DR11

DR11 GEN NPR INTFC
CZDRLCO

AH-E780C-MC
FICHE 1 OF 1

```
.REM    @

        IDENTIFICATION


PRODUCT CODE:           AC-E779C-MC

PRODUCT NAME:           CZDRLCO DR11 GEN NPR INTFC

DATE RELEASED:          OCTOBER, 1980

MAINTAINER:             DIAGNOSTIC ENGINEERING

AUTHOR:                 DAN P. MILLEVILLE
```

```
34
35                                          HISTORY
36
37                          REV             DATE            NOTE
38                          A               1977            INITIAL RELEASE
39                          B               1980            CORRECTION OF CODING ERRORS
40                          C               1980            11/05 PROBLEM AND ENDLESS LOOP PROBLEM FIXED
```

TABLE OF CONTENTS

```
                        10.0    SUBROUTINE DESCRIPTIONS

                                10.1    READ
                                10.2    ERCAPT
                                10.3    FTCAPT
                                10.4    FIXTBL
                                10.5    LODBUF
                                10.6    CHKBFF
                                10.7    INTA
                                10.8    DATCHK
                                10.9    CLENUP
                                10.10   CHKCAB
                                10.11   DATOCK
                                10.12   ERRCHK
                                10.13   DEVADS
                                10.14   BPINIT
                                10.15   DRGET
                                10.16   TYPCNF
                                10.17   CHK4DR
                                10.18   ASIZE
                                10.19   VCTADS
                                10.20   CATCH
                                10.21   PSTATE
                                10.22   PNTPRI
                                10.23   SETTUP

                        11.0    DATA STACKS

                                11.1    PATRNS
                                11.2    EXPAT0
                                11.3    EXPAT1
```

```
117                             1.0     ABSTRACT
118
119                                     THIS DIAGNOSTIC PROGRAM IS CAPABLE OF TESTING THE DR11-W
120                                     NPR GENERAL INTERFACE IN DR11-W OR DR11-B MODE.
121
122                                     IT HAS THE FOLLOWING FEATURES:
123
124                                     1.  APT11/XXDP COMPATIBLE
125                                     2.  MULTIPLE BOARD TESTING USING TABLE CREATED BY USER
126                                     3.  BURST DATA LATE CALIBRATION
127                                     4.  INDEPENDENT 'LOGIC WRAP-AROUND' AND 'CABLE WRAP-AROUND' TESTING
```

2.0 REQUIREMENTS

2.1 EQUIPMENT

1. PDP11 STANDARD COMPUTER
2. I/O TYPE TERMINAL
3. 1-16 DR11-W MODULE(S)
4. LOOP BACK CABLE (NEEDED TO FULLY CHECK THE MODULE WITH THIS DIAGNOSTIC)

2.2 HARDWARE SWITCH SETTINGS

THE ADDRESS SELECTION SWITCH, E120, IS SET UP AS BELOW:

	1	2	3	4	5	6	7	8	9	10
ADDRESS BITS:	12	11	10	9	8	7	6	5	4	3

EXAMPLE: DEVICE ADDRESS 172410, SWITCHES 1, 3, 5 & 10 SHOULD BE OFF, AND ALL OTHERS SHOULD BE ON.

THE E105 SWITCHPACK: THIS SWITCHPACK MUST BE IN THE FOLLOWING POSITIONS TO RUN THIS DIAGNOSTIC:

1 - OFF
2 - ON
3 - OFF
4 - OFF
5 - ON FOR -W MODE, OFF FOR -B MODE

SINGLE SWITCH NEAR THE E105 SWITCHPACK:

2 CYCLE MODE - SWITCH HANDLE TOWARDS PACK E105
N CYCLE MODE - SWITCH HANDLE TOWARDS E94

THE VECTOR SELECTION SWITCH, E15, IS SET UP AS BELOW:

	1	2	3	4	5	6	7	8
VECTOR BITS:	1	2	3	4	5	6	7	8

EXAMPLE: VECTOR ADDRESS 300, SWITCHES 6 & 7 SHOULD BE OFF, AND ALL OTHERS SHOULD BE ON.

2.3 STORAGE

THE PROGRAM USES APPROX. 52200 WORDS OF MEMORY

3.0 TESTING MODE

3.1 DEFINITION

THE DR11-W DIAGNOSTIC ACCOMPLISHES DEVICE REGISTER BIT TESTS, INTERNAL "LOGIC" WRAP-AROUND TESTS, AND WITH THE BC06-R WRAP-AROUND CABLE IN J1 AND J2, PROVIDES EXTERNAL "CABLE" WRAP-AROUND TESTS. IN ORDER TO FULLY CHECK THE MODULE, THE DIAGNOSTIC MUST BE RUN WITH AND WITHOUT THE WRAPAROUND CABLE IN PLACE, RESTARTING AT ADDRESS 200 EACH TIME, OR EDITING TO CHANGE THE CABLE MODE (SEE SECT. 7.1.1)

THERE ARE ONLY TWO LEGAL MODES OF OPERATION OF THIS DIAGNOSTIC:

1. DR11 WITH NO CABLE(S) IN USER SLOTS.
2. DR11 WITH WRAP-AROUND CABLE FROM J1 TO J2.

THIS DIAGNOSTIC IS NOT MEANT TO BE RUN IN THE FOLLOWING MODES:

1. DR11 CONNECTED TO ANOTHER DR11.
2. DR11 CONNECTED TO A USER DEVICE.

3.2 IMPLEMENTATION

DEVICE REGISTER BIT TESTS AND INTERNAL LOGIC WRAP-AROUND TESTS ARE EXECUTED UNCONDITIONALLY. CABLE WRAP-AROUND TESTS ARE EXECUTED ONLY IF THE BC06-R CABLE IS IN PLACE BETWEEN THE J1 AND J2 CONNECTORS ON THE DR11-W UNDER TEST. THE PRESENCE OF THIS CABLE IS "SIZED" FOR AUTOMATICALLY FOR EACH BOARD WHEN THE DIAGNOSTIC IS STARTED AT ADDRESS 200. THE USER *MUST* VERIFY THAT THE "SIZING" OCCURRED CORRECTLY BY OBSERVING THE OUTPUT OF THE PROGRAM WHEN STARTING AT 200. (REFER TO SECTION 5.1 FOR EXAMPLE) IF THIS SUMMARY IS NOT NEEDED, RAISE BIT 12 OF THE SWITCH REGISTER BEFORE PROGRAM EXECUTION.

IN MANUAL MODE (STARTING ADDRESS = 204), THE USER CAN FORCE UNIFORM TESTING PARAMETERS FOR ALL MODULES THROUGH USE OF THE EDIT FUNCTION (REFER TO SECTION 7.1.1).

```
215
216                          4.0     LOAD AND START PROCEDURE
217
218                                  1.  LOAD PROGRAM INTO MEMORY.
219                                  2.  LOAD STARTING ADDRESS 200, 204 OR 210.  (SEE SECTS. 7.1, 7.2, 7.3
220                                      RESPECTIVELY)
221                                  3.  PRESS START.
```

5.0 SWITCH REGISTER

5.1 OPTIONS

SWITCH	OCTAL	FUNCTION
SW15=1	100000	HALT ON ERROR

THIS WILL CAUSE THE PROCESSOR TO HALT AT THE NEXT ERROR.

SW14=1	040000	LOOP ON TEST

THIS WILL CAUSE THE PROCESSOR TO LOOP ON THE TEST IT IS THEN EXECUTING.

SW13=1	020000	INHIBIT ERROR TYPEOUTS

THIS WILL CAUSE ERROR TYPEOUTS TO BE INHIBITED.

SW12=1	010000	DO NOT PRINT BOARD CONFIGURATION

THIS WILL CAUSE THE LIST OF ALL BOARDS AND THEIR SETUP DATA THAT THE AUTOSIZE ROUTINE FOUND TO NOT PRINT.

SW11=1	004000	TEST NUMBER TRACE ENABLING

THIS ENABLES THE PRINTING OF THE FOLLOWING AT THE BEGINNING OF EACH TEST:

T # XX

THIS CAN BE USED WHEN AN UNEXPECTED TRAP OCCURS IN A TEST, BUT LOOPING ON THAT TEST RESULTS IN NO ERROR(S).

SW10=1	002000	BELL ON ERROR

THIS FUNCTION CAUSES THE TERMINAL BELL TO SOUND WHEN AN ERROR OCCURS. THIS CAN BE USED IN CONJUNCTION WITH LOOP-ON-TEST AND INHIBIT-ERROR-TYPEOUTS TO SEE IF A LOOSE CONNECTION MAY BE CAUSING THE ERROR.

SW09=1	001000	LOOP ON ERROR

THIS FUNCTION WILL CAUSE LOOPING ON ERROR. IT CAN BE USED IN CONJUNCTION WITH INHIBIT-ERROR-TYPEOUTS WHEN USING A SCOPE TO FIND A FAULTY COMPONENT.

```
                        SW08=1          000400          LOOP ON TEST IN SWR<6:0>

                                THIS FUNCTION CAUSES THE CPU TO JUMP TO THE
                                TEST IN BITS <6:0> AND EXECUTE THAT TEST UNCON-
                                DITIONALY.  CHANGE THE SWITCH REGISTER TO EXIT.
                                TO CREATE A TIGHTER LOOP ON THAT PARTICULAR
                                TEST, SET LOOP-ON-TEST (40000) IN THE SWR ONCE
                                THE TEST IS EXECUTING.

                        SW07=1          000200          INHIBIT MULTIPLE ERROR TYPEOUTS

                                ON ERROR CALLS IN LOOPS WHERE MULTIPLE ERRORS
                                ARE POSSIBLE, THIS FUNCTION INHIBITS ANY ADDI-
                                TIONAL DATA THAT MAY PRINT IN THAT LOOP.
                                EXAMPLE:

                                MULTIPLE TYPEOUTS ENABLED:

                                  [ERROR MESSAGE]
                                  [DATA HEADER]
                                  XXXXXX  XXXXXX  XXXXXX  XXXXXX
                                  XXXXXX  XXXXXX  XXXXXX  XXXXXX
                                  XXXXXX  XXXXXX  XXXXXX  XXXXXX
                                  XXXXXX  XXXXXX  XXXXXX  XXXXXX
                                  XXXXXX  XXXXXX  XXXXXX  XXXXXX

                                >>>>>>NOTE<<<<<<
                                  A MAXIMUM OF 17 (OCTAL) DATA LINES WILL
                                  PRINT.  IF THERE ARE MORE, A MESSAGE WILL
                                  PRINT AS FOLLOWS:

                                    THERE ARE STILL MORE ERRORS, BUT WILL NOT BE PRINTED.
                                    ERRORS WILL STILL BE COUNTED AND PRINTED AT THE EOP.

                                MULTIPLE TYPEOUTS DISABLED:

                                  [ERROR MESSAGE]
                                  [DATA HEADER]
                                  XXXXXX  XXXXXX  XXXXXX  XXXXXX

                                  (NO MORE DATA WILL PRINT) THE TOTAL NUMBER
                                  OF ERRORS WILL STILL BE TOTALED AND PRINTED
                                  AT THE EOP OR EOD.
```

5.2 SOFTWARE SWITCH REGISTER

IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST, OR IF ONE DOES AND IT CONTAINS '-1' (177777) THEN THE SOFTWARE SWITCH REGISTER (LOCATION 176) IS USED, WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER.

5.3 LOADING THE SOFTWARE SWITCH REGISTER

THIS PROGRAM SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (LOCATION 176) FROM THE TTY. THIS IS ACCOMPLISHED AS FOLLOWS:

1. TYPE CONTROL G <^G> REPEATEDLY, AS RESETS AND INITS DONE IN THE DIAGNOSTIC MAY CLEAR THE CHARACTER BEFORE THE CHARACTER IS RECOGNIZED. ONCE INPUT IS RECOGNIZED, THIS ALLOWS THE TTY TO ENTER DATA INTO LOCATION 176 AT THE END OF A TEST.

2. THE MACHINE WILL TYPE: SWR=XXXXXX NEW= (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER)

3. AFTER THE NEW=' THE OPERATOR CAN DO ONE OF THE FOLLOWING:

   A. TYPE A NUMBER TO BE LOADED INTO LOCATION 176 FOLLOWED BY A <CR> (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED).
   IF A <CR> IS THE FIRST ENTRY THE SOFTWARE SWITCH REGISTER WILL NOT BE CHANGED.

   B. IF A CONTROL U <^U> IS DEPRESSED, THE PROGRAM WILL GO BACK TO STEP 2.

5.4 PROGRAM AND/OR OPERATOR ACTION

LOADING AND STARTING AT 200 WITH ALL SWITCHES DOWN IS NORMAL LOGIC TESTING. IF AN ERROR IS DETECTED, THERE WILL BE A PRINTOUT. WHEN AN ERROR IS DETECTED AND IT IS NECESSARY TO SCOPE ON IT, PLACE 100000 (BIT 15) IN THE SWITCH REGISTER TO HALT ON ERROR. AFTER HALTING AT THE ERROR TO BE LOOPED ON, ENTER 60000, LOOP-ON-ERROR AND INHIBIT PRINTOUTS. IF THERE IS MORE THAN ONE ERROR CALLED IN A TEST, AND YOU WISH TO LOOP ON OTHER THAN THE 1ST ERROR, YOU MUST CORRECT THE CONDITION CAUSING THE PREVIOUS ERROR(S) BEFORE YOU CAN LOOP ON THAT ERROR. NOP'ING THE PREVIOUS ERRORS WILL PRODUCE UNPREDICTABLE RESULTS FOR ANY SUBSEQUENT ERRORS IN THE TEST.

```
                        6.0     ERROR REPORTING

                                EACH TEST WILL CALL AN ERROR CONTAINING THE TEST NUMBER, ERROR PC
                                AND DATA THAT IS SIGNIFICANT TO THE PROBLEM THAT CAUSED THE ERROR.

                                IN THE CASE OF MULTIPLE BOARD TESTING, THE FAILING MODULE IS IDENTIFIED
                                BY THE DEVICE REGISTER ADDRESS, AND THE END-OF-DEVICE-TEST MESSAGE
                                FOLLOWING ALL ERRORS FOR THAT PARTICULAR MODULE.
```

7.0 OPERATING MODE

7.1 MANUAL MODE (STARTING ADDRESS = 204)

DEFINED AS NON-AUTOMATIC USE OF THE DIAGNOSTIC.

THIS MODE IS INTENDED FOR USE IN MANUFACTURING WHEN APT IS NOT AVAILABLE.

IN MANUAL MODE, ALL DR11-W HARDWARE MODULES *MUST*BE*CONFIGURED* *AS*FOLLOWS*:

> W/B, PRIORITY LEVEL, 2/N CYCLE AND CABLE STATES SET IDENTICAL *IN*ALL*MODULES*.

> ALL DEVICE ADDRESSES MUST BE SET IN A SERIES SPACED 10 LOCATIONS APART, STARTING WITH THE ADDRESS INPUTED TO THE PROMPT 'STARTING DEVICE ADDRESS XXXXXX :'. (ALL MODULES MUST BE ADDRESSED WITHIN THE LEGAL ADDRESS RANGE OF 171000 TO 177000)

> ALL VECTOR ADDRESSES MUST BE SET IN A SERIES SPACED 10 LOCATIONS APART. (ALL MODULES MUST BE VECTORED WITHIN THE LEGAL VECTOR RANGE OF 300 TO 770)

> THE MODULE WITH THE LOWEST DEVICE ADDRESS MUST ALSO HAVE THE LOWEST VECTOR ADDRESS, THE MODULE WITH THE NEXT TO THE LOWEST DEVICE ADDRESS MUST ALSO HAVE THE NEXT TO THE LOWEST VECTOR ADDRESS, ETC. FOR EXAMPLE:

BOARD #	DEVICE ADDRESS	VECTOR ADDRESS
0	172410	300
1	172420	310
2	172430	320
3	172440	330 ETC.

ONLY UNDER MANUAL MODE DOES THE DIAGNOSTIC OFFER 'BURST DATA LATE' CALIBRATION. AFTER LOADING PROGRAM, DEPOSITING SA 204, AND PRESSING START, THE PROGRAM TYPES THE FOLLOWING:

```
  MULTIPLE BOARD DIALOGUE

  ENTER COMMAND ([E]DIT, [L]IST, [B]URST CALIBRATION, [R]UN):
```

THE PROGRAM WILL ALLOW ONLY 1 CHARACTER INPUT, AUTOMATICALLY PRINTING A <CRLF> WHEN THE CHARACTER IS INPUTED.

7.1.1 WHEN [E] IS ENTERED, THE PROGRAM ENTERS THE EDIT FUNCTION

NOTE: TO EXIT THIS ROUTINE AT ANY RESPONSE AND RETURN TO THE MBD PROMPT, ENTER CONTROL 'C' (^C). THIS DOES NOTHING BUT EXIT THE ROUTINE, AND DOES NOT CHANGE ANY VALUES PRESENT OR CHANGED. TO RETURN TO THE PREVIOUS PROMPT, TYPE <ESC>.

'EDIT' RESPONDS FIRST BY PRINTING:

  # OF BOARDS UNDER TEST X:

PROGRAM ACCEPTS A MAXIMUM OF 2 DECIMAL CHARACTERS. AN APPROPRIATE ERROR MESSAGE IS PRINTED IF THE NUMBER INPUTED IS OUT OF RANGE, OR AN ILLEGAL CHARACTER WAS INPUTED. ENTER <CR> IF PRESENT VALUE IS OK. NEXT:

  STARTING DEVICE ADDRESS XXXXXX :

THE USER SHOULD RESPOND WITH THE LOWEST DEVICE ADDRESS IN THE SERIES. PROGRAM ACCEPTS A MAXIMUM OF 6 OCTAL DIGITS BETWEEN 171000 AND 177000. AN APPROPRIATE ERROR MESSAGE IS PRINTED IF THE NUMBER INPUTED IS OUT OF RANGE, OR AN ILLEGAL CHARACTER WAS INPUTED. ENTER <CR> IF PRESENT VALUE IS OK. NEXT:

  STARTING VECTOR ADDRESS XXX :

THE USER SHOULD RESPOND WITH THE LOWEST VECTOR ADDRESS IN THE SERIES. PROGRAM ACCEPTS A MAXIMUM OF 3 OCTAL DIGITS BETWEEN 300 AND 777. AN APPROPRIATE ERROR MESSAGE IS PRINTED IF THE NUMBER INPUTED IS OUT OF RANGE, OR AN ILLEGAL CHARACTER WAS INPUTED. ENTER <CR> IF PRESENT VALUE IS OK. NEXT:

  DR11-W OR B (W=0) CURRENT STATE = X :

PROGRAM ACCEPTS EITHER A 0 OR 1, REPEATING THE PROMPT IF ANY OTHER CHARACTER IS INPUTED. ENTER <CR> IF PRESENT VALUE IS OK. NEXT:

  DEVICE PRIORITY PRESENT LEVEL = X :

PROGRAM ACCEPTS 1 CHARACTER BETWEEN 0 AND 7, REPEATING THE PROMPT IF ANOTHER CHARACTER IS INPUTED. ENTER <CR> IF PRESENT VALUE IS OK. NEXT:

  2 OR N CYCLE BURST (2 CY=0) PRESENT STATE = X .

PROGRAM ACCEPTS A 0 OR 1, REPEATING THE PROMPT IF ANY OTHER CHARACTER IS INPUTED. ENTER <CR> IF PRESENT VALUE IS OK. NEXT:

  DO CABLE TESTS (NO=0) PRESENT STATE = X :

PROGRAM ACCEPTS A 0 OR 1, REPEATING THE PROMPT IF ANY OTHER CHARACTER IS INPUTED. ENTER <CR> IF PRESENT VALUE IS OK. THEN THE COMMAND PROMPT IS REPRINTED.

