6        ;SAVR THE ERROR STACK POINTER
3917  014204  005167  165214                  COM     RFLG0           ;SET HARD RCVR ERROR FLAG
3918  014210  000415                          BR      2$              ;GO EXIT
3919  014212  005777  165174          1$:     TST     @DLRDBR         ;ANY SOFT ERRORS ??
3920  014216  100021                          BPL     3$              ;BR IF NOT
3921  014220  013767  177776  164754          MOV     @#PSW,$TMP0     ;SAVE THE ERROR PSW
3922  014226  010667  164744                  MOV     SP,$REG6        ;SAVE THE ERROR STACK POINTER
3923  014232  017767  165154  164744          MOV     @DLRDBR,$TMP1   ;SAVE THE ERROR REGISTER IN TMP1
3924  014240  005167  165162                  COM     RFLG1           ;SET THE SOFT ERROR FLAG
3925  014244  042777  000100  165142  2$:     BIC     #100,@DLXCSR    ;TURN OFF THE INTR. ENABLES
3926  014252  042777  000100  165130          BIC     #100,@DLRCSR
3927  014260  000411                          BR      4$              ;GO TO EXIT
3928  014262  022767  023040  165144  3$:     CMP     #BUFEND,IPTR    ;RECEIVED 256. BYTES YET ??
3929  014270  001405                          BEQ     4$              ;BR IF YES
3930  014272  117777  165114  165134          MOVB    @DLRDBR,@IPTR   ;INPUT A BYTE FROM THE DL11
3931  014300  005267  165130                  INC     IPTR            ;UPDATE BUFFER POINTER
3932  014304  000002                  4$:     RTI                     ;RETURN TO MAINLINE TEST
3933
3934                                    ;THE FOLLOWING ROUTINE IS USED BY THE USER UTILITY PROGRAMS TO WAIT
3935                                    ;A SPECIFIED NO. OF MILLISECONDS BETWEEN CHARACTER TRANSFERS
3936
3937  014306  017667  000000  000034  DELAY:  MOV     @(R6),DELCNT    ;GET THE NO. OF MSEC. DELAY COUNT
3938                                                                  ;TYPED IN BY USER
3939  014314  062716  000002                  ADD     #2,(R6)         ;SET UP THIS ROUTINE'S EXIT ADDRESS
3940  014320  005767  000024                  TST     DELCNT          ;IS THE DELAY COUNT ZERO?
3941  014324  001410                          BEQ     3$              ;BRANCH IF YES
3942  014326  012746  000226          1$:     MOV     #226,-(SP)      ;PUSH A 1 MSEC. COUNT TO STACK
3943  014332  005316                  2$:     DEC     (SP)            ;DECREMENT THE 1 MSEC. COUNT BY 1
3944  014334  001376                          BNE     2$              ;BRANCH IF 1 MSEC. NOT EATEN
3945                                                                  ;AWAY YET
3946  014336  005726                          TST     (SP)+           ;RESET STACK AFTER 1 MSEC. TIME UP
3947  014340  005367  000004                  DEC     DELCNT          ;DECREMENT THE TOTAL NO. OF
```

```
3948
3949  014344  001370                    BNE     1$              ;MSECS. COUNT
3950                                                            ;BRANCH IF WE HAVE MORE MSECS.
3951  014346  000207              3$:   RTS     PC              ;TO WAIT
3952  014350  000000              DELCNT: .WORD 0               ;GO BACK TO REISSUE A CHARACTER
3953                                                            ;THE NO. OF MSECS. NEEDED TO
3954                                    ;THE FOLLOWING ROUTINE IS USED BY USER PROGRAM #4 AND WILL ALLOW
3955                                    ;A RANDOM NUMBER OF MILLISECONDS BEFORE TRANSMISSION OF CHARACTER
3956                                    ;
3957  014352  016700  000062    STALL:  MOV     NUMONE,R0       ;GET THE LOW LIMIT
3958  014356  006100                    ROL     R0              ;MULTIPLY BY 4
3959  014360  006100                    ROL     R0              ;
3960  014362  066700  000054            ADD     NUMTWO,R0       ;ADD IN THE HIGH LIMIT
3961  014366  010067  000046            MOV     R0,NUMONE       ;STORE THIS AS NEW LOW LIMIT
3962  014372  006100                    ROL     R0              ;MULTIPLY NEW LOW LIMIT BY 4
3963  014374  006100                    ROL     R0              ;
3964  014376  066700  000040            ADD     NUMTWO,R0       ;ADD IN THE HIGH LIMIT
3965  014402  006100                    ROL     R0              ;MULTIPLY BY 4 AGAIN
3966  014404  006100                    ROL     R0              ;
3967  014406  010067  000030            MOV     R0,NUMTWO       ;STORE THIS AS NEW HIGH LIMIT
3968  014412  016700  000022            MOV     NUMONE,R0       ;SAVE THE RANDOMLY GENERATED NO.
3969  014416  046700  164664            BIC     STLMSK,R0       ;STRIP ALL BUT 1ST 5 BITS SO AS
3970                                                            ;NOT TO ALLOW THE STALL TO BE TOO
3971                                                            ;LARGE
3972  014422  001405                    BEQ     2$              ;BRANCH IF RESULT WAS ZERO
3973  014424  010067  000004            MOV     R0,1$           ;SET STALL TIME FOR DELAY ROUTINE
3974  014430  004767  177652            JSR     PC,DELAY        ;GO OFF TO STALL
3975  014434  000000              1$:   .WORD   0               ;THIS IS WHERE STALL TIME RESIDES
3976  014436  000207              2$:   RTS     PC              ;RETURN TO ISSUE CHARACTER
3977  014440  001233              NUMONE: 1233                  ;LOW LIMIT FOR RANDOM NO.
3978  014442  007622              NUMTWO: 7622                  ;HIGH LIMIT FOR RANDOM NO.
3979                                    ;THE FOLLOWING ROUTINE CHECKS THE 'DONE' BIT FOR BOTH THE RECEIVER
3980                                    ;AND TRANSMITTER.  THIS POUTINE IS USED BY PROGRAM #4
3981                                    ;
3982  014444  016767  164540  000044  TIMERX: MOV     $TMP3,DUT       ;GET THE TRANSMITTER CONTROL
3983                                                            ;STATUS REGISTER ADDRESS
3984  014452  162767  000004  000036          SUB     #4,DUT          ;FORM THE RECEIVER CONTROL
3985                                                            ;STATUS REGISTER ADDRESS
3986  014460  000403                    BR      TCONT           ;GO TO TIME OUT THE RECEIVERS'
3987                                                            ;DONE BIT
3988  014462  016767  164522  000026  TIMETX: MOV     $TMP3,DUT       ;GET THE TRANSMITTER CONTROL
3989                                                            ;STATUS REGISTER ADDRESS
3990  014470  005067  164522            TCONT:  CLR     $TMP6           ;INITIALIZE A TIME COUNT
3991  014474  005267  164516            1$:     INC     $TMP6           ;INCREMENT THE TIME COUNT
3992  014500  001405                    BEQ     2$              ;BRANCH IF TIME COUNTER OVERFLOWED
3993                                                            ;INDICATING DONE BIT NEVER SET
3994                                                            ;WITH PLENTY OF TIME ELAPSED
3995  014502  105777  000010            TSTB    @DUT            ;SEE IF DONE BIT IS SET YET
3996  014506  100372                    BPL     1$              ;WAIT SOME MORE IF IT ISN'T
3997  014510  062716  000006            ADD     #6,@R6          ;DONE BIT IS SET - SET UP EXIT
3998                                                            ;RETURN TO SKIP ERROR REPORT
3999  014514  000207              2$:   RTS     PC              ;RETURN TO PROGRAM #4
4000  014516  000000              DUT:  .WORD   0               ;THIS IS WHERE THE RCSR OR XCSR
4001                                                            ;ADDRESS RESIDES
4002                                    ;THIS ROUTINE IS USED BY PROGRAMS #4 & 5, AND WILL CHECK FOR CORRECT
4003                                    ;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
014534  016767  164426  164422          MOV     $REG2,$REG1     ;GET THE ADDRESS OF THE RECEIVER
                                                                ;DATA BUFFER
014542  162767  000002  164414          SUB     #2,$REG1        ;FORM THE ADDRESS OF THE RECEIVER
                                                                ;STATUS REGISTER FROM IT
014550  016767  164440  164412          MOV     $TMP5,$REG3     ;STORE THE CONTENTS OF THE RECEIVER
                                                                ;DATA BUFFER
014556  032767  170000  164432          BIT     #170000,$TMP6   ;ARE ANY ERROR BITS SET?
014564  001013                          BNE     1$              ;BRANCH IF YES
014566  004767  000720                  JSR     PC,UPMASK       ;GO TO MASK OFF BITS AS A FUNCTION OF
                                                                ;CHARACTER LENGTH( 5, 6, 7, OR 8 BITS)
014572  026767  164416  164432          CMP     $TMP5,$TMP14    ;WAS RECEIVED CHARACTER THE
                                                                ;SAME AS THE ONE TRANSMITTED?
014600  001406                          BEQ     2$              ;BRANCH IF YES
014602  016767  164424  164362          MOV     $TMP14,$REG4    ;STORE WHAT THE CONTENTS OF THE
                                                                ;RECEIVER DATA BUFFER SHOULD BE
014610  104010                          ERROR   +10             ;DATA RECEIVED WRONG!
014612  000401                          BR      2$              ;GET SET TO RETURN AFTER ERROR REPORT
014614  104007                  1$:     ERROR   +7              ;ERROR BIT/S SET FROM TRANSMISSION
014616  000207                  2$:     RTS     PC              ;RETURN TO PROGRAM #4