7.1.2 WHEN [L] IS ENTERED, THE PROGRAM ENTERS THE LIST FUNCTION

THE DIAGNOSTIC THEN PRINTS THE FOLLOWING:

# OF BOARDS	START REGADR	VECADR	W-B	P-LEV	2-N CYCLE	CABLE TESTS
XX	XXXXXX	XXX	X	X	X	X

AS PREVIOUSLY MENTIONED, ALL BOARDS MUST BE SPACED 10 ADDRESS LOCATIONS APART STARTING WITH THE 'REGADR' VALUE ABOVE, AND VECTORS SPACED 10 ADDRESS LOCATIONS APART STARTING WITH THE 'VECADR' VALUE ABOVE. THE EXPECTED W-B, PRIORITY LEVEL, 2-N CYCLE AND CABLE TEST STATES WILL BE THE SAME FOR ALL MODULES.

7.1.3 WHEN [B] IS ENTERED, THE PROGRAM ENTERS THE BURST DATA LATE CALIBRATION ROUTINE, AND THE FOLLOWING IS TYPED:

```
  BURST DATA LATE CALIBRATION IN PROGRESS..
  ATTACH SCOPE PROBE...
  TO CALIBRATE NEXT BOARD, TYPE ANY CHARACTER
  DEVICE #  0 UNDER CALIBRATION
```

THIS ROUTINE WILL NOT EXECUTE IF YOU HAVE NOT USED EDIT TO DEPOSIT A LEGAL STARTING ADDRESS AND VECTOR ADDRESS, OR THE PROGRAM HAS ALREADY BEEN STARTED AT 200. THE MULTIPLE BOARD DIALOGUE (MBD) PROMPT WILL BE RETURNED IF THIS IS THE CASE. AS STATED IN THE DR11 ENGINEERING SPECIFICATION, THE 'BURST DLT' MULTIVIBRATOR TIME OUT MUST BE CALIBRATED SO AS TO BE COMPATIBLE WITH THE USER DEFINED TRANSFER RATE IN BURST MODE OPERATION. THE PROGRAM SOFTWARE ROUTINE SETS THE CYCLE BIT IN THE CSR OF THE DR11, A SHORT DELAY IS EXECUTED, AND THEN THE CYCLE BIT IS CLEARED. THE DIAGNOSTIC THEN TESTS FOR ANY CHARACTER WAITING, INDICATING THE USER WISHES TO GO ON TO THE NEXT BOARD. IF NONE, IT RE-EXECUTES THE SETTING AND CLEARING OF THE CYCLE BIT. IF A CHARACTER WAS INPUTED, IT CHECKS FOR THE NEXT BOARD, AND IF ANY, SETS UP THE ADDRESSES FOR THAT MODULE, THEN PRINTS THE FOLLOWING:

```
  DEVICE #  X UNDER CALIBRATION
```

'X' BEING THE DEVICE NUMBER. IT THEN REACCOMPLISHES THE SETTING AND CLEARING OF THE CYCLE BIT FOR THAT DEVICE. IF NO FURTHER MODULES ARE FOUND, THE MESSAGE:

```
  BURST CALIBRATION COMPLETE
```

IS ISSUED, AND THE MBD PROMPT IS THEN RETURNED FOR ANOTHER COMMAND. TO ACCOMPLISH THE BURST DATA LATE CALIBRATION, ATTACH A SCOPE PROBE TO E83-7 ON THE DR11-W (REFER TO PRINT SET M8716-0-1). A POSITIVE PULSE WILL BE OBSERVED. THE PULSE SHOULD BE SET BETWEEN 3-30 US. BY ADJUSTING POT. R80.

7.1.4 WHEN [R] IS ENTERED, THE PROGRAM BEGINS DIAGNOSTIC TEST EXECUTION. THIS WILL BE BLOCKED IF LEGAL STARTING DEVICE ADDRESSES AND VECTOR ADDRESSES HAVE NOT BEEN SET UP. IF THEY ARE, THE REGISTER AND VECTOR TABLES ARE FILLED, AND NORMAL START IS EXECUTED.

7.2 AUTO-SIZE MODE (STARTING ADDRESS = 200)

THIS MODE IS THE NORMAL FIELD SERVICE MODE. IT SUPPORTS STANDALONE OPERATION AS WELL AS SCRIPT OPERATION UNDER ACT11 OR XXDP (CHAIN).

THE DR11 DIAGNOSTIC HAS THE FOLLOWING RUN CHARACTERISTICS WHEN OPERATING IN AUTO MODE:

A. THE PROGRAM WILL TEST THE BOARDS RECOGNIZED BY THE AUTOSIZE ROUTINE. THE AUTOSIZE ROUTINE WILL LOOK AT ADDRESSES BETWEEN 172414 AND 172604 IN STEPS OF 10 (20 OCTAL LOCATIONS). IT WILL INITIALLY DETERMINE IF THE LOCATION IT FOUND TO EXIST IS A DR11 BY FORCING AN INTERRUPT. IF THE BOARD FAILS TO INTERRUPT, YOU MUST USE MANUAL MODE TO FORCE TEST EXECUTION OF THAT MODULE. THE PURPOSE OF THIS INITIAL TEST IS TO ELIMINATE TESTING A MODULE THAT IS NOT A DR11, AND DETERMINE THE INTERRUPT PRIORITY AND VECTOR OF THAT MODULE. THE ONLY LEGAL INTERRUPT VECTORS THE DR11 CAN BE SET UP FOR ARE AS FOLLOWS: 40, 50-174, AND 254-774, ALL IN STEPS OF 4. EACH BOARD CAN HAVE A VECTOR ANYWHERE IN THE STATED RANGES WITH NO RESTRICTIONS, ALLOWING COMPLETE FLEXIBILITY IN THE TEST SEQUENCE.

   IN THE CASE OF MULTIPLE DR11-W'S ON THE SAME CPU, EACH DR11-W MUST HAVE ITS OWN UNIQUE DEVICE/VECTOR ADDRESSES. THERE ARE NO CONSTRAINTS THAT THE BOARDS MUST START WITH THE FIRST DEVICE ADDRESS 172410, OR THAT MULTIPLE BOARDS ARE ASSIGNED CONSECUTIVE DEVICE ADDRESSES. WHEN OPERATING IN AUTO-SIZE MODE, THE USER SHOULD VERIFY THE "SIZED" CONFIGURATION BY KNOWING HOW THE BOARDS ARE SET UP AND COMPARING WITH THE AUTOSIZE OUTPUT WHEN STARTING AT 200.

   AUTO-SIZING WILL DETERMINE THE INTERRUPT PRIORITY, INTERRUPT VECTOR, W/B, 2/N CYCLE, AND CABLE STATES OF EACH BOARD, INDEPENDENT OF THE STATES OF OTHER BOARDS.

```
572                     B.  THE FOLLOWING WILL NOT BE OFFERED TO THE USER IN AUTOSIZE
573                         MODE:
574
575                         1.  BURST DATA LATE CALIBRATION
576                         2.  MULTIPLE BOARD DIALOGUE
577
578                     THE DIAGNOSTIC WILL PRINT THE FOLLOWING:
579
580                       DIAGNOSTIC HAS DETERMINED THE FOLLOWING ABOUT THE
581                       DR11-W(S) IT HAS FOUND.  USER *MUST* DETERMINE ACCURACY
582
583                          BOARD#  REGADR   VECADR   W/B    P-LEV   2-N CY  CABLE
584                            X     XXXXXX     XXX     X       X       X       X
585
586                             DATA WILL CONTINUE TO PRINT UNTIL DATA FOR ALL MODULES
587                             HAS BEEN PRINTED.
588
589                       (^X) INHIBITS EOP'S, (^Y) FOR ERROR SUMMARY
590                       UNIBUS HANG?  RESTART AT ADDRESS XXXXXX
591
592                       CZDRLC0 DR11 GEN NPR INTFC LOGIC TEST
593
594                     THE CONTROL X (^X) FEATURE BYPASSES THE SECTIONS THAT PRINT THE
595                     END-OF-PASS AND END-OF-DEVICE MESSAGES.  THIS IS TO IMPROVE THE
596                     NUMBER OF PASSES EXECUTED OVER ANY PERIOD OF TIME, AS WELL AS
597                     MAKE OVERNIGHT OR WEEKEND RUNS USE LESS PAPER.  ERROR TYPEOUTS
598                     ARE NOT DISABLED IN THIS MODE.  WHEN AN ERROR OCCURS, THE END-
599                     OF-PASS (EOP) WILL PRINT FOR THAT PASS, AND, IF MORE THAN ONE
600                     MODULE IS BEING TESTED, AN END-OF-DEVICE (EOD) AS WELL AS END-
601                     OF-PASS WILL PRINT SO YOU WILL KNOW WHICH DEVICE AND PASS WAS
602                     EXECUTING WHEN THE ERROR OCCURED.  IN ORDER TO GET A PROGRESS
603                     REPORT, HIT ANY KEY REPEATEDLY, SINCE INITS AND RESETS DONE
604                     DURING THE EXECUTION OF THE DIAGNOSTIC MAY CLEAR THE CHARACTER
605                     WAITING FLAG BEFORE THE CHECK FOR THIS BIT.  WHEN THE CHARACTER
606                     IS RECOGNIZED, AN EOP, AND IF MORE THAN ONE MODULE, AN EOD
607                     MESSAGE WILL PRINT GIVING THE USER A PROGRESS REPORT.  TO DIS-
608                     ABLE THIS FEATURE, REPEATEDLY ENTER (^X) AGAIN UNTIL THE CPU
609                     RECOGNIZES YOUR INPUT.
```

```
610                     THE CONTROL Y (^Y) FUNCTION CALLS FOR A SUMMARY OF DEVICE(S)
611                     AND PASS(ES) THAT HAD ERRORS.  IF NO ERRORS OCCURED SINCE THE
612                     BEGINNING OF THE DIAGNOSTIC, OR SINCE THE LAST ERROR REPORT,
613                     THE FOLLOWING IS PRINTED:
614
615                       NO ERROR TOTALS TO REPORT
616
617                     IF THERE WERE ERRORS, THE FOLLOWING IS PRINTED:
618
619                       SUMMATION OF ERRORS SINCE BEGINNING OR LAST REPORT
620                         BOARD #  PASS #  ERRTTL
621                           X        X        X
622                           X        X        X
623                           X        X        X (ETC.)
624
625                     THE INFORMATION IS STORED ON A STACK THAT WILL HOLD UP TO 150
626                     (DECIMAL) DEVICE-PASS ERROR DATA LINES ABOVE.  IF THE LIMIT IS
627                     REACHED, DIAGNOSTIC WILL CONTINUE, BUT FURTHER DATA WILL NOT BE
628                     STORED.  THE DATA ACCUMULATED IS NOT WRITTEN OVER, BUT WHEN (^Y)
629                     IS ENTERED, THE FOLLOWING IS PRINTED JUST BEFORE THE 'SUMMATION...'
630                     STATEMENT ABOVE:
631
632                       STACK IS FULL - DATA MAY HAVE BEEN LOST
633
634                     WHEN THE DATA IS PRINTED, THE STACK IS REINITIALIZED AND WILL
635                     START STORING UP TO ANOTHER 150 ERROR DATA LINES.
636
637                     IN THE EVENT THE UNIBUS BECOMES HUNG, AND YOU HAVE NON-VOLATILE
638                     MEMORY OR BATTERY BACKUP, RESTART THE PROGRAM AT THE ADDRESS
639                     SPECIFIED BY THE 'UNIBUS HUNG...' PROMPT AT THE START OF THE
640                     DIAGNOSTIC.  THE PRINTOUT WILL BE AS FOLLOWS:
641
642                       DEVICE ADDRESS - XXXXXX, TEST NUMBER - XXXXXX, PASS NUMBER - XXXXXX
643
644                     CPU WILL HALT.  HITTING CONTINUE WILL CAUSE THE PROGRAM TO
645                     RESTART AS THOUGH YOU HAD STARTED AT 200.
```

646
647
648
649
650
651
652
653
654
655
656
657
658

7.3 RESTARTING PROGRAM IN MEMORY (STARTING ADDRESS = 210)

WHENEVER THE PROGRAM IS HALTED, ALL HISTORY OF PREVIOUS TESTING IS SAVED. IT WILL REMAIN INTACT UNTIL:

1. ANOTHER PROGRAM IS LOADED INTO MEMORY
2. THE USER RE-EDITS THE TABLE

TO RESTART THE PROGRAM, ENTER SA 210 AND START. THIS START PRECLUDES ANY SETUP AND NEGATES THE START MESSAGE OBTAINED WHEN STARTING AT 200. DO NOT START AT THIS LOCATION IF THE DIAGNOSTIC HAS NOT BEEN PREVIOUSLY "STARTED" AT EITHER 200 OR 204.

```
                    7.4     TESTING UNDER APT (AUTOMATED PRODUCT TESTING)

                            TO SET UP FOR MULTIPLE BOARDS FOR TESTING UNDER APT CONTROL,
                            THE APT SYSTEM MANAGER SHOULD ANSWER THE APT QUERIES TO THE
                            FOLLOWING ITEMS AS INDICATED BELOW:

                            SOFTWARE ENVIRONMENT:       000 - DUMP MODE
                                                        001 - SCRIPT MODE (APT MONITORS
                                                              DIAGNOSTIC)

                            ENVIRONMENT MODE ($ENVM):   000 - LET DIAGNOSTIC AUTO-SIZE
                                                              CONFIGURATOR AND TEST
                                                              ACCORDINGLY

                                                        200 - DIAGNOSTIC MUST USE CONFIGURA-
                                                              TION SPECIFIED BY APT ($VECT1,
                                                              $VASE, $DEVM, $DDWX)

                            VECTOR ADDRESS ($VFCT1):    300

                            DEVICE ADDRESS ($BASE):     172410

                            DEVICE MAP ($DEVM):         XXXXXX - EACH SET BIT INDICATES THAT
                                                                 BOARD IS PRESENT AND SHOULD
                                                                 BE TESTED. EXAMPLES:

                                                          BIT 0 = BOARD #0 (DEVICE ADR = 172410, VEC ADR = 300)
                                                          BIT 1 = BOARD #1 (DEVICE ADR = 172420, VEC ADR = 310)
                                                          BIT 2 = BOARD #2 (DEVICE ADR = 172430, VEC ADR = 320)
                                                                  .
                                                                  .
                                                                  .
                                                          BIT 15 = BOARD #15 (DEVICE ADR = 172600, VEC ADR = 470)

                        DEVICE DESCRIPTOR WORDS:  XXXXXX - THERE IS 1 DESCRIPTOR WORD
                                                           FOR EACH DEVICE:

                                                           $DDW0 IS FOR DEVICE 0
                                                           $DDW1 IS FOR DEVICE 1, ETC.

                                                           EACH DESCRIPTOR WORD MUST BE
                                                           SET UP AS FOLLOWS:

                                                            BIT 0 - DR11-W OR -B MODE
                                                                    (W=0, B=1)
                                                            BIT 1 - 2/N CYCLE
                                                                    (0=2 CY, 1=N CY)
                                                            BIT 2 - CABLE TESTS
                                                                    (0=NO, 1=YES)
                                                            BIT 5 \
                                                            BIT 6  > DEVICE PRIORITY
                                                            BIT 7 /
```

8.0 MISCELLANEOUS

8.1 POWER FAIL

IF A POWER FAILURE OCCURS AND BATTERY BACKUP MAINTAINS THE PROGRAM IN MEMORY, OR A NON-VOLATILE MEMORY EXISTS, THE PROGRAM WILL RESTART PRINTING THE FOLLOWING:

```
POWER FAILURE - RESTARTING PROGRAM
```

THE DIAGNOSIC WILL THEN RESTART AT ADDRESS 210.

IF CPU IS TURNED OFF WHILE RUNNING, THE ABOVE PROCEDURE IS FOLLOWED. IF THE PROCESSOR IS HALTED FIRST, THEN TURNED OFF, THE PROCESSOR WILL COME BACK UP HALTED. TO RESTART THE PROGRAM, HIT CONTINUE, AND THE REMAINING PROCEDURE IS THE SAME AS ABOVE.

8.2 END-OF-PASS MESSAGE

THE EOP WILL PRINT AS FOLLOWS WITH NO ERRORS ON THAT PASS:

```
END PASS # XXXXXX
```

THE EOP WILL PRINT AS FOLLOWS WITH SOME ERRORS WHEN TESTING 1 DEVICE:

```
END PASS # XXXXXX   TOTAL ERRORS SINCE LAST REPORT XXXXXX
```

THE EOP WILL PRINT THE SAME AS WITH NO ERRORS ON ANY PARTICULAR PASS WHEN TESTING MORE THAN ONE DEVICE AND ONE OR MORE DEVICES HAS FAILED, SINCE 'TOTAL ERRORS' IS MEANINGLESS AND WILL MORE THAN LIKELY BE INCORRECT.

THE PASS NUMBER IS CAPABLE OF GOING UP TO 99,999,999 DECIMAL, OR ABOUT 3 MONTHS RUNNING WITH EOP DISABLED AND NO ERRORS. IN OTHER WORDS, 32767 IS NOT THE LIMIT AS WITH OTHER DIAGNOSTICS.

```
747                                 9.0     EXECUTION TIME
748
749                                         ON A PDP11/44:
750
751                                         IN ALL MODES: APPROXIMATELY 8 PASSES PER SECOND WITH EOP MESSAGES
752                                         DISABLED AND NO ERRORS.
```

10.0 SUBROUTINE ABSTRACT

10.1 READ

THE READ SUBROUTINE IS USED IN THE EDIT ROUTINE TO INPUT UP TO 6 DIGITS IN OCTAL, 2 DIGITS IN DECIMAL, OR A SINGLE NON-NUMERIC CHARACTER. R4 IS USED AS THE LOCATION TO HOLD THE NUMBER IN OCTAL, AND IS CLEARED FOR THAT PURPOSE AT THE START OF THE SUBROUTINE. R3 IS TO BE PRELOADED WITH THE NUMBER OF DIGITS EXPECTED, SINCE A <CRLF> IS PRINTED WHEN THE LIMIT IS REACHED. ENTERING A <CRLF> BEFORE THE LIMIT IS ACCEPTABLE, AS IT WILL BE INTERPRETED AS A NON-NUMERIC CHARACTER AND EXIT. IN ANY CASE, THE LAST INPUTED ASCII CHARACTER IS LEFT IN LOCATION 'ANSWER'. IF A NUMERIC CHARACTER IS INPUTED, IT WILL CLEAR ALL BUT THE 1ST 4 BITS IN LOCATION 'ANSWER', EXPOSING THE VALUE OF THE DIGIT INPUTED, ROTATE R4 TO THE LEFT 3 PLACES TO MAKE ROOM FOR THE INPUTED DIGIT, AND ADD IT TO R4. LOCATION 'LRGSTC' IS TO BE LOADED WITH THE LARGEST ASCII NUMBER DIGIT ACCEPTABLE FOR THIS NUMBER, I.E. 7 OR 9 (FOR OCTAL OR DECIMAL INPUT RESPECTIVELY). ANY CHARACTER OUTSIDE ASCII '0' OR '7/9' IS TREATED AS A NON-NUMERIC, TRIGGERING AN AUTOMATIC <CRLF> AND EXIT.