                                ;;**************************************************************************
                                        ;SUBROUTINE TO SETUP ERROR INFORMATION FOR ERROR MESSAGES
                                ;;**************************************************************************


014620  013767  177776  164354  SUER2:  MOV     @#PSW,$TMP0     ;SAVE THE [PSW]
014626  016701  164556          MOV     DLRCSR,R1       ;PUT DEVADR IN R1
014632  011203                  MOV     (R2),R3         ;PUT WAS INFO IN R3
014634  010667  164336          MOV     SP,$REG6        ;SAVE THE [SP]
014640  062767  000002  164330  ADD     #2,$REG6        ;CORRECT FOR CALLING JSR
014646  116700  164230  SUERR1: MOVB    $TSTNM,R0       ;PUT TEST NO. IN R0
014652  010067  164304          MOV     R0,$REG0        ;SAVE [R0] THRU [R4]
014656  010167  164302          MOV     R1,$REG1
014662  010267  164300          MOV     R2,$REG2
014666  010367  164276          MOV     R3,$REG3
014672  010467  164274          MOV     R4,$REG4
014676  000207                  RTS     PC              ;RETURN TO CALLING TEST

014700  013767  177776  164274  SUERT1: MOV     @#PSW,$TMP0             ;SAVE THE [PSW]
014706  116700  164170          MOVB    $TSTNM,R0               ;PUT TEST NO. IN R0
014712  016701  164472          MOV     DLRCSR,R1               ;PUT DEVADR IN R1
014716  010067  164240          MOV     R0,$REG0                ;SAVE [R0]
014722  010167  164236          MOV     R1,$REG1                ;SAVE [R1]
014726  013767  177776  164250  SUERT2: MOV     @#PSW,$TMP1             ;SAVE THE [PSW]
014734  010667  164236          MOV     SP,$REG6                ;SAVE THE [SP]
014740  062767  000002  164230          ADD     #2,$REG6                ;CORRECT FOR CALLING JSR
014746  010267  164214          MOV     R2,$REG2                ;SAVE [R2]
014752  000207                  RTS     PC                      ;RETURN

                                ;SUBROUTINE TO SETUP VECTORS FOR 256. BYTE BLOCK TRANSFER TESTS
```

```
 4060  014754  016705  164440          SUVEC:  MOV     DLVECT,R5       ;GET FIRST VECTOR ADDRESS
 4061  014760  012725  014164                  MOV     #RINT,(R5)+     ;SET UP RCVR VECTOR
 4062  014764  016725  164312                  MOV     DLPRI,(R5)+
 4063  014770  012725  014076                  MOV     #XINT,(R5)+     ;SET UP XMIT VECTOR
 4064  014774  016715  164302                  MOV     DLPRI,(R5)
 4065  015000  000207                          RTS     PC              ;RETURN TO CALLER
 4066
 4067                                  ;SUBROUTINE TO PRIME DATA BUFFERS AND DEVICE FOR 256. BYTE TRANSFER
 4068
 4069  015002  005077  164406          PRIME:  CLR     @DLXCSR         ;CLEAR XMIT AND RCVR CSR'S
 4070  015006  005077  164376                  CLR     @DLRCSR
 4071  015012  005067  164404                  CLR     XFLG0           ;INITIALIZE ERROR FLAGS
 4072  015016  005067  164402                  CLR     RFLG0
 4073  015022  005067  164400                  CLR     RFLG1
 4074  015026  012767  022040  164376          MOV     #DLBUF0,OPTR    ;SET UP OUTPUT POINTER
 4075  015034  012767  022440  164372          MOV     #DLBUFI,IPTR    ;SET UP INPUT POINTER
 4076  015042  004767  000044                  JSR     PC,CLDLBF       ;GO CLEAR THE BUFFERS
 4077  015046  004777  164364                  JSR     PC,@LDOUT       ;GO SET UP THE PATTERN
 4078  015052  005067  164362                  CLR     TIMR1           ;INIT TIMEOUT COUNTERS
 4079  015056  012767  000036  164356          MOV     #30.,TIMR2
 4080  015064  005777  164322                  TST     @DLRDBR         ;FLUSH 'DONE' BIT IN RCVR CSR
 4081  015070  005777  164316                  TST     @DLRDBR
 4082  015074  052777  000100  164306          BIS     #100,@DLRCSR    ;ENABLE RCVR INTR.
 4083  015102  052777  000104  164304          BIS     #104,@DLXCSR    ;ENABLE XMIT INTR. AND MAINT MODE
 4084  015110  000207                          RTS     PC
 4085
 4086
 4087
 4088
 4089                                  ;THIS ROUTINE IS CALLED TO CLEAR THE INPUT AND OUTPUT BUFFERS
 4090
 4091  015112  012705  022040          CLDLBF: MOV     #DLBUF0,R5      ;R5 POINTS TO BEGINNING OF BUFFER AREA
 4092  015116  005025                  1$:     CLR     (R5)+           ;CLEAR A WORD
 4093  015120  022705  023040                  CMP     #BUFEND,R5      ;DONE ALL WORDS ??
 4094  015124  001374                          BNE     1$              ;BR IF NOT
 4095  015126  000207                          RTS     PC              ;RETURN TO CALLER
 4096
 4097                                  ;THIS ROUTINE IS CALLED TO SET UP THE NULL-DEL-NULL PATTERN
 4098
 4099  015130  012705  022040          LDOUT1: MOV     #DLBUF0,R5      ;R5 POINTS TO OUTPUT BUFFER
 4100  015134  105025                  1$:     CLRB    (R5)+           ;MOVE A NULL CHAR
 4101  015136  112725  000377                  MOVB    #377,(R5)+      ;MOV A DEL CHAR
 4102  015142  022705  022440                  CMP     #DLBUFI,R5      ;ALL DONE ??
 4103  015146  001372                          BNE     1$              ;BR IF NOT
 4104  015150  000207                          RTS     PC              ;RETURN TO CALLER
 4105
 4106                                  ;THIS ROUTINE IS USED TO LOAD AN ASCENDING BINARY COUNT PATTERN
 4107
 4108  015152  005005                  LDOUT2: CLR     R5              ;START WITH 000
 4109  015154  110565  022040          1$:     MOVB    R5,DLBUF0(R5)   ;LOAD ONE BYTE
 4110  015160  005205                          INC     R5              ;INCREMENT BYTE
 4111  015162  022705  000400                  CMP     #400,R5         ;DONE 000 THRU 377 ??
 4112  015166  001372                          BNE     1$              ;BR IF NOT
 4113  015170  000207                          RTS     PC              ;RETURN TO CALLER
 4114
 4115                                  ;THIS ROUTINE IS USED TO LOAD A DESCENDING BINARY COUNT PATTERN
```

```
 4116
 4117  015172  112767  000377  164020  LDOUT3: MOVB    #377,$TMP7      ;START WITH A 377 BYTE
 4118  015200  012705  022040                  MOV     #DLBUFO,R5      ;R5 POINTS TO OUTPUT BUFFER
 4119  015204  116725  164010          1$:     MOVB    $TMP7,(R5)+     ;LOAD ONE BYTE
 4120  015210  022705  022440                  CMP     #DLBUFI,R5      ;ALL DONE ??
 4121  015214  001403                          BEQ     2$              ;BR IF YES
 4122  015216  105367  163776                  DECB    $TMP7           ;GENERATE NEXT BYTE
 4123  015222  000770                          BR      1$              ;GO MOVE IT
 4124  015224  000207          2$:     RTS     PC                      ;RETURN TO CALLER
 4125
 4126
 4127                                  ;THIS ROUTINE LOADS A COMPLEMENTING WORST CASE PATTERN
 4128
 4129  015226  012705  022040          LDOUT4: MOV     #DLBUFO,R5      ;R5 POINTS TO OUTPUT BUFFER
 4130  015232  005067  163762                  CLR     $TMP7           ;INIT. BYTE GENERATOR
 4131  015236  116725  163756          1$:     MOVB    $TMP7,(R5)+     ;MOVE A BYTE
 4132  015242  105167  163752                  COMB    $TMP7           ;COMPLEMENT IT
 4133  015246  116725  163746                  MOVB    $TMP7,(R5)+     ;NOW LOAD THE 1'S COMPLEMENT
 4134  015252  105267  163743                  INCB    $TMP7+1         ;INCREMENT THE BYTE
 4135  015256  116767  163737  163734          MOVB    $TMP7+1,$TMP7   ;SET UP TO LOAD NEXT TWO
 4136  015264  022705  022440                  CMP     #DLBUFI,R5      ;ALL DONE ??
 4137  015270  001362                          BNE     1$              ;BR IF NOT
 4138  015272  000207                          RTS     PC              ;RETURN TO CALLER
 4139
 4140                                  ;THIS ROUTINE CHECKS FOR DATA COMPARE ERRORS IN 256. BYTE BLOCK TRANSFERS
 4141
 4142  015274  042777  000104  164112  CHKDAT: BIC     #104,@DLXCSR    ;DISABLE BOTH XMIT AND RCVR INTR. ENAB.
 4143  015302  042777  000100  164100          BIC     #100,@DLRCSR
 4144  015310  012702  022040                  MOV     #DLBUFO,R2      ;R2 POINTS TO S/B DATA IN OUTPUT BUFFER
 4145  015314  004767  000070                  JSR     PC,MASKING      ;GO TO MASK OFF BITS AS A FUNCTION OF
 4146                                                                  ;CHARACTER LENGTH(5, 6, 7, OR 8 BITS)
 4147  015320  012701  022440                  MOV     #DLBUFI,R1      ;R1 POINTS TO WAS DATA IN RCVR. BUFFER
 4148  015324  122221          1$:     CMPB    (R2)+,(R1)+     ;DID S/B = WAS ??
 4149  015326  001004                          BNE     3$              ;BR IF NOT
 4150  015330  022701  023040          2$:     CMP     #BUFEND,R1      ;CHECKED ALL BYTES ??
 4151  015334  001373                          BNE     1$              ;BR IF NOT
 4152  015336  000207                          RTS     PC              ;RETURN TO CALLER
```