10.2 ERCAPT

THIS SUBROUTINE SAVES THE UNIT NUMBER, PASS NUMBER AND TOTAL ERRORS FOR THAT DEVICE/PASS WHENEVER IT ENCOUNTERED ERRORS. THIS ROUTINE SAVES DATA FOR 150 (DECIMAL) PASSES. IF THE STACK SHOULD BECOME FULL, DATA STARTING WITH THE 151ST PASS CONTAINING ERRORS IS LOST.

10.3 PTCAPT

THIS SUBROUTINE PRINTS THE DATA STORED BY THE ERCAPT SUBROUTINE AND RESETS THE SPECIAL STACK POINTER. IF NO DATA WAS STORED, A MESSAGE STATING NO DATA WAS STORED IS PRINTED. IF THE STACK IS FOUND FULL, A MESSAGE ANNOUNCING THIS FINDING IS PRINTED, WARNING THAT DATA MAY HAVE BEEN LOST.

10.4 FIXTBL

THIS SUBROUTINE FILLS THE 17 OCTAL LOCATIONS STARTING AT 'REGADR' AND 'VECADR' FROM THE STARTING VALUES ALREADY LOADED IN THE FIRST LOCATIONS IN STEPS OF 10 FOR EACH TABLE.

10.5 LODBUF

THE INBUF BUFFER IS LOADED WITH AN INCREMENTING PATTERN (0,1,2,3,...) BEGINNING AT THE STARTING ADDRESS OF INBUF. THE NUMBER OF WORDS LOADED IS DETERMINED BY THE CONTENTS OF BUFLEN.

10.6 CHKBFF

THE CHKBUF BUFFER IS LOADED WITH A MODIFIED INCREMENTING PATTERN (0,0,2,2,4,4,6,6,...) BEGINNING AT THE STARTING ADDRESS OF CHKBUF. THE NUMBER OF WORDS LOADED IS DETERMINED BY THE CONTENTS OF BUFLEN. THIS BUFFER IS LOADED ONLY FOR TESTS WHICH USE THE MAINTENANCE MODE OF THE DR11-W WHICH HAS A SPECIAL ALTERNATING DATI-DATO SEQUENCE OF OPERATION.

10.7 INTA

THE IE BIT IS CLEARED IN THE CSR THEN THE CSR IS CHECKED FOR THE ABSENCE OF THE ERROR BIT AND THE PRESENCE OF READY. THE WCR IS CHECKED TO SEE THAT IT IS EQUAL TO ZERO. THE CORRECT CONTENTS OF THE BAR ARE CALCULATED AND CHECKED. THE PROGRAM WILL FAIL TO UPDATE THE PC RETURN ADDRESS BY 2 IF ERROR IS SET, READY IS CLEAR, READY AND ERROR ARE CLEAR OF THE CSR, WCR IS NOT ZERO OR THE BAR CONTENTS IS NOT ZERO. THIS WILL CALL THE ERROR THAT IS JUST AFTER THE JSR CALL IN THE TEST. IF ALL DATA IS ACCEPTABLE, THE PC IS UPDATED, AND THE RETURN FROM THE SUBROUTINE IS AFTER THE ERROR CALL.

10.8 DATCHK

THIS ROUTINE IS ENTERED TO CHECK INBUF AFTER A MAINTENANCE MODE OPERATION. THE CONTENTS OF INBUF AND THE CONTENTS OF CHKBUF ARE CHECKED TO SEE THAT THEY ARE THE SAME. THE NUMBER OF COMPARISONS MADE IS DETERMINED BY THE CONTENTS OF BUFLEN. ANY ERRORS RESULT IN AN RTS TO THE TEST TO CALL THE ERROR THERE. A JSR BACK TO THE SUBROUTINE IS EXECUTED TO RESUME ITS CHECKING. WHEN RETURNING, SP RETURN ADDRESS IS UPDATED BY 6 TO RETURN AFTER THE ERROR CALL AND JSR RETURN.

10.9 CLENUP

THE ROUTINE IS ENTERED AT THE END OF SEVERAL TESTS TO CLEAR ANY DATA THAT MAY HAVE BEEN LEFT IN ANY REGISTERS, AND TO RESTORE THE INTERRUPT VECTORS.

10.10 CHKCAB

THIS ROUTINE IS USED IN VARIOUS TESTS TO ALTER THE EXPECTED DATA IF THE WRAP-AROUND CABLE IS OUT.

10.11 DATOCK

AFTER A STRING OF DATO'S HAS BEEN COMPLETED THIS ROUTINE CHECKS THAT THE CORRECT DATA PATTERN WAS TRANSFERRED TO INBUF. THE NUMBER OF COMPARISONS MADE IS DETERMINED BY THE CONTENTS OF BUFLEN. AN ERROR IN THE CHECK RESULTS IN AN RTS TO THE TEST TO CALL THE FIRST ERROR AFTER THE JSR CALL, WHERE A JSR RETURNS CONTROL BACK TO THE SUBROUTINE FOR FURTHER CHECKING. AN ADDITIONAL CHECK IS MADE ON BUFLEN+2 TO INSURE THAT NOT TOO MANY WORDS WERE TRANSFERRED. IF THEY WERE, THE PC RETURN ADDRESS IS ALTERED SO THAT THE SECOND ERROR AFTER THE JSR IS CALLED. IF NO ERRORS, RETURN IS ALTERED TO JUST AFTER THE SECOND ERROR CALL.

10.12 ERRCHK

THIS ROUTINE CLEARS IE AND UPDATES THE PC FOR RETURN AFTER THE ERROR IN THE TEST IF ERROR IS CLEAR. IF SET, RETURN IS EXECUTED WITHOUT UPDATING THE PC RETURN SO THE ERROR CALL AFTER THE JSR CALL IN THE TEST WILL BE CALLED.

10.13 DEVADS

THIS ROUTINE GENERATES AN ADDRESS TABLE LOCATED AT REGADR STARTING WITH THE BASE DEVICE ADDRESS (CONTENTS OF $BASE) IN STEPS OF 10.

10.14 BPINIT

THIS SUBROUTINE RELOADS THE ".+2" AND "BPT" INTO THE UNUSED LOCATIONS BETWEEN 4 AND 776.

10.15 DRGET

THIS SUBROUTINE EXTRACTS INFORMATION ABOUT THE DR11 THAT INTERRUPTED AND LOADS THE ACCUMULATED DATA INTO THE DEVICE DESCRIPTOR WORD FOR THAT BOARD.

10.16 TYPCNF

THIS SUBROUTINE PRINTS THE BOARD CONFIGURATIONS THAT THE ASIZE SUBROUTINE FOUND ON THE UNIBUS.

10.17 CHK4DR

THIS SUBROUTINE CHECKS FOR A LOCATION AS BELONGING TO A DR11 BY TRYING TO FORCE AN INTERRUPT A TOTAL OF 4 TIMES WITH THE CPU AT PRIORITIES 6 THROUGH 3. IF THE ATTEMPT FAILS, ROUTINE CORRECTS THE STACK TO RETURN AFTER THE DR11 EXTRACTION ROUTINE. IF THE LOCATION DOES BELONG TO A DR11, THE RETURN ADDRESS OF THE SUBROUTINE ON THE STACK IS MOVED DOWN ONE LOCATION, AND THE ADDRESS+4 OF THE INTERRUPT VECTOR OF THE DR11 IS PUT IN THE RETURN ADDRESSES PREVIOUS LOCATION. THE STACK IS THEN POPPED 3 TIMES BY ADJUSTING THE POINTER UP 6, AND A NORMAL RETURN IS EXECUTED.

10.18 ASIZE

THIS ROUTINE AUTOSIZES THE BOARD CONFIGURATION AND PRINTS THE CONFIGURATION IF BIT 12 (10000) IS SET IN THE SWR.

10.19 VCTADS

THIS ROUTINE GENERATES THE VECTOR ADDRESS TABLE STARTING WITH THE ADDRESS IN LOCATION 'VECADR'.

10.20 CATCH

THIS ROUTINE REPORTS UNEXPECTED OR ERRONEOUS TRAPS OR INTERRUPTS THROUGH THE BREAK-POINT-TRAP LOADED IN LOCATIONS 4-776. THE STACK IS CLEANED 4 TIMES BEFORE THE ERROR CALL, AND RESTORED TWICE AFTER THE ERROR CALL FOR RETURNING TO THE SOURCE OF THE TRAP.

10.21 PSTATE

THIS ROUTINE PRINTS THE STATE OF THE BIT IN THE DDW THAT WAS PRELOADED IN LOCATION 'BITTST'.

10.22 PNTPRI

THIS ROUTINE PRINTS THE DEVICE PRIORITY IN THE DDW LOCATION.

10.23 SETTUP

THIS SUBROUTINE INITIALIZES THE TRAP AND INTERRUPT VECTORS.

10.24 TSTMM

THIS SUBROUTINE CHECKS FOR EXISTENCE OF MEMORY MANAGEMENT AND IF IT EXISTS, CHECKS FOR THE ERROR CONDITION OF NO MEMORY LOCATION, BUT NO ERROR AND NEX BIT SETS. IF MEMORY MANAGEMENT IS NOT THERE, AN EXIT UPDATING THE RETURN ADDRESS BY 2 IS DONE. IF THERE, THE XBA16 AND XBA17 BITS OF THE EXPECTED DATA ARE CHECKED. IF BOTH ZERO, AN EXIT UPDATING THE RETURN ADDRESS BY 2 IS DONE. IF EITHER OR BOTH ARE SET, THE UPPER BYTE OF THE MEMORY MANAGEMENT LOCATION IS CHECKED FOR THE EXISTENCE OF UPPER MEMORY, INITIALIZED AT THE BEGINNING OF THE DIAGNOSTIC. IF NOT THERE (BITS 0, 1 OR 2 OF UPPER BYTE CLEAR), A NORMAL EXIT IS EXECUTED SO THE BRANCH IMMEDIATELY FOLLOWING THE JSR CALL WILL CAUSE A CHECK FOR THE ERROR BITS IN THE EXPECTED TO BE SET FOR ANOTHER CHECK.

```
                        11.0    DATA STACK

                                11.1    PATRNS

                                        THIS SET OF 7 DATA WORDS IS USED TO CHECK ANY LOCATION FOR
                                        STUCK OR SHORTED BITS.

                                11.2    EXPAT0

                                        THIS SET OF DATA WORDS IS USED IN TEST 31 TO CHECK ALL POS-
                                        SIBLE COMBINATIONS OF SET BITS IN THE CSR WITH THE MAINTENANCE
                                        BIT CLEAR.  IT CONTAINS THE EXPECTED DATA THAT THE CSR SHOULD
                                        CONTAIN AFTER THE BIT COMBINATION IS WRITTEN TO THE CSR.

                                11.3    EXPAT1

                                        THIS SET OF DATA WORDS IS USED IN TEST 3 TO CHECK ALL POSSIBLE
                                        COMBINATIONS OF SET BITS IN THE CSR WITH THE MAINTENANCE BIT
                                        SET.  IT CONTAINS THE EXPECTED DATA THAT THE CSR SHOULD CONTAIN
                                        AFTER THE BIT COMBINATION IS WRITTEN TO THE CSR.
```

```
 967                    @
 968                    .NLIST  MC,MD,CND
1006                    .TITLE  CZDRLC0-DR11 GEN NPR INTFC
                        ;*COPYRIGHT (C) 1980
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY DAN MILLEVILLE
                        ;*
                        ;*      THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*      PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977, MODIFIED FOR
                        ;*      THE CZDRLC DIAGNOSTIC.  CHANGES ARE NOTED BY [;&& ] IN THE
                        ;*      COMMENT FIELDS OF THE $SCOPE, $EOP, $TYPE, $ERROR, $CKSWR
                        ;*      $PWRDN, $PWRUP (ELIMINATED), $ERRTYP AND $TYPDS ROUTINES.
                        ;*      NEWTST WAS MODIFIED IN FUNCTION AS DESCRIBED BELOW.  CHANGES
                        ;*      WERE AS FOLLOWS:
                        ;*
                        ;*      $SCOPE:   END-OF-PASS MESSAGE DISABLING AND REENABLING CAPABILITY,
                        ;*                PLUS UPDATING $TESTN JUST AFTER $TSTNM IS UPDATED, AS
                        ;*                WELL AS ANY NEEDED MESSAGES.
                        ;*
                        ;*      $EOP:     CHANGED TO RECOGNIZE WHETHER OR NOT THE USER WISHES
                        ;*                THE EOP MESSAGES PRINTED, AND PRINTS THE NUMBER OF
                        ;*                ERRORS IN THAT PASS IF THERE WERE ANY, AND TYPES AN
                        ;*                EXTRA <CRLF> IF THERE WERE ERRORS TO SPACE EOP FROM
                        ;*                THE ERROR.  CHANGED ALSO TO RECOGNIZE WHEN THE PASS
                        ;*                COUNT GOES NEGATIVE, AND IF SO, CLEARS THE PASS COUNT,
                        ;*                AND INCREMENTS THE NEXT LOCATION AFTER $PASS TO COUNT
                        ;*                A BLOCK OF 32768 PASSES HAS OCCURED.  THIS LOCATION
                        ;*                IS USED IN CONJUNCTION WITH $PASS TO PRINT UP TO
                        ;*                PASS # 99,999,999 DECIMAL.
                        ;*
                        ;*      $TYPE:    INSTEAD OF USING THE STACK TO LOAD A CHARACTER INPUTED
                        ;*                WHILE PRINTING, LOOSING THE CHARACTER IN THE PROCESS,
                        ;*                LOCATION 'CHARCT' IS USED TO SAVE IT FOR $SCOPE TO USE.
                        ;*
                        ;*      $ERROR:   INCREMENT LOCATION 'ERRCNT' FOR POSSIBLE USE IN MULTIPLE
                        ;*                ERROR PRINTOUTS.
                        ;*
                        ;*      $CKSWR:   TESTING AND PROCESSING OF THE CONTENTS OF LOCATION
                        ;*                'CHARCT' WAS ADDED TO INCREASE CHANCES OF DIAGNOSTIC
                        ;*                CATCHING USER REQUEST FOR A SOFTWARE SWITCH REGISTER
                        ;*                CHANGE.
                        ;*
                        ;*      $ERRTYP:  ADDED CAPABILITY TO PROCESS ERRORS WITHIN LOOPS (ERRORS
                        ;*                WITH ERROR NUMBERS BETWEEN 201-377) SO THE MESSAGE AND
                        ;*                DATA HEADER ARE PRINTED DURING THE 1ST ERROR ONLY, WITH
                        ;*                DATA ONLY PRINTED FOR 2ND AND SUBSEQUENT ERRORS.  IT
                        ;*                CANCELS DATA PRINTING AFTER 20 (OCTAL) ERRORS HAVE BEEN
                        ;*                DONE SO AS TO ELIMINATE MASSIVE ERROR TYPEOUTS, BUT
                        ;*                CONTINUES TO TALLY THE ERRORS IN LOCATION '$ERTTL' SO THE
                        ;*                EOP MESSAGE WILL SHOW THE TOTAL NUMBER OF ERRORS IN THAT
                        ;*                PASS.  IF AN ERROR NUMBER BELOW 201 IS CALLED, 'ERRCNT'
                        ;*                IS CLEARED SO IF THIS ERROR IS IN A LOOP, ANY SUBSEQUENT
                        ;*                201+ ERRORS WILL HAVE THERE HEADER REPRINTED.
```

```
;*      NEWTST:   '.PAGE' WAS ADDED SO EACH NEW TEST WOULD BE ON A NEW PAGE.
;*                AT THE BEGINING OF EACH NEW TEST, THE TITLE AND TEST NUMBER
;*                ARE WRITTEN IN A SUBTITLE SO THAT EACH TEST WILL APPEAR IN
;*                THE TABLE OF CONTENTS AT THE BEGINING OF THE DIAGNOSTIC.
;*                AN ADDITIONAL TRACE FEATURE WAS ADDED AT THE BEGINNING
;*                OF EACH TEST TO CHECK BIT 11 OF THE SWITCH REGISTER.  WITH
;*                THIS BIT SET, THE USER CAN SEE WHAT TEST WAS EXECUTING
;*                WHEN THE OPERATION CEASED TO BE SEQUENTIAL.
;*
;*      $PWRDN:   THE OPERATIONS TO SAVE REGISTER CONTENTS WERE ELIMINATED
;*                DUE TO THE LACK OF THE NEED.  A LOAD OF THE $PWRUP ADDRESS,
;*                LOCATED JUST AFTER THE POWER DOWN HALT, IS LOADED INTO THE
;*                POWER TRAP VECTOR SO THAT ON POWER UP, THE PROGRAM WILL
;*                RESTART.  IF PROCESSOR IS UNABLE TO GET TO THE POWER DOWN
;*                ROUTINE ON A POWER FAIL (CPU HALTED WHEN POWER FAILURE
;*                OCCURED), PROCESSOR WILL EXECUTE THE POWER DOWN ROUTINE AND
;*                HALT.  HITTING CONTINUE WILL RESULT IN THE POWER UP ROUTINE
;*                EXECUTING AS IT WOULD IN THE EVENT OF A RESTORATION OF POWER
;*                AFTER A POWER FAILURE WITH THE CPU RUNNING.
;*
;*      $TYPDS:   THE ADDITION OF THE $TYPDE FUNCTION WAS ADDED.  THIS FUNCTION
;*                ALLOWS THE PRINTING OF NUMBERS LARGER THAN 32767 DECIMAL.
;*                THE LOCATION THAT CONTAINS THE COUNT IS TO BE TESTED AFTER
;*                BEING INCREMENTED.  IF NEGATIVE, IT IS TO BE CLEARED AND A
;*                SECOND (OVERFLOW) LOCATION IS TO BE INCREMENTED.  WHEN
;*                CALLING THE ROUTINE, THE OVERFLOW LOCATION IS TO BE PUT ON
;*                THE STACK, THEN THE NUMBER, THEN THE CALL.  IF THE OVERFLOW
;*                LOCATION IS NON-ZERO, IT WILL ADD 32768 TO THE ASCII NUMBER
;*                FOR EACH COUNT IN THAT OVERFLOW LOCATION.
```

```
1008                    .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH                  USE
                        ;*      ------          --------------------
                        ;*        15            HALT ON ERROR
                        ;*        14            LOOP ON TEST
                        ;*        13            INHIBIT ERROR TYPEOUTS
                        ;*        12            DO NOT PRINT BOARD CONFIGURATION
                        ;*        11            TEST NUMBER TRACE ENABLING
                        ;*        10            BELL ON ERROR
                        ;*         9            LOOP ON ERROR
                        ;*         8            LOOP ON TEST IN SWR<6:0>
1009                    ;*         7            INHIBIT MULTIPLE ERROR TYPEOUTS
1010
1011                    ;*************************************************************************
1012                    ;*                      >>>>>>>>>>>>>>NOTE<<<<<<<<<<<<<<
1013                    ;*      THE "TEST NUMBER TRACE ENABLING" ABOVE IS TO ENABLE PRINTING OF THE
1014                    ;*      TEST NUMBER AT THE BEGINNING OF EACH TEST, SHOULD YOU HAVE A PROBLEM
1015                    ;*      WITH THE DIAGNOSTIC JUMPING TO AN ILLEGAL LOCATION FROM ANYWHERE IN
1016                    ;*      THE TEST.  BY ENABLING IT, YOU CAN SEE THE LAST TEST ENTERED BEFORE
1017                    ;*      THE OUT-OF-SEQUENCE JUMPING OCCURED.  THIS IS USEFUL IF YOU ENCOUNTER
1018                    ;*      AN UNEXPECTED TRAP FROM A PARTICULAR TEST, BUT DISCOVER THAT THE TEST
1019                    ;*      RAN SUCCESSFULLY WHEN LOOPED.
1020                    ;*
1021                    ;*************************************************************************
```

```
1023                            .SBTTL  BASIC DEFINITIONS
                                ;*INITIAL ADDRESS OF THE STACK POINTER *** 1300 ***
001300                          STACK=  1300
104000                                  ERROR=EMT
000004                                  SCOPE=IOT
                                ;*MISCELLANEOUS DEFINITIONS
000011                          HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                          LF=     12              ;;CODE FOR LINE FEED
000015                          CR=     15              ;;CODE FOR CARRIAGE RETURN
000200                          CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776                          PS=     177776          ;;PROCESSOR STATUS WORD
177776                                  PSW=PS
177774                          STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                          PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                          DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                          DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
                                ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                          R0=     %0              ;;GENERAL REGISTER
000001                          R1=     %1              ;;GENERAL REGISTER
000002                          R2=     %2              ;;GENERAL REGISTER
000003                          R3=     %3              ;;GENERAL REGISTER
000004                          R4=     %4              ;;GENERAL REGISTER
000005                          R5=     %5              ;;GENERAL REGISTER
000006                          R6=     %6              ;;GENERAL REGISTER
000007                          R7=     %7              ;;GENERAL REGISTER
000006                          SP=     %6              ;;STACK POINTER
000007                          PC=     %7              ;;PROGRAM COUNTER
                                ;*PRIORITY LEVEL DEFINITIONS
000000                          PR0=    0               ;;PRIORITY LEVEL 0
000040                          PR1=    40              ;;PRIORITY LEVEL 1
000100                          PR2=    100             ;;PRIORITY LEVEL 2
000140                          PR3=    140             ;;PRIORITY LEVEL 3
000200                          PR4=    200             ;;PRIORITY LEVEL 4
000240                          PR5=    240             ;;PRIORITY LEVEL 5
000300                          PR6=    300             ;;PRIORITY LEVEL 6
000340                          PR7=    340             ;;PRIORITY LEVEL 7
                                ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
100000                          SW15=   100000
040000                          SW14=   40000
020000                          SW13=   20000
010000                          SW12=   10000
004000                          SW11=   4000
002000                          SW10=   2000
001000                          SW09=   1000
000400                          SW08=   400
000200                          SW07=   200
000100                          SW06=   100
000040                          SW05=   40
000020                          SW04=   20
000010                          SW03=   10
000004                          SW02=   4
000002                          SW01=   2
000001                          SW00=   1
001000                                  SW9=SW09
000400                                  SW8=SW08
000200                                  SW7=SW07
000100                                  SW6=SW06
```

```
                000040                  SW5=SW05
                000020                  SW4=SW04
                000010                  SW3=SW03
                000004                  SW2=SW02
                000002                  SW1=SW01
                000001                  SW0=SW00
                                ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
                100000          BIT15=  100000
                040000          BIT14=  40000
                020000          BIT13=  20000
                010000          BIT12=  10000
                004000          BIT11=  4000
                002000          BIT10=  2000
                001000          BIT09=  1000
                000400          BIT08=  400
                000200          BIT07=  200
                000100          BIT06=  100
                000040          BIT05=  40
                000020          BIT04=  20
                000010          BIT03=  10
                000004          BIT02=  4
                000002          BIT01=  2
                000001          BIT00=  1
                001000                  BIT9=BIT09
                000400                  BIT8=BIT08
                000200                  BIT7=BIT07
                000100                  BIT6=BIT06
                000040                  BIT5=BIT05
                000020                  BIT4=BIT04
                000010                  BIT3=BIT03
                000004                  BIT2=BIT02
                000002                  BIT1=BIT01
                000001                  BIT0=BIT00
                                ;*BASIC "CPU" TRAP VECTOR ADDRESSES
                000004          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
                000010          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
                000014          TBITVEC=14              ;;"T" BIT
                000014          TRTVEC= 14              ;;TRACE TRAP
                000014          BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
                000020          IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
                000024          PWRVEC= 24              ;;POWER FAIL
                000030          EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
                000034          TRAPVEC=34              ;;"TRAP" TRAP
                000060          TKVEC=  60              ;;TTY KEYBOARD VECTOR
                000064          TPVEC=  64              ;;TTY PRINTER VECTOR
                000240          PIRQVEC-240             ;;PROGRAM INTERRUPT REQUEST VECTOR
1024            000004          BUSERR  =ERRVEC
1025            172410          ABASE   =172410         ;BASE DEVICE ADDRESS
1026            000300          AVECT1  -300            ;BASE VECTOR ADDRESS
```

```
                                .SBTTL  DEFINITIONS OF THE CSR BITS
                        ;*****************************************************************
000001                  GO      =1      ;GO
000002                  F1      =2      ;FNCT1
000004                  F2      =4      ;FNCT2
000010                  F3      =10     ;FNCT3
000016                  FNC     =16     ;FNCT1 & FNCT2 & FNCT3
000020                  X6      =20     ;XBA16
000040                  X7      =40     ;XBA17
000100                  IE      =100    ;IE
000200                  RY      =200    ;READY
000400                  CY      =400    ;CYCLE
000400                  N2      =400    ;2/N BIT
001000                  DSC     =1000   ;DSTAT C
002000                  DSB     =2000   ;DSTAT B
004000                  DSA     =4000   ;DSTAT A
006000                  DAB     =6000   ;DSTAT A & B
005000                  DAC     =5000   ;DSTAT A & C
003000                  DBC     =3000   ;DSTAT B & C
007000                  DST     =7000   ;DSTAT A & B & C
010000                  MA      =10000  ;MAINT
020000                  AT      =20000  ;ATTN
040000                  NX      =40000  ;NEX
100000                  EIR     =100000 ;EIR
100000                  FR      =100000 ;ERROR
```

```
                                .SBTTL  CSR BIT COMPLIMENT DEFINITIONS
                        ;***************************************************************************
                        ;
177776          CGO     =177776 ;COMPLIMENT OF GO
177775          CF1     =177775 ;COMPLIMENT OF FNCT1
177773          CF2     =177773 ;COMPLIMENT OF FNCT2
177767          CF3     =177767 ;COMPLIMENT OF FNCT3
177761          CFNC    =177761 ;COMPLIMENT OF FNCT1 & FNCT2 & FNCT3
177757          CX6     =177757 ;COMPLIMENT OF XBA16
177737          CX7     =177737 ;COMPLIMENT OF XBA17
177677          CIE     =177677 ;COMPLIMENT OF IE
177577          CRY     =177577 ;COMPLIMENT OF READY
177377          CCY     =177377 ;COMPLIMENT OF CYCLE
176777          CDSC    =176777 ;COMPLIMENT OF DSTAT C
175777          CDSB    =175777 ;COMPLIMENT OF DSTAT B
173777          CDSA    =173777 ;COMPLIMENT OF DSTAT A
171777          CDAB    =171777 ;COMPLIMENT OF DSTAT A & B
172777          CDAC    =172777 ;COMPLIMENT OF DSTAT A & C
174777          CDBC    =174777 ;COMPLIMENT OF DSTAT B & C
170777          CDST    =170777 ;COMPLIMENT OF DSTAT A & B & C
167777          CMA     =167777 ;COMPLIMENT OF MAINT
157777          CAT     =157777 ;COMPLIMENT OF ATTN
137777          CNX     =137777 ;COMPLIMENT OF NEX
077777          CEIR    =77777  ;COMPLIMENT OF EIR
077777          CER     =77777  ;COMPLIMENT OF ERROR
```

```
1076                                    .SBTTL  COMPLEMENTS OF BIT DEFINITIONS
1077                                    ;*****************************************************************
1078            177776                  CBIT0   =177776 ;COMPLIMENT OF BIT0
1079            177775                  CBIT1   =177775 ;COMPLIMENT OF BIT1
1080            177773                  CBIT2   =177773 ;COMPLIMENT OF BIT2
1081            177767                  CBIT3   =177767 ;COMPLIMENT OF BIT3
1082            177757                  CBIT4   =177757 ;COMPLIMENT OF BIT4
1083            177737                  CBIT5   =177737 ;COMPLIMENT OF BIT5
1084            177677                  CBIT6   =177677 ;COMPLIMENT OF BIT6
1085            177577                  CBIT7   =177577 ;COMPLIMENT OF BIT7
1086            177377                  CBIT8   =177377 ;COMPLIMENT OF BIT8
1087            176777                  CBIT9   =176777 ;COMPLIMENT OF BIT9
1088            175777                  CBIT10  =175777 ;COMPLIMENT OF BIT10
1089            173777                  CBIT11  =173777 ;COMPLIMENT OF BIT11
1090            167777                  CBIT12  =167777 ;COMPLIMENT OF BIT12
1091            157777                  CBIT13  =157777 ;COMPLIMENT OF BIT13
1092            137777                  CBIT14  =137777 ;COMPLIMENT OF BIT14
1093            077777                  CBIT15  =77777  ;COMPLIMENT OF BIT15
1094            057777                  CB1513  =57777  ;COMPLIMENT OF BIT15 & BIT 13
```

```
                                .SBTTL  PRIORITY LEVELS AND OTHER DEFINITIONS
                               ;*****************************************************************
000140                         LEVEL3  =140
000200                         LEVEL4  =200
000240                         LEVEL5  =240
000300                         LEVEL6  =300
000340                         LEVEL7  =340
000033                         ESC     =33
000003                         CNTLC   =3
000015                         CARETN  =15
177572                         MMR0    =177572
172304                         KIPDR2  =172304
172324                         KDPDR2  =172324
172344                         KIPAR2  =172344
172364                         KDPAR2  =172364
000250                         MMVECT  =250
000252                         MMPS    =252
000004                         TOVECT  =4
000006                         TMOPSW  =6
000003                         BPT     =3
024360                         TST40=ENDEV     ;BRANCH TO TEST 40 = BRANCH TO ENDEV (THERE IS NO TEST 40)
```

```
                                ;*****************************************************************
                                ;*  ALL UNUSED LOCATIONS FROM 4-776 WILL CONTAIN A ".+2,BPT" SEQUENCE
                                ;*  TO CATCH ILLEGAL TRAPS & INTERRUPTS TO THE 'CATCH' LOCATION
                                ;*  0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                                ;*****************************************************************
000014                          .=14                    ;THE BPT TRAP VECTOR POINTS TO THE
000014 005536                   BPTVCT: .WORD   CATCH    ;   ILLEGAL TRAP HANDLER "CATCH"
000016 000340                           .WORD   LEVEL7
000042                          .=42
000042 000000                           .WORD   0       ;CLEAR THIS LOCATION (FOR APT MONITOR STARTING ADDRESS)
000174                          .=174
000174 000000                   DISPRE: .WORD   0
000176 000000                   SWREG:  .WORD   0
                                ;*****************************************************************
                                ;