```
 4153  015340  013767  177776  163634  3$:     MOV     @#PSW,$TMP0     ;SAVE THE [PSW]
 4154  015346  010667  163624          3$:     MOV     SP,$REG6        ;SAVE THE [SP]
 4155  015352  114204                  3$:     MOVB    -(R2),R4        ;GET THE S/B DATA
 4156  015354  042704  177400          3$:     BIC     #177400,R4      ;CLEAR JUNK FROM HI BYTE
 4157  015360  114103                  3$:     MOVB    -(R1),R3        ;GET THE WAS DATA
 4158  015362  042703  177400          3$:     BIC     #177400,R3      ;CLEAR JUNK FROM HI BYTE
 4159  015366  004767  177254          3$:     JSR     PC,SUERR1       ;GO SET UP ERROR INFO.
 4160  015372  012767  015402  163644  3$:     MOV     #4$,$ESCAPE     ;RETURN TO 4$ AFTER ERROR PRINT
 4161  015400  104003                  3$:     ERROR+3                 ;DATA COMPARE ERROR
 4162  015402  005202                  4$:     INC     R2              ;REPOSITION BUFFER POINTERS
 4163  015404  005201                  4$:     INC     R1
 4164  015406  000750                  4$:     BR      2$              ;GO CHECK NEXT BYTE
 4165
 4166                                  ;THIS ROUTINE IS USED BY THE PATTERN TESTS
 4167                                  ;IT WILL MASK OFF THE CHARACTER SENT OUT BY THE XMITTER
 4168                                  ;BEFORE THE COMPARISON OF DATA OF WHAT WAS RECEIVED AND WHAT WAS TRANSMITTED
 4169                                  ;IS DONE.  THE MASKING IS DONE AS A FUNCTION OF CHARACTER LENGTH WHICH
 4170                                  ;CAN BE EITHER 5, 6, 7, OR 8 BITS .
 4171
 4172  015410  005005                  MASKING:        CLR     R5      ;INITIALIZE TABLE OFFSET
 4173                                                                  ;FOR PICKING UP MASK WORD
 4174
 4175  015412  022767  000010  163614          CMP     #8.,$TMP15      ;IS THE CHARACTER LENGTH 8 BITS?
 4176  015420  001427                  3$      BEQ     3$              ;BRANCH IF IT IS
 4177  015422  062705  000002          3$      ADD     #2,R5           ;SET UP FOR NEXT MASK WORD
 4178                                  3$                              ;IT COULD BE THIS ONE
 4179  015426  022767  000007  163600          CMP     #7.,$TMP15      ;IS THE CHARACTER LENGTH 7 BITS?
 4180  015434  001410                  1$      BEQ     1$              ;BRANCH IF IT IS
 4181  015436  062705  000002          1$      ADD     #2,R5           ;SET UP FOR NEXT MASK WORD
 4182                                  1$                              ;IT COULD BE THIS ONE
 4183  015442  022767  000006  163564          CMP     #6.,$TMP15      ;IS THE CHARACTER LENGTH 6 BITS?
 4184  015450  001402                  1$      BEQ     1$              ;BRANCH IF IT IS
 4185  015452  062705  000002          1$      ADD     #2,R5           ;SET UP FOR NEXT MASK WORD
 4186                                  1$                              ;IT MUST BE THIS ONE!!!!
 4187  015456  016505  015502          1$:     MOV     CHARL(R5),R5    ;PICK UP THE MASK WORD
 4188  015462  005105                  1$:     COM     R5              ;FORM THE BITS THAT ARE TO BE MASKED
 4189  015464  140522                  2$:     BICB    R5,(R2)+        ;MASK A BYTE
 4190  015466  022702  022440          2$:     CMP     #DLBUFI,R2      ;ARE WE AT THE END OF THE XMITTER
 4191                                  2$:                             ;OUTPUT BUFFER
 4192  015472  001374                  2$:     BNE     2$              ;BRANCH IF NO TO MASK NEXT BYTE
 4193  015474  012702  022040          2$:     MOV     #DLBUFO,R2      ;RESTORE R2 BEFORE RETURNING
 4194  015500  000207                  3$:     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
 4343                                  ;ERROR INFORMATION FOR ERROR MESSAGE 5
 4344
 4345  016516  046104  030461  051440  EM5:    .ASCIZ  'DL11 SOFT ERROR (PARITY,FRAMING, OR OVERRUN)'
 4346  016524  043117  020124  051105
 4347  016532  047522  020122  050050
 4348  016540  051101  052111  026131
 4349  016546  051106  046501  047111
 4350  016554  026107  047440  020122
 4351  016562  053117  051105  052522
 4352  016570  024516     000
 4353  016573     040  050050  024503  DH5:    .ASCIZ  ' (PC)    (PS)    (SP)    TEST   DEVADR  REGADR   (REG)'
 4354  016600  020040  020040  050050
 4355  016606  024523  020040  020040
 4356  016614  051450  024520  020040
 4357  016622  020040  042524  052123
 4358  016630  020040  042040  053105
 4359  016636  042101  020122  051040
 4360  016644  043505  042101  020122
 4361  016652  020040  051050  043505
 4362  016660  000051
 4363                                  .EVEN
 4364  016662  001116  001202  001176  DT5:    .WORD   $ERRPC,$TMP0,$REG6,$REG0,$REG1,$REG2,$REG3,0
 4365  016670  001162  001164  001166
 4366  016676  001170  000000
 4367
 4368                                  ;INFORMATION FOR ERROR MESSAGE 6
 4369
 4370  016702  024040  041520  020051  DH6:    .ASCIZ  ' (PC)    (PS)    (SP)   REGADR'
 4371  016710  020040  024040  051520
 4372  016716  020051  020040  024040
 4373  016724  050123  020051  020040
 4374  016732  042522  040507  051104
 4375  016740     000
 4376          016742                  .EVEN
```

```
 4377  016742  001116  001204  001176  DT6:    .WORD   $ERRPC,$TMP1,$REG6,$REG2,0
 4378  016750  001166  000000
 4379
 4380                                  ;INFORMATION FOR ERROR MESSAGE 7
 4381
 4382  016754  024040  041520  020051  DH7:    .ASCIZ  ' (PC)   DEVADR  REGADR   (REG)'
 4383  016762  020040  042504  040526
 4384  016770  051104  020040  042522
 4385  016776  040507  051104  020040
 4386  017004  024040  042522  024507
 4387  017012      000
 4388          017014                  .EVEN
 4389  017014  001116  001164  001166  DT7:    .WORD   $ERRPC,$REG1,$REG2,$REG3,0
 4390  017022  001170  000000
 4391
 4392                                  ;INFORMATION FOR ERROR MESSAGE 10
 4393
 4394  017026  024040  041520  020051  DH10:   .ASCIZ  ' (PC)   DEVADR  REGADR   (REG)    S/B'
 4395  017034  020040  042504  040526
 4396  017042  051104  020040  042522
 4397  017050  040507  051104  020040
 4398  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>'CZDLCDO DL11-C,D,E OFLNE TST'<15><12>
 4443  017424  041514  030104  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  047040          .ASCII  <15><12>'IF NOT - MODIFY THE PROPER LOCATIONS, THEN'
 4660  021716  052117  026440  046440
 4661  021724  042117  043111  020131
 4662  021732  044124  020105  051120
 4663  021740  050117  051105  046040
 4664  021746  041517  052101  047511
 4665  021754  051516  020054  044124
 4666  021762  047105
 4667  021764  005015  042522  052123          .ASCIZ  <15><12>'RESTART THE PROGRAM AT ADDRESS 200'
 4668  021772  051101  020124  044124
 4669  022000  020105  051120  043517
 4670  022006  040522  020115  052101
 4671  022014  040440  042104  042522
 4672  022022  051523  031040  030060
 4673  022030     000
 4674
 4675  022031     040  020075  041520          PCMSG:  .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  1453#  1821*  1853*  1854*  1864*  2973   2957
BASEAD  001264  1439#  1773*  1859*  1860   1867*  1875*  2985*  3001  3020*  3022
BASEIV  001270  1446#  1793*  2988*  3003   3021*  3023
BEGIN   001452  1235   1582#
BIT0  = 000001  1339#  1760   1783   1841   2298   2304   2317   2576  2660   2741  2846
BIT00 = 000001  1329#  1339
BIT01 = 000002  1328#  1338
BIT02 = 000004  1327#  1337
BIT03 = 000010  1326#  1336
BIT04 = 000020  1325#  1335
BIT05 = 000040  1324#  1334
BIT06 = 000100  1323#  1333
BIT07 = 000200  1322#  1332
BIT08 = 000400  1321#  1331   3086
BIT09 = 001000  1320#  1330   3094   3155
BIT1  = 000002  1338#  2206   2222   2268
BIT10 = 002000  1319#  3139
BIT11 = 004000  1318#  3101
BIT12 = 010000  1317#
BIT13 = 020000  1316#  3146
BIT14 = 040000  1315#  3072
BIT15 = 100000  1314#  2140   2152   2171   2183   2185   2207   2220  2223   2282
BIT2  = 000004  1337#  2012   2018   2085   2096   2109   2139   2151  2706   2804  2909
BIT3  = 000010  1336#  2170   2184
BIT4  = 000020  1335#
BIT5  = 000040  1334#  2243   2249   2267
BIT6  = 000100  1333#  2061   2067   2084   2111
BIT7  = 000200  1332#
BIT8  = 000400  1331#
BIT9  = 001000  1330#
BPTVEC= 000014  1346#
BUFEND  023040  2339   2398   2455   2512   3928   4093   4150  4688#
BUSERR  015602  1686   4230#
CHARL   015502  4187   4196#  4223
CHGD1   000246  1233#
CHGD2   000256  1237#
CHKDAT  015274  2341   2400   2457   2514   4142#
CKSUM = 104407  3071   3135   3154   3839#
CLDLBF  015112  4076   4091#
CONQUE  002672  1826   1870   1882#  1923
CR    = 000015  1254#  3728   3738
CRLF  = 000200  1255#  3699   3738
DATCHK  014520  2818   2922   4005#
DDISP = 177570  1261#  1383   1609
DEFAUL  020075  1731   4498#
DELAY   014306  2624   2711   3937#  3974
DELCNT  014350  3937*  3940   3947*  3952#
DH1     015763  1504   4268#
DH10    017026  1553   4394#
DH2     016104  1511   4287#
DH3     016254  1518   4309#
DH4     016446  1525   4334#
DH5     016573  1532   4353#
DH6     016702  1539   4370#
DH7     016754  1546   4382#
DISPLA  001142  1383#  1609*  1617*  3115*  3138*
```