                                ;PROGRAM STARTING LOCATIONS
                                ;
                                ;*****************************************************************
000200  000137  010266                  JMP     START1          ;NORMAL START
000204  000137  032130                  JMP     MBD             ;ENTER MULTIPLE BOARD DIALOQUE
000210  005037  001416           STAGIN: CLR     $PASS           ;CLEAR $PASS
000214  005037  002716                   CLR     $PASS2          ;CLEAR $PASS2
000220  005037  001422                   CLR     $UNIT           ;CLEAR $UNIT
000224  005037  001420                   CLR     $DEVCT          ;CLEAR $DEVCT
000230  005037  001414                   CLR     $TESTN          ;CLEAR $TESTN
000234  005037  002706                   CLR     EOPLOC          ;CLEAR EOPLOC
000240  012737  042456  042454          MOV     #CAPSTK,CAPNTR  ;RESET THE CAPTURE POINTER
000246  012706  010272                   MOV     #START,SP       ;RESET THE STACK POINTER
000252  000137  011006                   JMP     BEGIN1          ;JUMP TO BEGIN1 FOR RESTART WITHOUT HEADER PRINTING
001000                          .=1000
```

```
1147                                    .SBTTL  ACT11 HOOKS
                                        ;*****************************************************************
                                        ;HOOKS REQUIRED BY ACT11
                001000                          $SVPC=.                 ;SAVE PC
                000046                          .=46
        000046  024750                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                000052                          .=52
        000052  000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
                001000                          .=$SVPC                 ;; RESTORE PC
1148                                    .SBTTL  APT PARAMETER BLOCK
                                        ;*****************************************************************
                                        ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                        ;*****************************************************************
                001000                          .$X=.   ;;SAVE CURRENT LOCATION
                000024                          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
        000024  000200                          200     ;;FOR APT START UP
                000044                          .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
        000044  001000                          $APTHDR ;;POINT TO APT HEADER BLOCK
                001000                          .=.$X   ;;RESET LOCATION COUNTER
                                        ;*****************************************************************
                                        ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                        ;INTERFACE SPEC.
                001000                  $APTHD:
                001000  000000          $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
                001002  001410          $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
                001004  000001          $TSTM:  .WORD   1       ;;RUN TIM OF LONGEST TEST
                001006  000001          $PASTM: .WORD   1       ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
                001010  000000          $UNITM: .WORD   0       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
                001012  000052                  .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
1150                                    .SBTTL  COMMON TAGS
                                        ;**********************************************************************
                                        ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                        ;*USED IN THE PROGRAM.
                 001300                         .=1300
001300                          $CMTAG:                         ;;START OF COMMON TAGS
001300  000000                          .WORD   0
001302     000                  $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001303     000                  $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001304  000000                  $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001306  000000                  $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001310  000000                  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001312  000000                  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001314     000                  $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001315     001                  $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001316  000000                  $EPRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001320  000000                  $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001322  000000                  $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001324  000000                  $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001326  000000                  $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001330  000000                          .WORD   0               ;;RESERVED--NOT TO BE USED
001332  000000                          .WORD   0
001334     000                  $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001335     000                  $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001336  000000                          .WORD   0
001340  177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001342  177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001344  177560                  $TKS:   177560                  ;;TTY KBD STATUS
001346  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
001350  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001352  177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001354     000                  $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001355     002                  $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001356     012                  $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001357     000                  $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07> 0-YES)
                 000007                 .REPT   7
001360  000000                  $TMP0:  .WORD   0               ;;USER DEFINED
001362  000000                  $TMP1:  .WORD   0               ;;USER DEFINED
001364  000000                  $TMP2:  .WORD   0               ;;USER DEFINED
001366  000000                  $TMP3:  .WORD   0               ;;USER DEFINED
001370  000000                  $TMP4:  .WORD   0               ;;USER DEFINED
001372  000000                  $TMP5:  .WORD   0               ;;USER DEFINED
001374  000000                  $TMP6:  .WORD   0               ;;USER DEFINED
001376  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001400     207      377     377 $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001404     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
001405     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001406     012      000         $LF:    .ASCIZ  <12>            ;;LINE FEED
                                        ;**********************************************************************
```

```
                                .SBTTL  APT MAILBOX-ETABLE
                                ;*****************************************************************
                                .EVEN
001410                          $MAIL:                          ;;APT MAILBOX
001410  000000                  $MSGTY: .WORD   AMSGTY  ;;MESSAGE TYPE CODE
001412  000000                  $FATAL: .WORD   AFATAL  ;;FATAL ERROR NUMBER
001414  000000                  $TESTN: .WORD   ATESTN  ;;TEST NUMBER
001416  000000                  $PASS:  .WORD   APASS   ;;PASS COUNT
001420  000000                  $DEVCT: .WORD   ADEVCT  ;;DEVICE COUNT
001422  000000                  $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
001424  000000                  $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
001426  000000                  $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
001430                          $ETABLE:                        ;;APT ENVIRONMENT TABLE
001430     000                  $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
001431     000                  $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
001432  000000                  $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
001434  000000                  $USWR:  .WORD   AUSWR   ;;USER SWITCHES
001436  000000                  $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
                                ;*                              BITS 15-11=CPU TYPE
                                ;*                                      11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                                ;*                                      11/70=06,PDQ=07,Q=10
                                ;*                              BIT 10=REAL TIME CLOCK
                                ;*                              BIT  9=FLOATING POINT PROCESSOR
                                ;*                              BIT  8=MEMORY MANAGEMENT
001440     000                  $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
001441     000                  $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
                                ;*                              MEM.TYPE BYTE   --  (HIGH BYTE)
                                ;*                                      900 NSEC CORE=001
                                ;*                                      300 NSEC BIPOLAR=002
                                ;*                                      500 NSEC MOS=003
001442  000000                  $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
                                ;*                                MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF ''TYPE'' ABOVE
001444     000                  $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
001445     000                  $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
001446  000000                  $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
001450     000                  $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
001451     000                  $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
001452  000000                  $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
001454     000                  $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
001455     000                  $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
001456  000000                  $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
001460  000300                  $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001462  000000                  $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001464  172410                  $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001466  000000                  $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
001470  000000                  $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
001472  000000                  $CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
001474  000000                  $DDW0:  .WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
001476  000000                  $DDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
001500  000000                  $DDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
001502  000000                  $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
001504  000000                  $DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
001506  000000                  $DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
001510  000000                  $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
001512  000000                  $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
001514  000000                  $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
001516  000000                  $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
```

```
001520  000000          $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
001522  000000          $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
001524  000000          $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
001526  000000          $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
001530  000000          $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
001532  000000          $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
001534                  $ETEND:
                        .MEXIT
```

```
                                .SBTTL  ERROR POINTER TABLE
                                ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
                                ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                ;*      DH              ;;POINTS TO THE DATA HEADER
                                ;*      DT              ;;POINTS TO THE DATA
                                ;*      DF              ;;POINTS TO THE DATA FORMAT
    001534                      $ERRTB:
1151
1152 001534  044766             ;ITEM 1
1153 001536  050307                     .WORD   EM2     ;CANNOT ACCESS DR11 REGISTER
1154 001540  053052                     .WORD   DH2     ;TEST #  ERR PC  ABRTPC  REGISTER
1155 001542  000000                     .WORD   DT2     ;$TESTN,$ERRPC,OLDPC1,DREG,0
1156                                    .WORD   0       ;PRINT ALL DATA OCTAL
1157
1158 001544  045022             ;ITEM 2
1159 001546  050350                     .WORD   EM3     ;DR11-B OR W MODE INCORRECT (0=B, 1=W)
1160 001550  053064                     .WORD   DH3     ;TEST #  ERR PC  EXPMOD  ACTMOD  CSRADR
1161 001552  000000                     .WORD   DT3     ;$TESTN,$ERRPC,$TMP1,BORW,CSR,0
1162                                    .WORD   0       ;PRINT ALL DATA OCTAL
1163
1164 001554  045070             ;ITEM 3
1165 001556  050417                     .WORD   EM4     ;INIT FAILED TO CLEAR WCR
1166 001560  053100                     .WORD   DH4     ;TEST #  ERR PC  WCRADR  WCRCONTENTS
1167 001562  000000                     .WORD   DT4     ;$TESTN,$ERRPC,WCR,RWCR,0
1168                                    .WORD   0       ;PRINT ALL DATA OCTAL
1169
1170 001564  045121             ;ITEM 4
1171 001566  050463                     .WORD   EM5     ;INIT FAILED TO CLEAR BAR
1172 001570  053112                     .WORD   DH5     ;TEST #  ERR PC  BARADR  BAREXP  BARRCV
1173 001572  000000                     .WORD   DT5     ;$TESTN,$ERRPC,BAR,EBAR,RBAR,0
1174                                    .WORD   0       ;PRINT ALL DATA OCTAL
1175
1176 001574  045152             ;ITEM 5
1177 001576  050532                     .WORD   EM6     ;INIT FAILED TO CLEAR BDR
1178 001600  053126                     .WORD   DH6     ;TEST #  ERR PC  BDRADR  BDRCONTENTS
1179 001602  000000                     .WORD   DT6     ;$TESTN,$ERRPC,BDR,RBDR,0
1180                                    .WORD   0       ;PRINT ALL DATA OCTAL
1181
1182 001604  045203             ;ITEM 6
1183 001606  050576                     .WORD   EM7     ;INIT FAILED TO CLEAR ALL CSR R-W BITS
1184 001610  053140                     .WORD   DH7     ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
1185 001612  000000                     .WORD   DT7     ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
1186                                    .WORD   0       ;PRINT ALL DATA OCTAL
1187
1188 001614  045251             ;ITEM 7
1189 001616  050417                     .WORD   EM10    ;RESET FAILED TO CLEAR WCR
1190 001620  053100                     .WORD   DH4     ;TEST #  ERR PC  WCRADR  WCRCONTENTS
1191 001622  000000                     .WORD   DT4     ;$TESTN,$ERRPC,WCR,RWCR,0
                                        .WORD   0       ;PRINT ALL DATA OCTAL
```

```
1192                                    ;ITEM 10
1193 001624  045303                             .WORD   EM11    ;ATTEMPT TO SET ALL WCR BITS FAILED
1194 001626  050417                             .WORD   DH4     ;TEST #  ERR PC  WCRADR  WCRCONTENTS
1195 001630  053100                             .WORD   DT4     ;$TESTN,$ERRPC,WCR,RWCR,0
1196 001632  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1197
1198                                    ;ITEM 11
1199 001634  045346                             .WORD   EM12    ;RESET FAILED TO CLEAR BAR
1200 001636  050463                             .WORD   DH5     ;TEST #  ERR PC  BARADR  BAREXP  BARRCV
1201 001640  053112                             .WORD   DT5     ;$TESTN,$ERRPC,BAR,EBAR,RBAR,0
1202 001642  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1203
1204                                    ;ITEM 12
1205 001644  045400                             .WORD   EM13    ;ATTEMPT TO SET ALL BAR BITS TO 1 FAILED
1206 001646  050463                             .WORD   DH5     ;TEST #  ERR PC  BARADR  BAREXP  BARRCV
1207 001650  053112                             .WORD   DT5     ;$TESTN,$ERRPC,BAR,EBAR,RBAR,0
1208 001652  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1209
1210                                    ;ITEM 13
1211 001654  045450                             .WORD   EM14    ;CSR BIT TEST FAILED (FATAL - DIAGNOSTIC NOT CONTINUED)
1212 001656  050652                             .WORD   DH14    ;               BIT(S)
1213                                                            ;TEST #  ERR PC  TESTED  CSRADR  CSREXP  CSRCONTENTS
1214 001660  053154                             .WORD   DT14    ;$TESTN,$ERRPC,BUT,CSR,ECSR,RCSR,0
1215 001662  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1216
1217                                    ;ITEM 14
1218 001664  045537                             .WORD   EM15    ;CSR BIT TEST FAILED
1219 001666  050652                             .WORD   DH14    ;               BIT(S)
1220                                                            ;TEST #  ERR PC  TESTED  CSRADR  CSREXP  CSRCONTENTS
1221 001670  053154                             .WORD   DT14    ;$TESTN,$ERRPC,BUT,CSR,ECSR,RCSR,0
1222 001672  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1223
1224                                    ;ITEM 15
1225 001674  045563                             .WORD   EM16    ;EIR BIT TEST FAILED
1226 001676  050765                             .WORD   DH16    ;                   BIT(S)
1227                                                            ;TEST #  ERR PC  TESTED  EIRADR  EIREXP  EIRCONTENTS
1228 001700  053172                             .WORD   DT16    ;$TESTN,$ERRPC,BUT,CSR,EEIR,REIR,0
1229 001702  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1230
1231                                    ;ITEM 16
1232 001704  045607                             .WORD   EM17    ;READY AND MAINTENANCE ARE NOT THE ONLY BITS SET IN CSR
1233 001706  051100                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
1234 001710  053210                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
1235 001712  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1236
1237                                    ;ITEM 17
1238 001714  045676                             .WORD   EM20    ;ATTN AND ERROR FAILED TO SET PROPERLY
1239 001716  051100                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
1240 001720  053210                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
1241 001722  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1242
1243                                    ;ITEM 20
1244 001724  045744                             .WORD   EM21    ;ATTN AND ERROR FAILED TO CLEAR PROPERLY
1245 001726  051100                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
1246 001730  053210                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
1247 001732  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
```

```
1248                                    ;ITEM 21
1249 001734  046014                            .WORD   EM22    ;ERROR BIT SHOULD HAVE BEEN CLEAR
1250 001736  051100                            .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
1251 001740  053210                            .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
1252 001742  000000                            .WORD   0       ;PRINT ALL DATA OCTAL
1253
1254                                    ;ITEM 22
1255 001744  046124                            .WORD   EM24    ;READY OF CSR WAS NOT SET
1256 001746  051100                            .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
1257 001750  053210                            .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
1258 001752  000000                            .WORD   0       ;PRINT ALL DATA OCTAL
1259
1260                                    ;ITEM 23
1261 001754  046155                            .WORD   EM25    ;BIT 0 OF THE BAR WAS SET
1262 001756  050463                            .WORD   DH5     ;TEST #  ERR PC  BARADR  BAREXP  BARRCV
1263 001760  053112                            .WORD   DT5     ;$TESTN,$ERRPC,BAR,EBAR,RBAR,0
1264 001762  000000                    .       .WORD   0       ;PRINT ALL DATA OCTAL
1265
1266                                    ;ITEM 24
1267 001764  046302                            .WORD   EM30    ;FUNCTION BIT(S) ARE NOT CLEAR
1268 001766  051100                            .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
1269 001770  053210                            .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
1270 001772  000000                            .WORD   0       ;PRINT ALL DATA OCTAL
1271
1272                                    ;ITEM 25
1273 001774  046340                            .WORD   EM31    ;DSTAT A, B OR C ARE NOT AS EXPECTED
1274 001776  051100                            .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
1275 002000  053210                            .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
1276 002002  000000                            .WORD   0       ;PRINT ALL DATA OCTAL
1277
1278                                    ;ITEM 26
1279 002004  046404                            .WORD   EM32    ;BDR IS NOT CLEAR
1280 002006  050532                            .WORD   DH6     ;TEST #  ERR PC  BDRADR  BDRCONTENTS