```
DISPRE  000174          1222#  1617
DLADDR  010000          1680   1771   2940#  3005   3025
DLBASE  001260          1433#  1679*  1767*  1770   2940   2942*  2944   2946*  2948   2950*  2952   3001*  3022*
DLBUFI  022440          3903   4075   4102   4120   4136   4147   4190   4684#
DLBUFO  022040          4074   4091   4099   4109*  4118   4129   4144   4193   4681#
DLPRI   001302          1469#  1640*  1896*  1902*  1903*  1904*  1905*  1906*  1907   2031   2078   2262   4062
                        4064
DLRCSR  001410          1561#  1708*  1914*  1932   1991   2060   2082   2137   2140   2152   2167   2171   2185
                        2203   2207   2223   2241   2266   2347*  2354   2363   2372   2406*  2413   2422   2431
                        2463*  2470   2479   2488   2520*  2526   2534   2542   2940*  3010*  3900*  3913   3926*
                        4035   4049   4070*  4082*  4143*
DLRDBR  001412          1562#  1709   1710   1915   1916   1962   2119   2373   2432   2489   2543   2944*  3011
                        3012   3919   3923   3930   4080   4081
DLVECT  001420          1565#  1682*  1790*  1792   1793   2029   2076   2260   3003*  3023*  4060
DLXCSR  001414          1563#  1707*  1913*  1947   2001   2011   2034   2297   2316   2348*  2355   2407*  2414
                        2464*  2471   2521*  2527   2948*  3009*  3895   3901*  3925*  4069*  4083*  4142*
DLXDBR  001416          1564#  1977   2120   2952*  3905*
DSWR  = 177570          1260#  1382   1608
DT1     016042          1505   4277#
DT10    017074          1554   4402#
DT2     016202          1512   4299#
DT3     016352          1519   4321#
DT4     016504          1526   4340#
DT5     016662          1533   4364#
DT6     016742          1540   4377#
DT7     017014          1547   4389#
DUT     014516          3982*  3984*  3988*  3995   4000#
EMTVEC= 000030          1349#  1593*  1594*
EM1     015714          1503   1538   4261#
EM2     016060          1510   4283#
EM3     016224          1517   1552   4305#
EM4     016374          1524   4230*  4231*  4232*  4237*  4238*  4239*  4327#
EM5     016516          1531   1545   4345#
ERRVEC= 000004          1342#  1606   1607*  1618*  1686*  1687*  1930   1931*  1939*  1945   1946*  1954*  1960
                        1961*  1969*  1975   1976*  1984*  2590   2591*  2600*  2602*  2673   2674*  2683*  2685*
                        2754   2755*  2764*  2766*  2859   2860*  2869*  2871*  3077   3078*  3080*  3083*
FAILSA  021453          1663   4631#
FIRSTD  002244          1749#  1801
FLUSH   002770          1736   1913#
FOULUP  020670          2975   4565#
GNS   = ******  U       3831   3832   3833   3834   3835   3837   3839   3840   3841   3842   3843
GO      002144          1706   1720#  1798
GTSWR = 104406          1625   3837#
HOWMAN  002442          1794   1805#
HT    = 000011          1252#  3697   3738
INTFLG  001446          1576#  2027*  2036   2048*  2079*  2091   2112*  2264*  2269   2280*
IOTVEC= 000020          1347#  1591*  1592*
IPTR    001434          1571#  2339   2398   2455   2512   3928   3930*  3931*  4075*
KEEPAD  001262          1435#  1770*  1773   1867   1875   3020
KEEPIV  001266          1442#  1792*  3021
LASTAD  001300          1465#  1848*  1860
LASTD   002512          1820   1827#  1920
LDOUT   001436          1572#  2331*  2390*  2447*  2504*  4077
LDOUT1  015130          2331   4099#
LDOUT2  015152          2390   4108#
LDOUT3  015172          2447   4117#
```

```
LDOUT4  015226            2504    4129#
LENGTH  020013            1720    2784    2875    4489#
LESS1   001304            1471#   1907*   1909*   1911*
LF    = 000012            1253#   3732    3738
LINTAD  021102            2565    2649    2730    2835    4590#
MASKIN  015410            4145    4172#
MFIRST  020202            1749    4511#
MLASTD  020433            1830    4537#
MRANGE  020531            1877    4548#
MULDEV  020346            1805    4528#
MULTD   001272            1449#   1631*   1812*   1815*   1818    2970    2980*
MVECT   020270            1774    4520#
NUMONE  014440            3957    3961*   3968    3977#
NUMTWO  014442            3960    3964    3967*   3978#
ONCE    002120            1667    1696    1707#
OPTR    001432            1570#   3903    3905    3906*   4074*
PCMSG   022031            1665    4675#
PIRQ  = 177772            1259#
PIRQVE= 000240            1353#
PLEVEL  020615            1888    4557#
PRG1    001734            1224    1631#
PRG2    006424            1226    2562#
PRG2A   006432            2565#   2635
PRG2B   006552            2607#
PRG3    006632            1228    2646#
PRG3A   006640            2649#   2717
PRG3B   006760            2690#
PRG4    007050            1230    2727#
PRG4A   007056            2730#   2822
PRG4B   007210            2776#
PRG4C   007222            2784#   2825
PRG5    007410            1232    2832#
PRG5A   007416            2835#   2927
PRG5B   007606            2894#   2924
PRG5C   007536            2875#   2930
PRIME   015002            2332    2391    2448    2505    4069#
PROG2M  017460            2563    4449#
PROG3M  017524            2647    4455#
PROG4M  017570            2728    4461#
PROG5M  017634            2833    4467#
PR0   = 000000            1276#
PR1   = 000040            1277#
PR2   = 000100            1278#
PR3   = 000140            1279#
PR4   = 000200            1280#
PR5   = 000240            1281#
PR6   = 000300            1282#
PR7   = 000340            1283#
PS    = 177776            1234*   1256#   1257
PSW   = 177776            1257#   2095    3897    3915    3921    4034    4047    4052    4153
PWRVEC= 000024            1348#   1597*   1598*   3849*   3850*   3859*   3865*   3877*   3878*
RDB     017747            2810    2914    4480#
RDCHR = 104410            3558    3840#
RDDEC = 104413            1722    2786    2877    3843#
RDLIN = 104411            3633    3758    3841#
RDOCT = 104412            1733    1752    1775    1807    1879    1889    2568    2609    2617    2652    2692    2700
```