1281 002010  053126                            .WORD   DT6     ;$TESTN,$ERRPC,BDR,RBDR,0
1282 002012  000000                            .WORD   0       ;PRINT ALL DATA OCTAL
1283
1284                                    ;ITEM 27
1285 002014  046425                            .WORD   EM33    ;ALL BDR BITS ARE NOT SET
1286 002016  050532                            .WORD   DH6     ;TEST #  ERR PC  BDRADR  BDRCONTENTS
1287 002020  053126                            .WORD   DT6     ;$TESTN,$ERRPC,BDR,RBDR,0
1288 002022  000000                            .WORD   0       ;PRINT ALL DATA OCTAL
1289
1290                                    ;ITEM 30
1291 002024  046516                            .WORD   EM35    ;BDR SHOULD NOT HAVE BEEN LOADED WITH NEW PATTERN
1292 002026  051304                            .WORD   DH34    ;TEST #  ERR PC  BDRADR  BDREXP  BDRCONTENTS
1293 002030  053254                            .WORD   DT34    ;$TESTN,$ERRPC,BDR,EBDR,RBDR,0
1294 002032  000000                            .WORD   0       ;PRINT ALL DATA OCTAL
1295
1296                                    ;ITEM 31
1297 002034  046577                            .WORD   EM36    ;BDR PATTERN NOT CORRECT
1298 002036  051304                            .WORD   DH34    ;TEST #  ERR PC  BDRADR  BDREXP  BDRCONTENTS
1299 002040  053254                            .WORD   DT34    ;$TESTN,$ERRPC,BDR,EBDR,RBDR,0
1300 002042  000000                            .WORD   0       ;PRINT ALL DATA OCTAL
```

```
1301                                    ;ITEM 32
1302 002044  046627                             .WORD   EM37    ;READY IS NOT THE ONLY BIT SET
1303 002046  051100                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
1304 002050  053210                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
1305 002052  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1306
1307                                    ;ITEM 33
1308 002054  046665                             .WORD   EM40    ;READY SHOULD NOT BE SET
1309 002056  051100                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
1310 002060  053210                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
1311 002062  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1312
1313                                    ;ITEM 34
1314 002064  046715                             .WORD   EM41    ;READY WAS CLEARED BUT NEVER SET AGAIN
1315 002066  051100                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
1316 002070  053210                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
1317 002072  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1318
1319                                    ;ITEM 35
1320 002074  047017                             .WORD   EM43    ;DR11 FAILED TO INTERRUPT
1321 002076  051360                             .WORD   DH43    ;TEST #  ERR PC  CSRADR  CSRCONTENTS
1322 002100  053270                             .WORD   DT43    ;$TESTN,$ERRPC,CSR,RCSR,0
1323 002102  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1324
1325                                    ;ITEM 36
1326 002104  047050                             .WORD   EM44    ;DR11 INTERRUPTED, BUT IT SHOULDN'T HAVE
1327 002106  051360                             .WORD   DH43    ;TEST #  ERR PC  CSRADR  CSRCONTENTS
1328 002110  053270                             .WORD   DT43    ;$TESTN,$ERRPC,CSR,RCSR,0
1329 002112  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1330
1331                                    ;ITEM 37
1332 002114  047120                             .WORD   EM45    ;ERROR BIT SHOULD NOT BE CLEAR
1333 002116  051100                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
1334 002120  053210                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
1335 002122  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1336
1337                                    ;ITEM 40
1338 002124  047233                             .WORD   EM47    ;CSR IS WRONG
1339 002126  051100                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
1340 002130  053210                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
1341 002132  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1342
1343                                    ;ITEM 41
1344 002134  047250                             .WORD   EM50    ;TRANSFERS SHOULD HAVE BEEN INHIBITED
1345 002136  051424                             .WORD   DH50    ;TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  BARADR  BAREXP  BARRCV
1346 002140  053302                             .WORD   DT50    ;$TESTN,$ERRPC,WCR,EWCR,RWCR,BAR,EBAR,RBAR,0
1347 002142  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
1348
1349                                    ;ITEM 42
1350 002144  047315                             .WORD   EM51    ;DR11 SHOULD NOT HAVE INTERRUPTED A SECOND TIME
1351 002146  051360                             .WORD   DH43    ;TEST #  ERR PC  CSRADR  CSRCONTENTS
1352 002150  053270                             .WORD   DT43    ;$TESTN,$ERRPC,CSR,RCSR,0
1353 002152  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
```

```
1354
1355 002154  047374                    ;ITEM 43
1356 002156  051360                            .WORD   EM52    ;EXPECTED INTERRUPT DID NOT OCCUR
1357 002160  053270                            .WORD   DH43    ;TEST #  ERR PC  CSRADR  CSRCONTENTS
1358 002162  000000                            .WORD   DT43    ;$TESTN,$ERRPC,CSR,RCSR,0
1359                                           .WORD   0       ;PRINT ALL DATA OCTAL
1360
1361 002164  047463                    ;ITEM 44
1362 002166  051230                            .WORD   EM54    ;BAR IS WRONG
1363 002170  053240                            .WORD   DH26    ;TEST #  ERR PC  BARADR  BAREXP  BARCONTENTS
1364 002172  000000                            .WORD   DT26    ;$TESTN,$ERRPC,BAR,EBAR,RBAR,
1365                                           .WORD   0       ;PRINT ALL DATA OCTAL
1366
1367 002174  047500                    ;ITEM 45
1368 002176  051304                            .WORD   EM55    ;BAD DATA IN BDR
1369 002200  053254                            .WORD   DH34    ;TEST #  ERR PC  BDRADR  BDREXP  BDRCONTENTS
1370 002202  000000                            .WORD   DT34    ;$TESTN,$ERRPC,BDR,EBDR,RBDR,
1371                                           .WORD   0       ;PRINT ALL DATA OCTAL
1372
1373 002204  047556                    ;ITEM 46
1374 002206  051602                            .WORD   EM57    ;BUFFER DATA NOT CORRECT
1375                                           .WORD   DH57    ;                CHECK   CHECK   INPUT   INPUT
1376 002210  053342                                            ;TEST #  ERR PC  BUFADR  BUFDAT  BUFADR  BUFDAT  CSRADR
1377 002212  000000                            .WORD   DT57    ;$TESTN,$ERROC,$TMP4,$TMP2,$TMP5,$TMP3,CSR,0
1378                                           .WORD   0       ;PRINT ALL DATA OCTAL
1379
1380 002214  047606                    ;ITEM 47
1381 002216  051747                            .WORD   EM60    ;TOO MANY WORDS WERE TRANSFERED
1382                                           .WORD   DH60    ;                DIDNOT
1383 002220  053362                                            ;TEST #  ERR PC  EXPECT  ADRESS  CSRADR
1384 002222  000000                            .WORD   DT60    ;$TESTN,$ERRPC,$TMP2,$TMP3,CSR,0
1385                                           .WORD   0       ;PRINT ALL DATA OCTAL
1386
1387 002224  047645                    ;ITEM 50
1388 002226  052045                            .WORD   EM61    ;UNEXPECTED TRAP OR INTERRUPT TO TRAP ADDRESS BELOW
1389 002230  053376                            .WORD   DH61    ;TEST #  ERR PC  WCRADR  OLDPC   TRAP ADR
1390 002232  000000                            .WORD   DT61    ;$TESTN,$ERRPC,WCR,OLDPC2,BDVECT,0
1391                                           .WORD   0       ;PRINT ALL DATA OCTAL
1392
1393 002234  047730                    ;ITEM 51
1394 002236  052116                            .WORD   EM62    ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
1395 002240  053412                            .WORD   DH62    ;TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  CSREXP  CSRRCV  BAREXP  BAR
1396 002242  000000                            .WORD   DT62    ;$TESTN,$ERRPC,WCR,EWCR,RWCR,ECSR,RCSR,EBAR,RBAR,0
1397                                           .WORD   0       ;PRINT ALL DATA OCTAL
1398
1399 002244  047777                    ;ITEM 52
1400 002246  052225                            .WORD   EM63    ;DR11 INTERRUPTED AT WRONG LEVEL
1401 002250  053436                            .WORD   DH63    ;TEST #  ERR PC  EXPLVL  RCVLVL  CSRADR
1402 002252  000000                            .WORD   DT63    ;$TESTN,$ERRPC,DRLEV,LEVEL,CSR,0
1403                                           .WORD   0       ;PRINT ALL DATA OCTAL
1404
1405 002254  047017                    ;ITEM 53
1406 002256  052274                            .WORD   EM43    ;DR11 FAILED TO INTERRUPT
1407 002260  053452                            .WORD   DH64    ;TEST #  ERR PC  EXPLVL  CSRADR
1408 002262  000000                            .WORD   DT64    ;$TESTN,$ERRPC,$TMP1,CSR,0
                                               .WORD   0       ;PRINT ALL DATA OCTAL
```

```
1409                                    ;ITEM 54
1410 002264  050037                             .WORD   EM65    ;2-N CYCLE BURST SWITCH IN WRONG POSITION
1411 002266  052333                             .WORD   DH65    ;TEST #  ERR PC  CSRADR  EIREXP  EIRRCV
1412 002270  053464                             .WORD   DT65    ;$TESTN,$ERRPC,CSR,EEIR,REIR,0
1413 002272  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
```

```
1414 002274                    ER200:                  ;THIS IS THE STARTING POINT FOR ERROR MESSAGES 201
1415                                                   ;THROUGH 277.  THEY ARE USED FOR MULTIPLE ERROR MESSAGES.
1416                           ;ITEM 201
1417 002274  047556                     .WORD   EM57    ;BUFFER DATA NOT CORRECT
1418 002276  051602                     .WORD   DH57    ;                       CHECK   CHECK   INPUT   INPUT
1419                                                   ;TEST #  ERR PC  BUFADR  BUFDAT  BUFADR  BUFDAT  CSRADR
1420 002300  053342                     .WORD   DT57    ;$TESTN,$ERRPC,$TMP4,$TMP2,$TMP5,$TMP3,CSR,0
1421 002302  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
1422
1423                           ;ITEM 202
1424 002304  050153                     .WORD   EM202   ;CSR PATTERN NOT CORRECT
1425 002306  052451                     .WORD   DH202   ;TEST #  ERR PC  CSRADR  PATLDD  CSREXP  CSRRCV
1426 002310  053514                     .WORD   DT202   ;$TESTN,$ERRPC,CSR,BUT,ECSR,RCSR,0
1427 002312  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
1428
1429                           ;ITEM 203
1430 002314  046206                     .WORD   EM26    ;BIT PATTERN TEST FAILED IN BAR
1431 002316  051230                     .WORD   DH26    ;TEST #  ERR PC  BARADR  BAREXP  BARCONTENTS
1432 002320  053240                     .WORD   DT26    ;$TESTN,$ERRPC,BAR,EBAR,RBAR,
1433 002322  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
1434
1435                           ;ITEM 204
1436 002324  046245                     .WORD   EM27    ;WCR DATA PATTERN NOT CORRECT
1437 002326  051154                     .WORD   DH23    ;TEST #  ERR PC  WCRADR  WCREXP  WCRCONTENTS
1438 002330  053224                     .WORD   DT23    ;$TESTN,$ERRPC,WCR,EWCR,RWCR,0
1439 002332  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
1440
1441                           ;ITEM 205
1442 002334  046577                     .WORD   EM36    ;BDR PATTERN NOT CORRECT
1443 002336  051304                     .WORD   DH34    ;TEST #  ERR PC  BDRADR  BDREXP  BDRCONTENTS
1444 002340  053254                     .WORD   DT34    ;$TESTN,$ERRPC,BDR,EBDR,RBDR,0
1445 002342  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
1446
1447                           ;ITEM 206
1448 002344  047435                     .WORD   EM53    ;WCR NOT EQUAL TO ZERO
1449 002346  052670                     .WORD   DH210   ;TEST #  ERR PC  WCRADR  WCRCONTENTS
1450 002350  053564                     .WORD   DT210   ;$TESTN,$ERRPC,WCR,RWCR,0
1451 002352  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
1452
1453                           ;ITEM 207
1454 002354  047463                     .WORD   EM54    ;BAR IS WRONG
1455 002356  051230                     .WORD   DH26    ;TEST #  ERR PC  BARADR  BAREXP  BARCONTENTS
1456 002360  053240                     .WORD   DT26    ;$TESTN,$ERRPC,BAR,EBAR,RBAR,
1457 002362  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
1458
1459                           ;ITEM 210
1460 002364  047520                     .WORD   EM56    ;DATA NOT TRANSFERED CORRECTLY
1461 002366  051523                     .WORD   DH56    ;TEST #  ERR PC  NPR1AD  NPR1EX  NPR1RC  CSRADR
1462 002370  053324                     .WORD   DT56    ;$TESTN,$ERRPC,ANPR1,ENPR1,NPR1,CSR,0
1463 002372  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
1464
1465                           ;ITEM 211
1466 002374  050203                     .WORD   EM211   ;BDR AND-OR WCR AND-OR BAR ARE INCORECT
1467 002376  052734                     .WORD   DH211   ;TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  BDREXP  BDRRCV  BAREXP  BAR
1468 002400  053576                     .WORD   DT211   ;$TESTN,$ERRPC,WCR,EWCR,RWCR,ECSR,RCSR,EBAR,RBAR,0
1469 002402  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
```

```
1470                                    ;ITEM 212
1471 002404  047156                            .WORD   EM46    ;FUNCTION BITS DIDN'T INCREMENT IN MAINT MODE
1472 002406  051100                            .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
1473 002410  053210                            .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
1474 002412  000000                            .WORD   0       ;PRINT ALL DATA OCTAL
```

```
                                        .SBTTL  STORAGE LOCATIONS
                                ;*******************************************************************
                                ;
                                ;       STORAGE LOCATIONS
                                ;
                                ;*******************************************************************

002414  000001                  QTYBRD: .WORD   1       ;TOTAL # DR11 BOARDS BEING TESTED (DEFAULT = 1)

002416                          REGADR: .BLKW   16.     ;TOTAL: 16 LOCATIONS FOR BOARD ADDRESSES
002456                          VECADR: .BLKW   16.     ;TOTAL: 16 LOCATIONS FOR VECTOR ADDRESSES

                                ;REGISTER AND VECTOR ADDRESS STORAGE LOCATIONS FOR THE DR11 UNDER TEST

                                ;*******************************************************************
                                ;DO NOT INSERT ANY ITEMS BETWEEN ANY OF THE LOCATIONS BELOW
                                ;*******************************************************************
002516  000000                  WCR:    .WORD   0
002520  000000                  BAR:    .WORD   0
002522  000000                  CSR:    .WORD   0
002524  000000                  BDR:    .WORD   0
002526  000000                  DRINV:  .WORD   0
002530  000000                  DRVS:   .WORD   0
002532  000000                  SDRINV: .WORD   0
002534  000000                  SDRVS:  .WORD   0
                                ;*******************************************************************
                                ;DO NOT INSERT ANY ITEMS BETWEEN ANY OF THE LOCATIONS ABOVE
                                ;*******************************************************************
002536  000000                  BUT:    .WORD   0       ;BIT(S) UNDER TEST LOCATION
002540  000000                  LEVEL:  .WORD   0       ;BR LEVEL LOCATION
002542  000000                  BDVECT: .WORD   0
002544  000000                  DEVMSK: .WORD   0
002546  000000                  TABINX: .WORD   0
002550  000000                  DREG:   .WORD   0
002552  000000                  DRLEV:  .WORD   0
002554  000000                  NXTTST: .WORD   0
002556  000000                  PASCNT: .WORD   0

002560  000000                  RCSR:   .WORD   0       ;CSR ACTUALLY READ FROM DEVICE UNDER TEST
002562  000000                  REIR:   .WORD   0       ;EIR ACTUALLY READ FROM DEVICE UNDER TEST
002564  000000                  RBDR:   .WORD   0       ;BDR ACTUALLY READ FROM DEVICE UNDER TEST
002566  000000                  RBAR:   .WORD   0       ;BAR ACTUALLY READ FROM DEVICE UNDER TEST
002570  000000                  RWCR:   .WORD   0       ;WCR ACTUALLY READ FROM DEVICE UNDER TEST

002572  000000                  ECSR:   .WORD   0       ;CSR EXPECTED
002574  000000                  EEIR:   .WORD   0       ;EIR EXPECTED
002576  000000                  EBDR:   .WORD   0       ;BDR EXPECTED
002600  000000                  EBAR:   .WORD   0       ;BAR EXPECTED
002602  000000                  EWCR:   .WORD   0       ;WCR EXPECTED
002604  000000                  ENPR1:  .WORD   0       ;EXPECTED OF NPR1
002606  002612                  ANPR1:  .WORD   NPR1    ;ADDRESS OF NPR1
002610  000000                  BORW:   .WORD   0
002612  052525                  NPR1:   .WORD   52525
002614  173000                  DIOMEM: .WORD   173000
002616  040450                  INBUF:  .WORD   XINBUF
002620  041452                  CHKBUF: .WORD   XCHKBU
002622  000000                  BUFLEN: .WORD   0
```

```
1532 002624  000000              LENCHK: .WORD   0
1533 002626  000000              BRWAIT: .WORD   0
1534 002630  000000              WCLEN:  .WORD   0
1535 002632  000000              RDYCHK: .WORD   0
1536 002634  177560              TKS:    .WORD   177560
1537 002636  177562              TKB:    .WORD   177562
1538 002640  177564              TPS:    .WORD   177564
1539 002642  177566              TPB:    .WORD   177566
1540 002644  000000              MSG:    .WORD   0
1541 002646  000000              ADDR:   .WORD   0
1542 002650  000000              MESSAG: .WORD   0
1543 002652  000000              FLAG:   .WORD   0
1544 002654  000000              FNCCNT: .WORD   0
1545 002656  000000              INBUF1: .WORD   0
1546 002660  000000              TIME:   .WORD   0       ;GENERAL PURPOSE TIMER
1547 002662  000000              LOOP:   .WORD   0       ;GENERAL PURPOSE LOOP COUNTER
1548 002664  000000              ANSWER: .WORD   0
1549 002666  000000              BDFAIL: .WORD   0
1550 002670  000000              MANSIZ: .WORD   0
1551 002672  000000              OLDPC1: .WORD   0       ;LOCATION TO STORE RETURN PC IN SUBROUTINES WITH ERROR CALLS
1552 002674  000000              OLDPS1: .WORD   0       ;LOCATION TO STORE PS
1553 002676  000000              OLDPC2: .WORD   0       ;LOCATION TO STORE RETURN PC IN SUBROUTINES WITH ERROR CALLS
1554 002700  000000              OLDPS2: .WORD   0       ;LOCATION TO STORE PS
1555 002702  000000              OFL:    .WORD   0       ;FIRST CHAR FLAG
1556 002704  000000              LRGSTC: .WORD   0       ;LOCATION FOR LARGEST NUMBER CHARACTER FOR THE READ SUBROUTINE
1557 002706  000000              EOPLOC: .WORD   0       ;LOCATION TO HOLD FLAG DECIDING IF EOP MSGS ARE TO BE PRINTED
1558 002710  000000              BITTST: .WORD   0       ;LOCATION TO PUT THE BIT STATE TO PRINT - USED BY SUBROUTINE PSTATE
1559 002712  000000              MEMGMT: .WORD   0       ;LOCATION TO HOLD FLAG SAYING MEMORY MANAGEMENT IS AVAILABLE
1560 002714  000000              ERRCNT: .WORD   0       ;LOCATION TO HOLD TOTAL ERRORS FOR A PARTICULAR TEST
1561 002716  000000              $PASS2: .WORD   0       ;LOCATION TO HOLD OVERFLOW PASS NUMBER
```

```
                                        .SBTTL  DEVICE DESCRIPTOR WORD BIT DESCRIPTION
                                ;***************************************************************
                                ;       DESCRIPTION OF BITS IN THE DDW (DEVICE DESCRIPTOR WORD):
                                ;
                                ;       BIT 0   DR11-W=0, DR11-B=1
                                ;       BIT 1   2 CYCLE=0, N CYCLE=1
                                ;       BIT 2   CABLE DOESN'T EXIST=0, CABLE DOES EXIST=1
                                ;       BIT 5   \
                                ;       BIT 6    > BR PRIORITY
                                ;       BIT 7   /
                                ;
002720  000000                  DDW:    .WORD   0       ;LOCATION FOR STORAGE OF THE DEVICE DESCRIPTOR WORD
```

```
                                        .SBTTL  SUBROUTINE TO INPUT A CHARACTER OR UP TO A 6 DIGIT NUMBER
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE IS USED IN THE EDIT ROUTINE TO INPUT NUMBERS AND A
                                ;*      SINGLE CHARACTER.  R3 IS TO BE LOADED WITH THE NUMBER OF DIGITS
                                ;*      EXPECTED.  THIS SUBROUTINE WILL EXIT IF A NON-NUMERIC CHARACTER IS
                                ;*      INPUTED, LEAVING THE CHARACTER IN LOCATION 'ANSWER'.  IF A NUMERIC
                                ;*      CHARACTER IS INPUTED, IT WILL CLEAR ALL BUT THE 1ST 4 BITS EXPOSING
                                ;*      THE VALUE OF THE DIGIT INPUTED, AND ADD IT TO R4, WHICH WAS CLEARED
                                ;*      AT THE BEGINING OF THIS SUBROUTINE.  LOCATION 'LRGSTC' IS TO BE LOADED
                                ;*      WITH THE LARGEST ASCII CHARACTER ACCEPTABLE FOR THIS NUMBER, I.E. 7
                                ;*      OR 9 (FOR OCTAL OP DECIMAL INPUT RESPECTIVELY).  IT WILL ONLY ACCEPT
                                ;*      THE NUMBER DIGIT EQUAL TO OR LESS THAN THIS DIGIT.  IT WILL ONLY ACCEPT
                                ;*      THE MAXIMUM NUMBER OF DIGITS SPECIFIED BY R3, PRINTING A <CRLF> WHEN
                                ;*      THAT LIMIT IS REACHED.  IF A <CR> IS INPUTED BEFORE THE MAXIMUM IS
                                ;*      REACHED, ROUTINE EXITS LEAVING THE INPUTED NUMBER IN R4.
                                ;*
                                ;*******************************************************************
002722  005004                  READ:   CLR     R4              ;CLEAR THE CHARACTER RECEIVER
002724  105737  030707                  TSTB    CHARCT          ;SEE IF A CHARACTER WAS INPUTED DURING PRINTING
002730  001406                          BEQ     1$              ;BRANCH TO INPUT A CHARACTER IF NOT