```
                    2733    2772    2778    2838    2887    3842#
REDO1   006622      2573    2575    2578    2585    2601    2634#
REDO2   007040      2657    2659    2662    2669    2684    2716#
REDO3   007370      2738    2740    2743    2750    2765    2821#
REDO3A  007400      2790    2792    2823#
REDO4   007760      2843    2845    2848    2855    2870    2926#
REDO4A  007770      2881    2883    2928#
RESTRT  002036      1685#   2982    3051    3882    4251
RESVEC= 000010      1343#   1688*   1689*
RETRY   002412      1726    1728    1796#
RETRY0  002422      1757    1759    1762    1766    1799#
RETRY1  002432      1780    1782    1785    1789    1802#
RETRY2  003014      1838    1840    1843    1847    1918#
RETRY3  003024      1893    1895    1921#
RFLG0   001424      1567#   2335    2394    2451    2508    3917*   4072*
RFLG1   001426      1568#   2337    2396    2453    2510    3924*   4073*
RINT    014164      3913#   4061
ROTADD  001276      1460#   1824*   1852*   1854    1857*   1864    1865*   2992*   2997    3018*
RSTALL  021364      2768    2886    4621#
RSVERR  015626      1688    4237#
RTRY    001430      1569#   2330*   2378*   2379    2389*   2437*   2438    2446*   2494*   2495    2503*   2548*   2549
SELCAR  021163      2607    2690    2776    4599#
SELDLY  021271      2614    2697    4611#
STACK   001100      1247#   1589    1685
STAD    001450      1224*   1226*   1228*   1230*   1232*   1577#   1630
STALL   014352      2799    2904    3957#
STCONT  000246      1225    1227    1229    1231    1234#
STKLMT= 177774      1258#
STLMSK  001306      1477#   3969
STMES   017417      1662    1698    4442#
SUERR1  014646      2102    2128    2358    2367    2375    2417    2426    2434    2474    2483    2491    2530    2538
                    2545    4039#   4159
SUERT1  014700      1935    1950    1965    1980    4047#
SUERT2  014726      2596    2679    2760    2865    4052#
SUER2   014620      1994    2004    2015    2021    2043    2053    2064    2070    2117    2142    2148    2154    2158
                    2173    2179    2187    2194    2209    2216    2225    2232    2246    2252    2273    2284    2288
                    2301    2307    2321    4034#
SUVEC   014754      2329    2388    2445    2502    4060#
SWR     001140      1382#   1587    1608*   1610    1616*   1623    1642    1702    2134    2165    2201    2239    2258
                    2294    3072    3086    3088    3094    3101    3139    3146    3151    3155    3402    3441    3496*
                    3857    3870*
SWREG   000176      1223#   1616    1623    3402    3441    3464
SW0     = 000001    1311#   1702
SW00    = 000001    1301#   1311
SW01    = 000002    1300#   1310
SW02    = 000004    1299#   1309
SW03    = 000010    1298#   1308
SW04    = 000020    1297#   1307
SW05    = 000040    1296#   1306
SW06    = 000100    1295#   1305
SW07    = 000200    1294#   1304
SW08    = 000400    1293#   1303
SW09    = 001000    1292#   1302
SW1     = 000002    1310#
SW10    = 002000    1291#
SW11    = 004000    1290#
```

```
SW12  = 010000  1289#  2134  2165  2201  2239  2258  2294
SW13    020000  1288#
SW14  = 040000  1287#
SW15  = 100000  1286#
SW2   = 000004  1309#
SW3   = 000010  1308#
SW4   = 000020  1307#
SW5   = 000040  1306#
SW6   = 000100  1305#
SW7   = 000200  1304#
SW8   = 000400  1303#  1642
SW9   = 001000  1302#
TABFLG  001256  1429#  1633*  1683*  1695  1700*  2981*
TBITVE= 000014  1344#
TCONT   014470  3986  3990#
TIMERX  014444  2808  2913  3982#
TIMETX  014462  2800  2905  3988#
TIMR1   001440  1573#  2343*  2402*  2459*  2516*  4078*
TIMR2   001442  1574#  2345*  2404*  2461*  2518*  4079*
TIMR3   001444  1575#  2028*  2040*
TKVEC = 000060  1351#  3379*  3380*
TPVEC = 000064  1352#
TRAPVE= 000034  1350#  1595*  1596*
TRPCOM  015652  4233  4240  4244#
TRTVEC= 000014  1345#
TST1    003034  1711  1917  1929#  3013
TST10   003404  2020  2026#
TST11   003544  2047  2052  2058#
TST12   003614  2069  2075#
TST13   004076  2108  2127  2133#
TST14   004226  2135  2157  2164#
TST15   004354  2166  2193  2200#
TST16   004502  2202  2231  2238#
TST17   004562  2240  2251  2257#
TST2    003102  1944#
TST20   004724  2259  2278  2287  2293#
TST21   005006  2295  2306  2314#
TST22   005040  2320  2327#
TST23   005332  2342  2353  2362  2371  2386#
TST24   005624  2401  2412  2421  2430  2443#
TST25   006116  2458  2469  2478  2487  2500#
TST3    003150  1959#
TST4    003216  1974#
TST5    003264  1989#
TST6    003306  1993  1999#
TST7    003332  2003  2009#
TYPDS = 104405  3040  3835#
TYPE  = 104401  1662  1663  1665  1698  1720  1731  1749  1774  1796  1799  1802  1805  1830
                1877  1888  1918  1921  2349  2408  2465  2522  2563  2565  2607  2614  2634
                2647  2649  2690  2697  2716  2728  2730  2768  2776  2784  2802  2810  2821
                2823  2833  2835  2875  2886  2907  2914  2926  2928  2975  3038  3041  3141
                3149  3172  3189  3191  3194  3196  3200  3207  3272  3348  3396  3408  3462
                3463  3466  3477  3487  3498  3517  3565  3571  3576  3580  3585  3586  3588
                3591  3595  3662  3664  3702  3795  3797  3831#  3879
TYPOC = 104402  3180  3204  3465  3832#
TYPON = 104404  3834#
```

```
TYPOS = 104403          3833#
UPMASK  015512          2087    2121    4017    4207#
VECT    002330          1774#   1804
XDB     017700          2802    2907    4473#
XFLGO   001422          1566#   2333    2392    2449    2506    3899*   4071*
XINT    014076          3895#   4063
XMSG1   017110          2349    4406#
XMSG2   017167          2408    4414#
XMSG3   017250          2465    4423#
XMSG4   017333          2522    4432#
$AUTOB  001134          1379#   1627*   3455    3612
$BDADR  001122          1374#
$BDDAT  001126          1376#
$BELL   001246          1420#   3141    3163    3408    3605
$CHARC  013446          3704*   3714*   3721    3730*   3735#
$CKSWR  012132          3441#   3839
$CMTAG  001100          1362#   1584    1585    1593    1599    1600
$CM1    = 000010        1394#   1395#   1396#   1397#   1398#   1399#   1400#   1401#   1402#
$CM2    = 000020        1394#   1395#   1396#   1397#   1398#   1399#   1400#   1401#   1402#
$CM3    = 000010        1392#   1394
$CM4    = 000020        1402#   1403#   1404#   1405#   1406#   1407#   1408#   1409#   1410#   1411#   1412#   1413#   1414#
                        1415#   1416#   1417#   1418#
$CNTLC  013030          3396    3477    3605#
$CNTLG  013042          3462    3607#
$CNTLU  013035          3487    3580    3606#
$CRLF   001253          1422#   3149    3163    3172    3191    3196    3200    3498    3585    3605    3667    3703    3738
                        3799
$DBLK   011616          3314    3348    3356#
$DOAGN  010362          3034    3043    3049#
$DTBL   011606          3317    3352#
$ENDAD  010352          3045#
$ENDCT  010320          3036#
$ENDMG  010371          3038    3053#
$ENULL  010366          3041    3052#
$EOP    010054          2515    2525    2533    2541    2551    2964#
$EOPCT  010312          3033#   3037
$ERFLG  001103          1365#   3061    3090    3092    3098*   3119    3136*   3163
$ERMAX  001115          1371#   1601*   3092    3114*   3119
$ERROR  010660          1593    3134#
$ERRPC  001116          1372#   3143*   3144*   3145    3163    3178    4277    4299    4321    4364    4377    4389    4402
$ERRTB  001310          1499#   3186
$ERRTY  011020          3148    3171#
$ERTTL  001112          1369#   3142*   3163
$ESCAP  001244          1419#   1600*   1936*   1951*   1966*   1981*   2044*   2103*   2275*   2350*   2359*   2368*   2376*
                        2409*   2418*   2427*   2435*   2466*   2475*   2484*   2492*   2523*   2531*   2539*   2546*   2597*
                        2680*   2761*   2866*   3113*   3158    3160    3163    4160*   4248*
$FILLC  001156          1390#   3707    3738
$FILLS  001155          1389#   3738
$GDADR  001120          1373#
$GDDAT  001124          1375#
$GET42  010342          3042#
$GTSWR  012222          3463#   3837
$HD     = 000001        1201    1202
$HIOCT  013230          3655*   3666#
$ICNT   001104          1366#   3105*   3106    3108*   3118
$ILLUP  014060          3849    3865    3884#
```