002732  113737  030707  002664          MOVB    CHARCT,ANSWER   ;MOVE THE CHARACTER TO THE ANSWER LOCATION
002740  105037  030707                  CLRB    CHARCT          ;CLEAR THAT SUCKER
002744  000403                          BR      2$              ;GO CHECK IT OUT, YOU DUMMY
002746  104411                  1$:     RDCHR                   ;GET A CHARACTER
002750  012637  002664                  MOV     (SP)+,ANSWER    ;POP INPUTED CHARACTER OFF STACK
002754  022737  000003  002664  2$:     CMP     #CNTLC,ANSWER   ;SEE IF A ^C WAS INPUTED
002762  001003                          BNE     3$              ;BRANCH AROUND ITS PRINTING IF NOT
002764  104401  035073                  TYPE    ,CNTRLC         ;TYPE:  '^C'
002770  000453                          BR      6$              ;KICK OUT OF THIS ROUTINE
002772  022737  000033  002664  3$:     CMP     #ESC,ANSWER     ;SEE IF AN <ESC> WAS INPUTED
003000  001003                          BNE     4$              ;BRANCH AROUND ITS PRINTING IF NOT
003002  104401  034762                  TYPE    ,ESCAPE         ;TYPE:  '<ESC>'
003006  000444                          BR      6$              ;KICK OUT OF THIS ROUTINE
003010  113737  002664  035071  4$:     MOVB    ANSWER,LETNCR   ;MOVE CHARACTER FOR PRINTING
003016  104401  035071                  TYPE    ,LETNCR         ;GO TYPE THE INPUTED CHARACTER
003022  022737  000057  002664          CMP     #'/,ANSWER      ;SEE IF A NON-NUMERIC/ALPH CHARACTER WAS INPUTED
003030  100016                          BPL     5$              ;BRANCH TO R4 TEST IF SO
003032  123737  002704  002664          CMPB    LRGSTC,ANSWER   ;SEE IF A NON-OCTAL/NUMERIC CHARACTER WAS INPUTED
003040  100427                          BMI     6$              ;BRANCH TO EXIT IF SO
003042  013746  002664                  MOV     ANSWER,-(SP)    ;MOVE ASCII TO STACK FOR PREPARATION
003046  042716  000060                  BIC     #60,(SP)        ;CLEAR ALL BUT THE NUMBER INPUTED
003052  006304                          ASL     R4              ;SHIFT R4 THREE PLACES
003054  006304                          ASL     R4              ;TO MAKE ROOM FOR
003056  006304                          ASL     R4              ;THIS CHARACTER
003060  062604                          ADD     (SP)+,R4        ;ADD THE OCTAL NUMBER TO R4
003062  005303                          DEC     R3              ;SUBTRACT 1 FROM THE LOOP COUNTER AND
003064  001330                          BNE     1$              ;BRANCH BACK IF NOT ALL CHARACTERS INPUTED
003066  122737  000071  002704  5$:     CMPB    #'9,LRGSTC      ;SEE IF NUMBER IS TO BE DECIMAL
003074  001011                          BNE     6$              ;BRANCH IF NOT
003076  022704  000007                  CMP     #7,R4           ;SEE IF INPUTED NUMBER IS 7 OR LESS
003102  100006                          BPL     6$              ;BRANCH IF SO
003104  022737  000015  002664          CMP     #CARETN,ANSWER  ;SEE IF CARRIAGE RETURN WAS INPUTED
003112  001402                          BEQ     6$              ;BRANCH IF SO - R4 IS CORRECT
003114  062704  000002                  ADD     #2,R4           ;ADD 2 TO R4 TO MAKE OCTAL NUMBER THE DECIMAL EQUIVALENT
003120  104401  001405          6$:     TYPE    ,$CRLF          ;PRINT A <CRLF>
003124  000207                          RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO CAPTURE UNIT #, PASS # & TOTAL ERRORS
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE IS CALLED BY $EOP AND ENDEV TO SAVE THE UNIT NUMBER,
                                ;*      PASS NUMBER, AND TOTAL ERRORS FOR THAT DEVICE/PASS FOR SAVING WHENEVER
                                ;*      A DEVICE PASS CONTAINS ERRORS.
                                ;*
                                ;*******************************************************************
003126  022737  044736  042454  ERCAPT: CMP     #ENDSTK,CAPNTR  ;SEE IF STACK IS FULL OF BULL
003134  001414                          BEQ     1$              ;KICK OUT IF FULL
003136  013700  042454                  MOV     CAPNTR,R0       ;MOVE CAPNTR CONTENTS TO R0
003142  013720  001422                  MOV     $UNIT,(R0)+     ;PUT UNIT NUMBER ON STACK
003146  013720  002716                  MOV     $PASS2,(R0)+    ;PUT OVERFLOW PASS NUMBER ON STACK
003152  013720  001416                  MOV     $PASS,(R0)+     ;PUT PASS NUMBER ON STACK
003156  013720  001312                  MOV     $ERTTL,(R0)+    ;PUT TOTAL ERRORS ON STACK
003162  010037  042454                  MOV     R0,CAPNTR       ;RESTORE CAPNTR TO NEW POINTER VALUE
003166  000207                  1$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO PRINT THE DATA STORED BY SUBROUTINE ERCAPT
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE IS INVOKED BY ENTERING (^Y) DURING THE EXECUTION OF THE
                                ;*      TEST.  IT PRINTS ALL DATA STORED IN THE 'CAPSTK' STACK STORED BY THE
                                ;*      'ERCAPT' SUBROUTINE, REINITIALIZES THE POINTER, AND RETURNS.  *DATA*
                                ;*      *PRINTED*IS*LOST* BECAUSE OF THE REINITIALIZATION OF THE COUNTER.
                                ;*
                                ;*******************************************************************
003170  022737  042456  042454  PTCAPT: CMP     #CAPSTK,CAPNTR  ;SEE IF THERE IS ANY DATA TO PRINT
003176  001003                           BNE     1$              ;BRANCH TO PRINT IT IF THERE IS
003200  104401  034211                   TYPE    ,NODATA         ;TYPE:  'NO ERROR TOTALS TO REPORT'
003204  000462                           BR      7$              ;KICK OUT
003206  022737  044736  042454  1$:      CMP     #ENDSTK,CAPNTR  ;SEE IF STACK IS FULL OF BULL
003214  001002                           BNE     2$              ;BRANCH AROUND STACK IS FULL MESSAGE IF NOT
003216  104401  034020                   TYPE    ,STKIFL         ;TYPE:  'STACK IS FULL - DATA MAY HAVE BEEN LOST'
003222  104401  034073           2$:     TYPE    ,ERCHDR         ;TYPE:  'SUMMATION OF ERRORS SINCE BEGINNING OR LAST REPORT'
                                                                ;       'BOARD #   PASS #   ERRTTL'
003226  012700  042456                   MOV     #CAPSTK,R0      ;MOVE ADDRESS OF STACK TO PRINT TO R0
003232  012701  000020                   MOV     #16.,R1         ;MOVE 16 BOARDS TO SEARCH TO R0
003236  005037  001362                   CLR     $TMP1           ;CLEAR $TMP1, DEVICE POINTER
003242  021037  001362           3$:     CMP     (R0),$TMP1      ;SEE IF UNIT NUMBER IS TO PRINT
003246  001403                           BEQ     4$              ;BRANCH TO PRINT DATA IF SO
003250  062700  000010                   ADD     #10,R0          ;GO TO NEXT SET OF DATA AND
003254  000420                           BR      6$              ;GO SEE IF ANY MORE TO CHECK
003256  012046           4$:     MOV     (R0)+,-(SP)     ;MOVE UNIT NUMBER TO STACK FOR PRINTING
003260  104405                   TYPDS                   ;GO TYPE UNIT NUMBER IN DECIMAL
003262  104401  035250           TYPE    ,SPACES         ;TYPE 2 SPACES
003266  012046                   MOV     (R0)+,-(SP)     ;MOVE OVERFLOW PASS NUMBER TO STACK FOR PRINTING
003270  001002                   BNE     5$              ;BRANCH IF NON-ZERO
003272  104401  035250           TYPE    ,SPACES         ;TYPE AN EXTRA 2 SPACES - NUMBER WILL NOT BE EXTENDED
003276  012046           5$:     MOV     (R0)+,-(SP)     ;MOVE PASS NUMBER TO STACK FOR PRINTING
003300  104406                   TYPDE                   ;GO TYPE PASS NUMBER IN EXTENDED DECIMAL
003302  104401  035250           TYPE    ,SPACES         ;TYPE 2 SPACES
003306  012046                   MOV     (R0)+,-(SP)     ;MOVE ERROR TOTAL TO STACK FOR PRINTING
003310  104405                   TYPDS                   ;GO TYPE ERROR TOTAL IN DECIMAL
003312  104401  001405           TYPE    ,$CRLF          ;TYPE <CRLF>
003316  020037  042454   6$:     CMP     R0,CAPNTR       ;SEE IF ALL DATA HAS BEEN PRINTED
003322  001347                   BNE     3$              ;BRANCH BACK IF NOT
003324  005237  001362           INC     $TMP1           ;INCREMENT DEVICE POINTER
003330  012700  042456           MOV     #CAPSTK,R0      ;INITIALIZE TO BEGINNING FOR ANOTHER POSSIBLE PASS
003334  005301                   DEC     R1              ;DECREMENT THE LOOP COUNTER AND
003336  001341                   BNE     3$              ;BRANCH UNTIL ALL DEVICE DATA PRINTED
003340  012737  042456  042454   MOV     #CAPSTK,CAPNTR  ;REINITIALIZE CAPNTR
003346  104401  001405           TYPE    ,$CRLF          ;TYPE ANOTHER <CRLF>
003352  105037  030707   7$:     CLRB    CHARCT          ;CLEAR ANY CHARACTER ENTERED DURING PRINTING
003356  000207                   RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO FILL ALL TABLE BOARD ENTRIES
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE FILLS ALL TABLE BOARD ENTRIES FOR THE ADDRESSES AND
                                ;*      VECTORS FROM THE VALUES IN 'REGADR' AND 'VECADR', AND SHOULD BE SET
                                ;*      UP BEFORE ENTERING THIS SUBROUTINE.
                                ;*
                                ;*******************************************************************
003360  005037  001466          FIXTBL: CLR     $DEVM           ;CLEAR THE DEVICE MASK
003364  000261                          SEC                     ;SET THE CARRY BIT AND
003366  006137  001466                  ROL     $DEVM           ;ROTATE IT INTO $DEVM FOR 1 BOARD
003372  022737  000001  002414          CMP     #1,QTYBRD       ;SEE IF ONLY 1 BOARD PRESENT
003400  001433                          BEQ     2$              ;KICK OUT IF SO - TABLE DOESN'T NEED FILLING
003402  013701  002414                  MOV     QTYBRD,R1       ;FILL ALL TABLE BOARD ENTRIES FROM FIRST
003406  005301                          DEC     R1              ;DECREMENT SINCE 1ST POSITION IS ALREADY FILLED
003410  005002                          CLR     R2              ;CLEAR INDEX TO SEND POINTER
003412  012703  000002                  MOV     #2,R3           ;PUT 2 IN INDEX TO RECEIVE POINTER
003416  000261                  1$:     SEC                     ;SET THE CARRY BIT AND
003420  006137  001466                  ROL     $DEVM           ;ROTATE IT INTO $DEVM
                002416          RA=REGADR               ;REDEFINE REGADR AS RA FOR SPACE REASONS
003424  016263  002416  002416          MOV     RA(R2),RA(R3)   ;TRANSFER ADDRESS TO NEXT POSITION AND
003432  062763  000010  002416          ADD     #10,RA(R3)      ;ADD 10 FOR NEXT POSITION
                002456          VA=VECADR               ;REDEFINE VECADR AS VA FOR SPACE REASONS
003440  016263  002456  002456          MOV     VA(R2),VA(R3)   ;TRANSFER VECTOR TO NEXT POSITION AND
003446  062763  000010  002456          ADD     #10,VA(R3)      ;ADD 10 FOR NEXT POSITION
003454  013763  001474  001474          MOV     $DDWO,$DDWO(R3) ;MOVE DEVICE DESCRIPTOR WORD TO NEXT POSITION
003462  022223                          CMP     (R2)+,(R3)+     ;UPDATE INDEX POINTERS
003464  005301                          DEC     R1              ;DECREMENT THE LOOP COUNTER
003466  001353                          BNE     1$              ;BRANCH BACK IF NOT DONE
003470  000207                  2$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO LOAD INBUF WITH AN INCREMENTING PATTERN
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE CLEARS THE FIRST LOCATION OF THE BUFFER AND LOADS
                                ;*      NUMBERS STARTING WITH 1 INTO THE BUFFER.
                                ;*
                                ;*******************************************************************
003472  013702  002616    LODBUF: MOV     INBUF,R2        ;MOVE STARTING ADDRESS OF INBUF TO R2
003476  013703  002622            MOV     BUFLEN,R3       ;MOVE LOOP COUNTER TO R3
003502  005203                    INC     R3              ;CORRECT COUNTER
003504  005001                    CLR     R1              ;CLEAR THE LOADING COUNTER
003506  010122            1$:     MOV     R1,(R2)+        ;LOAD NEXT BUFFER WORD
003510  005201                    INC     R1              ;INCREMENT THE LOADING COUNTER
003512  005303                    DEC     R3              ;DECREMENT THE LOOP COUNTER AND
003514  001374                    BNE     1$              ;BRANCH BACK IF NOT DONE
003516  000207                    RTS     PC              ;EXIT
```

```
1741                                            .SBTTL  SUBROUTINE TO LOAD THE CHKBUF WITH EVEN #'S STARTING WITH 0
1742                                    ;*******************************************************************
1743                                    ;*
1744                                    ;*      THIS SUBROUTINE CLEARS THE FIRST LOCATION OF THE BUFFER AND LOADS
1745                                    ;*      EVEN NUMBERS STARTING WITH 0 INTO THE BUFFER.
1746                                    ;*
1747                                    ;*******************************************************************
1748 003520  013702  002620            CHKBFF: MOV     CHKBUF,R2       ;STARTING ADDRESS OF CHECK-BUFFER TO R2
1749 003524  013701  002622                    MOV     BUFLEN,R1       ;MOVE LOOP COUNTER TO R1
1750 003530  005003                            CLR     R3              ;WIPE OUT R3
1751 003532  010322                    1$:     MOV     R3,(R2)+        ;MOVE R3 TO CHKBUF ADDRESS AND INC BY 2
1752 003534  010322                            MOV     R3,(R2)+        ;MOVE R3 TO NEXT CHKBUF ADDRESS AND INC BY 2
1753 003536  022341                            CMP     (R3)+,-(R1)     ;ADD 2 TO NUMBER FOR BUFFER & SUBTRACT 2 FROM LOOP COUNTER
1754 003540  005701                            TST     R1              ;SEE IF R1 HAS REACHED ZERO YET
1755 003542  001373                            BNE     1$              ;BRANCH BACK IF NOT DONE
1756 003544  000207                    2$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO CLEAR IE, CHECK ERROR, READY, WCR=0, AND BAR
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE HAS THE NEED TO CALL AN ERROR IN THE TEST.  THE ERROR
                                ;*      IS TO BE LOCATED IN THE TEST JUST AFTER THE JSR CALL.  FUTURE USE OF
                                ;*      THIS SUBROUTINE MUST BE HANDLED AS FOLLOWS:
                                ;*
                                ;*              JSR     PC,INTA         ;SUBROUTINE CALL
                                ;*              ERROR   +51             ;ERROR CALL
                                ;*              CONTINUE                ;SUBROUTINE RETURNS HERE IF NO ERROR
                                ;*
                                ;*******************************************************************
003546  042777  000100  176746 INTA:   BIC     #IE,@CSR        ;CLEAR IE
003554  013702  002622          MOV     BUFLEN,R2       ;BUFFER LENGTH TO R2
003560  063702  002622          ADD     BUFLEN,R2       ;NUMBER OF XFERS TIMES 2
003564  063702  002616          ADD     INBUF,R2        ;CORRECT BAR
003570  017737  176726  002560  MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
003576  032737  010000  002560  BIT     #MA,RCSR        ;SEE IF WE ARE IN MAINTENANCE MODE
003604  001005                  BNE     1$              ;BRANCH AROUND CABLE TEST IF WE ARE, BIT 0 WILL BE CLEAR
003606  032737  000004  002720  BIT     #BIT2,DDW       ;SEE IF THERE IS A CABLE
003614  001401                  BEQ     1$              ;BRANCH IF NO CABLE
003616  005202                  INC     R2              ;CABLE MODE TESTING LEAVES BIT 0 OF BAR SET. CHECK ODD ADRS
003620  017737  176674  002566 1$:     MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
003626  005037  002602          CLR     EWCR            ;CLEAR EXPECTED LOCATION
003632  010237  002600          MOV     R2,EBAR         ;MOVE EXPECTED DATA TO EBAR
003636  013737  002560  002572  MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
003644  042737  100000  002572  BIC     #ER,ECSR        ;MAKE SURE ERROR BIT IS CLEAR
003652  052737  000200  002572  BIS     #RY,ECSR        ;MAKE SURE READY BIT IS SET
003660  017737  176632  002570  MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
003666  001012                  BNE     2$              ;BRANCH TO RETURN WITHOUT UPDATING PC SO ERROR WILL CALL
003670  023737  002560  002572  CMP     RCSR,ECSR       ;DOES CSR CONTAIN WHAT IT SHOULD
003676  001006                  BNE     2$              ;BRANCH TO RETURN WITHOUT UPDATING PC SO ERROR WILL CALL
003700  023737  00256   002600  CMP     RBAR,EBAR       ;DOES BAR CONTAIN WHAT IT SHOULD
003706  001002                  BNE     2$              ;BRANCH TO RETURN WITHOUT UPDATING PC SO ERROR WILL CALL
003710  062716  000002          ADD     #2,(SP)         ;CORRECT PC RETURN TO AFTER THE ERROR CALL
003714  000207          2$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO CHECK INBUF AFTER A MAINTENANCE MODE OPERATION
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE HAS THE NEED TO CALL AN ERROR IN THE TEST AND RETURN
                                ;*      TO THE SUBROUTINE.  THE ERROR AND RETURN JSR ARE TO BE LOCATED IN THE
                                ;*      TEST JUST AFTER THE JSR CALL.  YOU *MUST* CLEAR LOCATION 'ERRCNT'
                                ;*      BEFORE EXECUTION OF THIS SUBROUTINE, OTHERWISE YOU MAY NOT GET ANY
                                ;*      ERRORS PRINTED, OR IF SO, JUST THE DATA WITHOUT THE HEADER. FUTURE
                                ;*      USE OF THIS SUBROUTINE MUST BE HANDLED AS FOLLOWS:
                                ;*
                                ;*      JSR     PC,DATCHK       ;SUBROUTINE CALL
                                ;*      ERROR   +201            ;ERROR CALL
                                ;*      JSR     PC,DATCH2       ;RETURN TO SUBROUTINE AFTER ERROR RTS
                                ;*      CONTINUE                ;SUBROUTINE RETURNS HERE IF NO ERROR(S) OR WHEN DONE
                                ;*
                                ;*******************************************************************
003716  011637  001362          DATCHK: MOV     (SP),$TMP1      ;SAVE PC RETURN
003722  013702  002620                  MOV     CHKBUF,R2       ;STARTING ADDRESS OF CHECK BUFFER TO R2
003726  013703  002616                  MOV     INBUF,R3        ;STARTING ADDRESS OF IN BUFFER TO R3
003732  005037  002624                  CLR     LENCHK          ;CLEAR LENGTH CHECK
003736  005237  002624          DATCH1: INC     LENCHK          ;MAKE A COMPARISON
003742  022223                          CMP     (R2)+,(R3)+     ;IS THE DATA CORRECT?
003744  001423                          BEQ     DATCH2          ;BRANCH IF OK
003746  013716  001362                  MOV     $TMP1,(SP)      ;RESTORE ORIGINAL PC RETURN
003752  016237  177776  001364          MOV     -2(R2),$TMP2    ;MOVE CHECK BUFFER CONTENTS TO $TMP2
003760  010237  001370                  MOV     R2,$TMP4        ;MOVE ADDRESS +2 TO $TMP4
003764  162737  000002  001370          SUB     #2,$TMP4        ;CORRECT SO IT POINTS TO ADDRESS CAUSING ERROR
003772  016337  177776  001366          MOV     -2(R3),$TMP3    ;MOVE INPUT BUFFER CONTENTS TO $TMP3
004000  010337  001372                  MOV     R3,$TMP5        ;MOVE ADDRESS +2 TO $TMP5
004004  162737  000002  001372          SUB     #2,$TMP5        ;CORRECT SO IT POINTS TO ADDRESS CAUSING ERROR
004012  000207                          RTS     PC              ;RETURN TO ERROR CALL - PC ON STACK ALREADY POINTS THERE
004014  023737  002624  002622  DATCH2: CMP     LENCHK,BUFLEN   ;SEE IF THE BUFFER HAS BEEN CHECKED
004022  001345                          BNE     DATCH1          ;GO BACK FOR ANOTHER TRY IF NOT
004024  013716  001362                  MOV     $TMP1,(SP)      ;RESTORE PC RETURN
004030  062716  000006                  ADD     #6,(SP)         ;CORRECT IT SO RETURN IS AFTER THE ERROR CALL
004034  000207                          RTS     PC              ;RETURN
```

```
                                        .SBTTL  SUBROUTINE TO RESTORE DR11 INT VECT & SET CPU PRIORITY TO 7.
                                ;*******************************************************************
                                ;*
                                ;*      THIS ROUTINE IS USED IN VARIOUS TESTS TO CLEAR ANY DATA THAT
                                ;*      MAY BE LEFT IN ANY REGISTERS, AND RESTORE CPU PRIORITY TO 7.
                                ;*
                                ;*******************************************************************
004036  012737  000340  177776  CLENUP: MOV     #LEVEL7,PSW     ;RESTORE CPU TO PRIORITY LEVEL 7
004044  012777  010000  176450          MOV     #MA,@CSR        ;DO AN INIT CLEARING WCR, BAR & BDR BY SETTING
004052  005077  176444                  CLR     @CSR            ;AND CLEARING THE MAINT BIT AND CLEAR THE CSR
004056  000207                          RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO CHECK FOR CABLE MODE AND ALTER EXPECTED DATA
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE CHECKS THE DDW (DEVICE DESCRIPTOR WORD) FOR THE CABLE
                                ;*      BEING IN OR OUT AND SETS BITS 12, 10, 8 AND 6 IN THE EXPECTED DATA
                                ;*      LOCATION IF THE CABLE IS OUT.
                                ;*
                                ;*******************************************************************
004060  032737  000004  002720  CHKCAB: BIT     #BIT2,DDW       ;CHECK FOR CABLE STATUS
004066  001006                          BNE     1$              ;EXIT IF CABLE DOES EXIST
004070  052737  127000  002572          BIS     #127000,ECSR    ;SET BITS 12, 10, 8, 7 & 6 - CABLE DOESN'T EXIST
004076  052737  127000  002536          BIS     #127000,BUT     ;SET BITS 12, 10, 8, 7 & 6 IN BITS UNDER TEST LOCATION TOO
004104  000207                  1$:     RTS     PC              ;RETURN
```

```
                                        .SBTTL  SUBROUTINE TO CHECK CORRECT DATA PATTERN WAS MOVED TO INBUF
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE HAS THE NEED TO CALL 2 ERRORS.  THE ERRORS ARE TO BE
                                ;*      LOCATED IN THE TEST JUST AFTER THE JSR CALL.  FUTURE USE OF THIS
                                ;*      SUBROUTINE MUST BE HANDLED AS FOLLOWS:
                                ;*
                                ;*      JSR     PC,DATOCK       ;SUBROUTINE CALL