```
$INTAG  001135            1380#  3460*  3479   3499   3612
$ITEMB  001114            1370#  3145*  3163   3175
$LF     001254            1423#  3163   3595   3605   3667   3738   3799
$LPADR  001106            1367#  1602*  3096*  3111*  3116   3118
$LPERR  001110            1368#  1603*  3096   3112*  3118   3157
$PWFIL = ****** U         1620   1623   3111   3151   3691
$PWNEW  013060            3466   3610#
$PWSUR  013047            3463   3608#
$PWXCNT 010656            3109   3118#
$NULL   001154            1388#  3709   3738
$NWTST= 000001            1926#  1941#  1956#  1971#  1986#  1996#  2006#  2023#  2055#  2072#  2130#  2161#  2197#
                          2235#  2254#  2290#  2311#  2324#  2383#  2440#  2497#
$OCNT   011376            3244*  3273*  3286#
$OMODE  011400            3239*  3243*  3248   3251*  3262*  3288#
$OVER   010642            3073   3089   3097   3107   3115#
$PASS   001100            1363#  3030*  3031*  3039   3052   3103   3119
$POWER  014066            3880   3887#
$PWRAD  014054            3882#
$PWRDN  013714            1597   3849#  3877
$PWRMG  014050            3880#
$PWRUP  013766            3859   3865#
$QUES   001252            1421#  1796   1799   1802   1918   1921   2634   2716   2821   2823   2926   2928   3163
                          3517   3588   3605   3664   3667   3738   3797   3799
$RDCHR  012474            3530#  3840
$RDDEC  013452            3753#  3843
$RDLIN  012564            3553#  3841
$RDOCT  013072            3628#  3842
$RDSZ = 000010            3546#
$REGAD  001160            1392#
$REG0   001162            1394#  4040*  4050*  4246*  4277   4299   4321   4340   4364
$REG1   001164            1395#  4009*  4011*  4041*  4051*  4277   4299   4321   4364   4389   4402
$REG2   001166            1396#  4007*  4009   4042*  4055*  4277   4299   4321   4364   4377   4389   4402
$REG3   001170            1397#  4013*  4043*  4299   4321   4364   4389   4402
$REG4   001172            1398#  4022*  4044*  4299   4321   4402
$REG5   001174            1399#
$REG6   001176            1400#  2098*  3898*  3916*  3922*  4037*  4038*  4053*  4054*  4154*  4244*  4277   4299
                          4321   4340   4364   4377
$REG7   001200            1401#  2159*  4249*  4340
$RTNAD  010364            3051#
$R2A  = ****** U          3844
$SAVRE= ****** U          3844
$SAVR6  014064            3858*  3866   3867*  3868*  3886#
$SCOPE  010406            1591   3070#
$SETUP= 000117            1581#  1590   1591   1593   1595   1597   1599   1600   1602   1620   3028   3071   3135
                          3154   3162   3400   3405   3406   3436   3612
$STUP = 177777            1581#
$SVLAD  010614            3081   3110#
$SWR  = 167400            1188#  1201   1209   1210   1211   1212   1213   1214   1215   1216   1418   1419   1420
                          1599   1600   1602   1603   1930   1945   1960   1975   1990   2000   2010   2027   2059
                          2076   2134   2165   2201   2239   2258   2294   2315   2328   2387   2444   2501   2961
                          3029   3044   3050   3052   3062   3063   3064   3065   3066   3072   3084   3086   3087
                          3090   3091   3092   3099   3100   3101   3112   3115   3118   3126   3127   3128   3129
                          3130   3139   3146   3151   3155   3163   3883
$SWRMK= 000000            1216   1217   3066   3067   3088
$TIMES  001242            1418#  1599*  2328*  2387*  2444*  2501*  3029*  3099*  3106   3109*  3118
$TKB    001146            1385#  3361   3381   3392   3417   3445   3472
```

```
$TKCNT  011626          3362#  3376*  3406   3423*  3537   3539*
$TKINT  011644          1629   3376#  3397   3458
$TKQEN= 011644          3366#  3431   3542
$TKQIN  011630          3363#  3377*  3378   3429*  3430*  3431   3433*
$TKQOU  011632          3364#  3378*  3540   3541*  3542   3544*
$TKQSR  011634          3365#  3377   3433   3544
$TKS    001144          1384#  3361   3382*  3413*  3415   3421*  3443   3459*  3469   3481*  3501*
$TKSRV  011714          3379   3392#
$TMP0   001202          1402#  2095*  2586*  2622*  2670*  2709*  2751*  2795   2806*  2856*  2900   2911*  3897*
                        3915*  3921*  4034*  4047*  4153*  4247*  4277   4299   4321   4340   4364
$TMP1   001204          1403#  2081*  2086*  2110*  2374   2433   2490   2544   2611*  2622   2694*  2709   2780*
                        2806   2898*  2911   3923*  4052*  4207   4377
$TMP10  001222          1410#
$TMP11  001224          1411#  2893*  2894   2897*
$TMP12  001226          1412#  2894*  2895*  2896*  2897   2898
$TMP13  001230          1413#
$TMP14  001232          1414#  2089   2090   2124   4019   4022   4207*  4225*
$TMP15  001234          1415#  1634*  1729*  2793*  2884*  4175   4179   4183   4211   4215   4219
$TMP16  001236          1416#
$TMP17  001240          1417#
$TMP2   001206          1404#  2619*  2620   2702*  2707   2774*  2797   2889*  2902
$TMP3   001210          1405#  2795*  2796*  2804*  2812   2900*  2901*  2909*  2916   3982   3988
$TMP4   001212          1406#  2812*  2814*  2816   2916*  2918*  2920   4007
$TMP5   001214          1407#  2816*  2920*  4005   4013   4019
$TMP6   001216          1408#  3990*  3991*  4005*  4015
$TMP7   001220          1409#  4117*  4119   4122*  4130*  4131   4132*  4133   4134*  4135*
$TN   = 000026          1201#  1926   1930#  1941   1945#  1956   1960#  1971   1975#  1986   1990#  1993   1996
                        2000#  2003   2006   2010#  2020   2023   2027#  2047   2052   2055   2059#  2069   2072
                        2076#  2108   2127   2130   2134#  2135   2157   2161   2165#  2166   2193   2197   2201#
                        2202   2231   2235   2239#  2240   2251   2254   2258#  2259   2278   2287   2290   2294#
                        2295   2306   2311   2315#  2320   2324   2328#  2342   2353   2362   2371   2383   2387#
                        2401   2412   2421   2430   2440   2444#  2458   2469   2478   2487   2497   2501#
$TPB    001152          1387#  3727*  3738
$TPFLG  001157          1391#  3685   3738
$TPS    001150          1386#  3725   3738
$TRAP   013630          1595   3808#
$TRAP2  013652          3819#  3830
$TRP  = 000014          3823#  3832#  3833#  3834#  3835#  3836#  3837   3838#  3839   3840#  3841#  3842#  3843#
                        3844#
$TRPAD  013664          3813   3830#
$TSTNPI 001102          1364#  3007*  3028*  3061   3088   3110*  3115   3119   3138   3163   4039   4048   4245
$TTYIN  013020          3555   3556   3568   3586   3600   3604#
$TYPBN= ****** U        3836
$TYPDS  011402          3302#  3835
$TYPE   013232          3685#  3823   3831
$TYPEC  013402          3503   3706   3713   3720   3725#  3726
$TYPEX  013450          3731   3733   3736#
$TYPOC  011200          3242#  3832
$TYPON  011214          3241   3244#  3834
$TYPOS  011154          3237#  3833
$XTSTR  010420          3075#
$$GET4= 000000          3044#
$OFILL  011377          3238*  3242*  3252   3287#
$40CAT= ****** U        3072   3148
.     = 023042          1221#  1237#  1239#  1361#  1424   1588   1602   1603   2159   3052   3056   3118   3119
                        3163   3210#  3356#  3361   3365#  3366   3367   3604#  3605   3612#  3667   3738   3799
```

3861 3885 4298# 4376# 4388# 4681# 4684#

```
COMMEN      1#  1354#
ENDCOM      1#  1354#
EOPBEG   1204#  2965
ERROR    1248#  1664  1937  1952  1967  1982  1995  2005  2016  2022  2045  2054  2065  2071  2104
         2118   2129  2143  2149  2155  2160  2174  2180  2188  2195  2210  2217  2227  2233  2247
         2253   2276  2285  2289  2302  2308  2322  2351  2360  2369  2377  2410  2419  2428  2436
         2467   2476  2485  2493  2524  2532  2540  2547  2598  2681  2762  2803  2811  2867  2908
         2915   4024  4026  4161  4250
ESCAPE      1#  1354#
GETPRI      1#  1354#
GETSWR      1#  1354#  1620
MORETA   1203#  1425
MULT        1#  1354#
NEWTST      1#  1354#  1926  1941  1956  1971  1986  1996  2006  2023  2055  2072  2130  2161  2197
         2235   2254  2290  2311  2324  2383  2440  2497
POP         1#  1189#  1354#  3343  3656  3788  3870  3871
PUSH        1#  1189#  1354#  3302  3630  3755  3851  3857
REPORT      1#  1354#
SCOPE    1249#  1929  1944  1959  1974  1989  1999  2009  2026  2058  2075  2133  2164  2200  2238
         2257   2293  2314  2327  2386  2443  2500  2969
SETPRI      1#  1354#  3533
SETTRA   3823#  3832  3833  3834  3835  3837  3839  3840  3841  3842  3843
SETUP       1#  1189#  1354#  1583
SKIP        1#  1354#  1934  1949  1964  1979  1993  2003  2014  2020  2046  2052  2063  2069  2108
         2116   2127  2135  2141  2147  2153  2157  2166  2172  2178  2186  2193  2202  2208  2215
         2224   2231  2240  2245  2251  2259  2277  2283  2287  2295  2300  2306  2320  2342  2352
         2361   2370  2401  2411  2420  2429  2458  2468  2477  2486
SLASH       1#  1354#
SPACE    1354#
STARS       1#  1189#  1354#  1357  1424  1557  1559  1926  1928  1941  1943  1956  1958  1971  1973
         1986   1988  1996  1998  2006  2008  2023  2025  2055  2057  2072  2074  2130  2132  2161
         2163   2197  2199  2235  2237  2254  2256  2290  2292  2311  2313  2324  2326  2383  2385
         2440   2442  2497  2499  2958  3027  3058  3122  3166  3214  3292  3360  3436  3451  3522
         3546   3616  3670  3741  3802  3847  3863  3891  3893  3909  3911  4029  4031  4255  4257
SWRSU       1#  1354#  1604#
TRMTRP   3823#
TYPBIN      1#  1354#
TYPDEC      1#  1354#  3039
TYPNAM      1#  1354#
TYPNUM      1#  1354#
TYPOCS      1#  1354#
TYPOCT      1#  1354#  3178  3202  3464
TYPTXT      1#  1354#
$$CMRE   1355#  1394  1395  1396  1397  1398  1399  1400  1401
$$CMTM   1355#  1402  1403  1404  1405  1406  1407  1408  1409  1410  1411  1412  1413  1414  1415
         1416   1417
$$ESCA      1#  1354#
$$NEWT      1#  1354#  1926  1941  1956  1971  1986  1996  2006  2023  2055  2072  2130  2161  2197
         2235   2254  2290  2311  2324  2383  2440  2497
$$SET    3823#  3832  3833  3834  3835  3837  3839  3840  3841  3842  3843
$$SKIP      1#  1354#  1993  2003  2020  2047  2052  2069  2108  2127  2135  2157  2166  2193  2202
         2231   2240  2251  2259  2278  2287  2295  2306  2320  2342  2353  2362  2371  2401  2412
         2421   2430  2458  2469  2478  2487
.EQUAT      1#  1189#  1244
.HEADE      1#  1189#  1191
.KT11       1#
```

```
.SETUP      1#   1189#   1581
.SURHI      1#   1189#   1205
.SURLO   1189#   1217#   1218
.$ACT1      1#
.$APTB      1#
.$APTH      1#
.$APTY      1#
.$ASTA      1#
.$CATC      1#   1189#
.$CMTA      1#   1189#   1355
.$DB2D      1#
.$DB2O      1#
.$DIV       1#
.$EOP       1#   1189#   2956
.$ERRO      1#   1189#   3120
.$ERRT      1#   1189#   3164
.$MULT      1#
.$POWE      1#   1189#   3845
.$RAND      1#
.$RDDE      1#   1189#   3739
.$RDOC      1#   1189#   3614
.$READ      1#   1189#   3358
.$R2AZ      1#
.$SAVE      1#
.$SB2D      1#
.$SB2O      1#
.$SCOP      1#   1189#   3056
.$SIZE      1#
.$SUPR      1#
.$TRAP      1#   1189#   3800
.$TYPB      1#
.$TYPD      1#   1189#   3290
.$TYPE      1#   1189#   3668
.$TYPO      1#   1189#   3212
.$40CA      1#
.1170       1#


. ABS.  023042     000


 ERRORS DETECTED:  0

 CZDLCD.BIN,CZDLCD.LST/CRF/SOL/NL:TOC=CZDLCD.SML,CZDLCD.P11
 RUN-TIME: 43 60 3 SECONDS
 RUN-TIME RATIO: 164/107=1.5
 CORE USED:  34K  (67 PAGES)
```