                                ;*      ERROR   +46             ;ERROR CALL
                                ;*      JSR     PC,DATOC2       ;RETURN TO SUBROUTINE AFTER ERROR RTS
                                ;*      ERROR   +47             ;2ND ERROR CALL
                                ;*      CONTINUE                ;SUBROUTINE RETURNS HERE WHEN DONE
                                ;*
                                ;*******************************************************************
004106  011637  001362          DATOCK: MOV     (SP),$TMP1      ;SAVE PC RETURN
004112  012702  052525                  MOV     #52525,R2       ;DATO NUMBER TO R2
004116  013703  002616                  MOV     INBUF,R3        ;STARTING ADDRESS OF IN BUFFER TO R3
004122  005037  002624                  CLR     LENCHK          ;CLEAR LENGTH CHECK
004126  005237  002624          DATOC1: INC     LENCHK          ;MAKE A COMPARISON
004132  020223                          CMP     R2,(R3)+        ;IS THE DATA CORRECT?
004134  001415                          BEQ     DATOC2          ;BRANCH IF OK
004136  013716  001362                  MOV     $TMP1,(SP)      ;MOVE OLD PC RETURN TO STACK
004142  010237  001364                  MOV     R2,$TMP2        ;MOVE EXPECTED DATA TO $TMP2
004146  016337  177776  001366          MOV     -2(R3),$TMP3    ;MOVE RECEIVED DATA TO $TMP3
004154  010337  001370                  MOV     R3,$TMP4        ;MOVE ADDRESS +2 TO $TMP4
004160  162737  000002  001370          SUB     #2,$TMP4        ;CORRECT ADDRESS SO IT POINTS TO ADDRESS CAJSING ERROR
004166  000207                          RTS     PC              ;RETURN TO ERROR CALL
004170  023737  002624  002622  DATOC2: CMP     LENCHK,BUFLEN   ;SEE IF THE BUFFER HAS BEEN CHECKED
004176  001353                          BNE     DATOC1          ;BUFFER CHECKED?
004200  020223                          CMP     R2,(R3)+        ;CHECK END OF BUFFER + 1
004202  001017                          BNE     1$              ;BRANCH IF NOT LOADED
004204  010237  001364                  MOV     R2,$TMP2        ;MOVE EXPECTED DATA TO $TMP2
004210  016337  177776  001366          MOV     -2(R3),$TMP3    ;MOVE RECEIVED DATA TO $TMP3
004216  010337  001370                  MOV     R3,$TMP4        ;MOVE ADDRESS +2 TO $TMP4
004222  162737  000002  001370          SUB     #2,$TMP4        ;CORRECT ADDRESS SO IT POINTS TO ADDRESS CAUSING ERROR
004230  013716  001362                  MOV     $TMP1,(SP)      ;CORRECT PC RETURN
004234  062716  000006                  ADD     #6,(SP)         ;POINT TO 2ND ERROR CALL AFTER JSR PC,DATOCK
004240  000207                          RTS     PC              ;RETURN TO ERROR CALL
004242  013716  001362          1$:     MOV     $TMP1,(SP)      ;RESTORE RETURN ADDRESS
004246  062716  000010                  ADD     #10,(SP)        ;POINT TO PROPER RETURN AFTER THE ERROR CALLS
004252  000207                          RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO CLEAR IE AND HALT IF ERROR IS SET
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE HAS THE NEED TO CALL AN ERROR IN THE TEST.  THE ERROR
                                ;*      IS TO BE LOCATED IN THE TEST JUST AFTER THE JSR CALL.  FUTURE USE OF
                                ;*      THIS SUBROUTINE MUST BE HANDLED AS FOLLOWS:
                                ;*      JSR     PC,ERRCHK       ;SUBROUTINE CALL
                                ;*      ERROR   +21             ;ERROR CALL
                                ;*      CONTINUE                ;SUBROUTINE RETURNS HERE IF NO ERROR
                                ;*
                                ;*******************************************************************
004254  042777  000100  176240  ERRCHK: BIC     #IE,@CSR        ;CLEAR IE
004262  017737  176234  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
004270  013737  002560  002572          MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
004276  042737  100000  002572          BIC     #ER,ECSR        ;CLEAR THE ERROR BIT
004304  052737  000200  002572          BIS     #RY,ECSR        ;SET THE READY BIT
004312  013701  002560                  MOV     RCSR,R1         ;MOVE DATA TO R1 FOR CHECKING
004316  012737  000200  001360          MOV     #RY,$TMP0       ;MOVE EXPECTED DATA TO $TMP0
004324  042701  077577                  BIC     #77577,R1       ;CLEAR ALL BUT THE ERROR AND READY BITS
004330  022701  000200                  CMP     #RY,R1          ;SEE IF ERROR BIT IS CLEAR AND READY IS SET
004334  001002                          BNE     1$              ;BRANCH AROUND PC CORRECTION SO ERROR WILL CALL
004336  062716  000002                  ADD     #2,(SP)         ;CORRECT PC RETURN - DATA OK
004342  000207                  1$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO GENERATE DEVICE ADDRESS TABLE
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE GENERATES AN ADDRESS TABLE LOCATED AT REGADR STARTING
                                ;*      WITH THE BASE DEVICE ADDRESS (CONTENTS OF $BASE) IN STEPS OF 10.
                                ;*
                                ;*******************************************************************
004344  012702  002416          DEVADS: MOV     #REGADR,R2      ;POINT R2 TO THE DEVICE ADDRESS TABLE
004350  013700  001464                  MOV     $BASE,R0        ;LOAD BASE DEVICE ADDRESS IN R0
004354  012701  000020                  MOV     #16.,R1         ;MOVE LOOP COUNTER TO R1
004360  010022                  1$:     MOV     R0,(R2)+        ;MOVE DEVICE ADDRESS TO TABLE
004362  062700  000010                  ADD     #10,R0          ;POINT R0 TO NEXT DEVICE ADDRESS
004366  005301                          DEC     R1              ;DECREMENT THE LOOP COUNTER AND
004370  001373                          BNE     1$              ;BRANCH IF NOT ALL DONE YET
004372  000207                          RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO RESET THE ".+2" AND "BPT" LOCATIONS
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE LOADS ".+2" AND "BPT" INTO ALL UNUSED LOCATIONS
                                ;*      BETWEEN 4-776.
                                ;*
                                ;*******************************************************************
004374  012700  004436          BPINIT: MOV     #BPTINT,R0      ;POINT R0 TO TABLE OF BPT INIT LOCATIONS
004400  012701  000003                  MOV     #3,R1           ;DO 3 SETS OF ".+2" AND "BPT" SETUPS
004404  012002                  1$:     MOV     (R0)+,R2        ;MOVE START ADDRESS TO R2
004406  012003                          MOV     (R0)+,R3        ;MOVE END ADDRESS TO R3
004410  010204                          MOV     R2,R4           ;MOVE ADDRESS TO R4
004412  005724                          TST     (R4)+           ;INCREMENT R4 TO PRODUCE THE ".+2" NUMBER
004414  010422                  2$:     MOV     R4,(R2)+        ;MOVE THE NUMBER TO THE LOCATION
004416  012722  000003                  MOV     #BPT,(R2)+      ;MOVE "BPT" TO THE NEXT LOCATION
004422  022424                          CMP     (R4)+,(R4)+     ;ADD 4 TO R4
004424  020203                          CMP     R2,R3           ;SEE IF WE HAVE DONE ALL FOR THIS LOCATION
004426  001372                          BNE     2$              ;BRANCH BACK FOR ANOTHER TRANSFER IF NOT
004430  005301                          DEC     R1              ;DECREMENT R1
004432  001364                          BNE     1$              ;BRANCH BACK IF 3 GROUPS NOT DONE
004434  000207                          RTS     PC              ;EXIT

004436  000004  000014  000050  BPTINT: .WORD   4,14,50         ;ADDRESSES USED TO PUT ".+2" & "BPT" BACK
004444  000174  000254  001000          .WORD   174,254,1000
```

```
                        .SBTTL  SUBROUTINE TO EXTRACT INFORMATION ABOUT THE DR11
                ;*******************************************************************
                ;*
                ;*      THIS SUBROUTINE EXTRACTS INFORMATION ABOUT THE DR11 THAT INTERRUPTED
                ;*      AND LOADS THE DATA FOUND INTO THE DEVICE DESCRIPTOR WORD FOR THAT
                ;*      BOARD.
                ;*
                ;*******************************************************************
004452 012710 010000            DRGET:  MOV     #MA,(R0)        ;SET THE MAINTENANCE BIT AND
004456 005010                           CLR     (R0)            ;CLEAR THE CSR TO DO AN INIT
004460 016612 000002                    MOV     2(SP),(R2)      ;MOVE VECTOR+4 FOR THIS MODULE TO VECTOR LOCATION
004464 012616                           MOV     (SP)+,(SP)      ;MOVE RETURN OF THIS SUBROUTINE TO ITS PROPER POSITION
004466 162712 000004                    SUB     #4,(R2)         ;DUMB VECTOR IS WRONG - CORRECT IT
004472 013711 002540                    MOV     LEVEL,(R1)      ;PUT THE PRIORITY LEVEL INTO THE $DDWXX LOCATION
004476 005237 001466                    INC     $DEVM           ;INDICATE DEVICE EXISTENCE IN DEVICE MAP
004502 005237 002414                    INC     QTYBRD          ;INCREMENT DEVICE COUNT
004506 052710 100000                    BIS     #EIR,(R0)       ;GO TO EIR TO GET B/W STATE
004512 011037 002562                    MOV     (R0),REIR       ;MOVE EIR TO REIR
004516 005010                           CLR     (R0)            ;GO BACK TO CSR
004520 012710 010000                    MOV     #MA,(R0)        ;SET THE MAINT BIT
004524 005010                           CLR     (R0)            ;DO AN INIT
004526 032737 000001 002562             BIT     #BIT0,REIR      ;TEST FOR B/W STATE
004534 001003                           BNE     1$              ;BRANCH IF A W
004536 052711 000001                    BIS     #BIT0,(R1)      ;SET STATE IN DEVICE DESCRIPTOR WORD
004542 000406                           BR      2$              ;GO TO CABLE STATUS TEST
004544 032737 000400 002562     1$:     BIT     #N2,REIR        ;CHECK 2/N CYCLE STATE
004552 001402                           BEQ     2$              ;BRANCH IF 2 CYCLE
004554 052711 000002                    BIS     #BIT1,(R1)      ;N CYCLE - SET BIT IN DEVICE DESC
004560 011037 002560            2$:     MOV     (R0),RCSR       ;MOVE RECEIVED DATA TO RCSR TO GET CABLE STATUS
004564 032737 127000 002560             BIT     #127000,RCSR    ;CHECK IF ANY BITS ARE SET - THEY ARE IF NO CABLE
004572 001015                           BNE     3$              ;BRANCH IF NO CABLE
004574 112710 000004                    MOVB    #F2,(R0)        ;CABLE IS POSSIBLY IN - SET FNCT2
004600 011037 002560                    MOV     (R0),RCSR       ;MOVE RECEIVED DATA TO RCSR
004604 052710 010000                    BIS     #MA,(R0)        ;SET THE MAINTENANCE BIT
004610 005010                           CLR     (R0)            ;CLEAR THE CSR TO DO AN INIT
004612 032737 020000 002560             BIT     #AT,RCSR        ;TEST TH  ATTN BIT
004620 001402                           BEQ     3$              ;BRANCH IF NOT SET - NO CABLE
004622 052711 000004                    BIS     #BIT2,(R1)      ;SET CABLE BIT IN DEVICE DESC
004626 000207                   3$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO PRINT THE AUTOSIZED BOARD CONFIGURATIONS
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE PRINTS THE BOARD CONFIGURATIONS FOUND BY ASIZE
                                ;*
                                ;*******************************************************************
004630  005037  002662          TYPCNF: CLR     LOOP            ;CLEAR THE BOARD NUMBER COUNTER
004634  013705  001466                  MOV     $DEVM,R5        ;GET DEVICE MAP
004640  012701  002416                  MOV     #REGADR,R1      ;MOVE THE ADDRESS OF THE REGISTER ADDRESS TABLE TO R1
004644  012702  002456                  MOV     #VECADR,R2      ;MOVE THE ADDRESS OF THE VECTOR ADDRESS TABLE TO R2
004650  005003                          CLR     R3              ;CLEAR THE DEVICE DESCRIPTOR ADDRESS POINTER
004652  104401  035433                  TYPE    ,NOBUT          ;TYPE:  'NO. BOARDS UNDER TEST:  '
004656  013746  002414                  MOV     QTYBRD,-(SP)    ;MOVE THE QUANTITY TO THE STACK FOR TYPEOUT
004662  104405                          TYPDS                   ;TYPE THE NUMBER
004664  104401  035472                  TYPE    ,BRVWPC         ;TYPE THE HEADER
004670  012700  000020                  MOV     #16.,R0         ;SET UP LOOP COUNTER FOR 16 BOARDS
004674  032705  000001          1$:     BIT     #BIT0,R5        ;DEVICE UNDER TEST?
004700  001466                          BEQ     8$              ;BRANCH IF NO
004702  013746  002662                  MOV     LOOP,-(SP)      ;PUT BOARD # ON STACK FOR TYPEOUT
004706  104405                          TYPDS                   ;PRINT BOARD # (0 TO 16)
004710  104401  035257                  TYPE    ,SPACE6         ;TYPE 6 SPACE CHARACTERS
004714  011146                          MOV     (R1),-(SP)      ;SAVE REGISTER ADDRESS FOR TYPEOUT
004716  104402                          TYPOC                   ;PRINT DEVICE REGISTER ADDRESS
004720  104401  035257                  TYPE    ,SPACE6         ;TYPE 6 SPACE CHARACTERS
004724  011246                          MOV     (R2),-(SP)      ;SAVE VECTOR ADDRESS FOR TYPEOUT
004726  104403                          TYPOS                   ;PRINT VECTOR ADDRESS
004730     003     000                  .BYTE   3,0             ;PRINT 3 DIGITS, LEADING ZEROS SUPPRESSED
004732  104401  035266                  TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
004736  016337  001474  002720          MOV     $DDW0(R3),DDW   ;MOVE DEVICE DESCRIPTOR WORD TO DDW
004744  032737  000001  002720          BIT     #BIT0,DDW       ;TEST WHICH STATE, B OR W, FOR THIS BOARD
004752  001403                          BEQ     2$              ;GO PRINT W STATE IF W
004754  104401  035303                  TYPE    ,B              ;TYPE A 'B'
004760  000402                          BR      3$              ;GO TO NEXT CHECK
004762  104401  035305          2$:     TYPE    ,W              ;'TYPE A 'W'
004766  104401  035266          3$:     TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
004772  004737  005630                  JSR     PC,PNTPRI       ;PRINT DEVICE PRIORITY
004776  104401  035266                  TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
005002  032737  000002  002720          BIT     #BIT1,DDW       ;TEST 2/N CYCLE STATE
005010  001403                          BEQ     4$              ;GO PRINT 2 STATE IF 2
005012  104401  035320                  TYPE    ,N              ;TYPE AN 'N'
005016  000402                          BR      5$              ;GO TO NEXT CHECK
005020  104401  035316          4$:     TYPE    ,TWO            ;TYPE A '2'
005024  104401  035266          5$:     TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
005030  032737  000004  002720          BIT     #BIT2,DDW       ;TEST CABLE STATE
005036  001403                          BEQ     6$              ;GO PRINT 'NO' IF NO CABLE
005040  104401  035312                  TYPE    ,YES            ;TYPE 'YES'
005044  000402                          BR      7$              ;GO TO LOOP CHECK
005046  104401  035307          6$:     TYPE    ,NO             ;TYPE 'NO'
005052  104401  001405          7$:     TYPE    ,$CRLF          ;TYPE A <CRLF>
005056  005237  002662          8$:     INC     LOOP            ;INCREMENT BOARD COUNT FOR POSSIBLE NEXT PASS
005062  022122                          CMP     (R1)+,(R2)+     ;INCREMENT COUNTERS
005064  006205                          ASR     R5              ;SHIFT R5 TO THE RIGHT TO MOVE BOARD BIT INTO BIT 0
005066  062703  000002                  ADD     #2,R3           ;ADD 2 TO THE DEVICE DESCRIPTOR WORD POINTER
005072  005300                          DEC     R0              ;DECREMENT THE LOOP COUNTER AND
005074  001277                          BNE     1$              ;BRANCH BACK FOR CHECK IF 16 BOARDS NOT DONE
005076  104401  001405                  TYPE    ,$CRLF          ;TYPE ANOTHER <CRLF>
005102  000207                          RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO CHECK FOR LOCATION BELONGING TO A DR11
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE CHECKS FOR THE LOCATION BELONGING TO A DR11.
                                ;*
                                ;*******************************************************************
005104  012737  000340  002540  CHK4DR: MOV     #LEVEL7,LEVEL   ;MOVE PRIORITY 7 TO LEVEL
005112  012703  005264                  MOV     #LEVELS,R3      ;MOVE ADDRESS OF PRIORITY LEVELS TO R3
005116  012704  000004                  MOV     #4,R4           ;DO 4 PRIORITY CHECKS
005122  012737  000400  002660  1$:     MOV     #400,TIME       ;SET UP WAIT LOOP COUNTER
005130  012710  010000                  MOV     #MA,(R0)        ;SET THE MAINTENANCE BIT AND
005134  005010                          CLR     (R0)            ;CLEAR TO POSSIBLY DO AN INIT
005136  013737  000014  001362          MOV     BPTVCT,$TMP1    ;SAVE BPT TRAP VECTOR
005144  012737  005222  000014          MOV     #3$,BPTVCT      ;INTERRUPTS TO 3$
005152  012337  177776                  MOV     (R3)+,PSW       ;SET CPU PRIORITY TO NEXT LEVEL
005156  000240                          NOP                     ;KILL A LITTLE TIME
005160  012710  000105                  MOV     #IE+F2+GO,(R0)  ;SET IE, FNCT2 AND GO ATTEMPTING ANOTHER INTERRUPT
005164  005337  002660          2$:     DEC     TIME            ;DECREMENT TIME
005170  001375                          BNE     2$              ;BRANCH BACK UNTIL ZERO
005172  012737  000340  177776          MOV     #LEVEL7,PSW     ;SET CPU PRIORITY BACK TO 7
005200  013737  001362  000014          MOV     $TMP1,BPTVCT    ;RESTORE BPT TRAP VECTOR
005206  162737  000040  002540          SUB     #40,LEVEL       ;PUT LOCATION 'LEVEL' AT NEXT PRIORITY - INTERRUPT FAILED
005214  005304                          DEC     R4              ;DECREMENT LOOP COUNTER
005216  001341                          BNE     1$              ;BRANCH BACK IF NOT ALL PRIORITY LEVELS CHECKED YET
005220  000416                          BR      4$              ;EXIT - THIS LOCATION DOESN'T BELONG TO A DR11
005222  012737  000340  177776  3$:     MOV     #LEVEL7,PSW     ;AHAH - THIS *IS* A DR11 - SET CPU PRIORITY BACK TO 7
005230  013737  001362  000014          MOV     $TMP1,BPTVCT    ;RESTORE BPT TRAP VECTOR
005236  016666  000010  000006          MOV     10(SP),6(SP)    ;MOVE THIS SUBROUT'NE'S RETURN UP ONE SPOT ON STACK
005244  011666  000010                  MOV     (SP),10(SP)     ;MOVE TRAP ADDRESS TO RETURN'S OLD LOCATION
005250  062706  000006                  ADD     #6,SP           ;KICK GARBAGE' OFF STACK - GOT TO KEEP IT CLEAN
005254  000402                          BR      5$              ;BRANCH TO KICK OUT
005256  062716  000012          4$:     ADD     #12,(SP)        ;CORRECT RETURN TO NOT DO DR11 SETUP
005262  000207                  5$:     RTS     PC              ;KICK OUT
                                
005264  000300  000240          LEVELS: .WORD   LEVEL6,LEVEL5   ;PRIORITY LEVELS TO LOAD INTO THE PSW
005270  000200  000140                  .WORD   LEVEL4,LEVEL3
```

```
                        .SBTTL  SUBROUTINE TO AUTO SIZE DR11 BOARD CONFIGURATION
                ;*******************************************************************
                ;*
                ;*      THIS SUBROUTINE AUTOSIZES THE BOARD CONFIGURATION AND CALLS FOR THE
                ;*      PRINTING OF THE CONFIGURATION IF BIT 12 OF THE SWITCH REGISTER IS
                ;*      CLEAR.
                ;*
                ;*******************************************************************
005274  012700  001474          ASIZE:  MOV     #$DDW0,R0       ;MOVE ADDRESS OF FIRST DEVICE DESCRIPTOR WORD TO R0
005300  012701  000020                  MOV     #16.,R1         ;CLEAR 16 WORDS OUT OF THE DICTIONARY
005304  005020                  1$:     CLR     (R0)+           ;CLEAR THE WORD - SORRY CAN'T LOOK IT UP ANY MORE
005306  005301                          DEC     R1              ;DECREMENT THE LOOP COUNTER AND
005310  001375                          BNE     1$              ;BRANCH BACK IF NOT DONE YET
005312  004737  004374                  JSR     PC,BPINIT       ;GO RESET THE ".+2" AND "BPT" LOCATIONS
005316  004737  004344                  JSR     PC,DEVADS       ;GENERATE DEVICE ADDRESS TABLE
005322  005037  002414                  CLR     QTYBRD          ;CLEAR DEVICE COUNT
005326  005037  001466                  CLR     $DEVM           ;CLEAR DEVICE MAP
005332  013737  000004  002672          MOV     TOVECT,OLDPC1   ;SAVE TIMEOUT VECTOR
005340  012737  005416  000004          MOV     #3$,TOVECT      ;SET TIMEOUT POINTER TO 3$
005346  013737  000006  002674          MOV     TMOPSW,OLDPS1   ;SAVE TIMEOUT PS
005354  012737  000340  000006          MOV     #LEVEL7,TMOPSW  ;CPU PRIORITY TO 7
005362  012700  172604                  MOV     #172604,R0      ;POINT R0 TO UNIT #16 CSR ADDRESS LOCATION
005366  012701  001532                  MOV     #$DDW15,R1      ;LOAD DEVICE DESC ADDRESS
005372  012702  002514                  MOV     #VECADR+36,R2   ;POINT R2 TO UNIT #16 VECTOR ADDRESS LOCATION
005376  012705  000020                  MOV     #16.,R5         ;DO 16 BOARDS
005402  005010                  2$:     CLR     (R0)            ;CHECK FOR REGISTER EXISTENCE
005404  004737  005104                  JSR     PC,CHK4DR       ;GO CHECK FOR A DR11 AT THIS LOCATION IF IT DOES
005410  004737  004452                  JSR     PC,DRGET        ;GO EXTRACT INFO FROM THE DR11
005414  000402                          BR      4$              ;BRANCH OVER THE STACK CORRECTION
005416  062706  000004          3$:     ADD     #4,SP           ;CORRECT STACK AFTER TIMEOUT
005422  162700  000010          4$:     SUB     #10,R0          ;POINT R0 TO NEXT DEVICE ADDRESS
005426  024142                          CMP     -(R1),-(R2)     ;POINT R1 AND R2 TO NEXT DEVICE & VECTOR LOCATIONS
005430  005305                          DEC     R5              ;DECREMENT LOOP COUNTER
005432  001403                          BEQ     5$              ;EXIT IF ALL DONE WITH 16 BOARDS
005434  006337  001466                  ASL     $DEVM           ;ADJUST DEVICE MAP FOR NEXT UNIT CHECK
005440  000760                          BR      2$              ;GO CHECK NEXT LOCATION
005442  013737  002674  000006  5$:     MOV     OLDPS1,TMOPSW   ;RESTORE TIMEOUT PS
005450  013737  002672  000004          MOV     OLDPC1,TOVECT   ;RESTORE TIMEOUT VECTOR
005456  032777  010000  173654          BIT     #BIT12,@SWR     ;CHECK FOR CONFIGURATION PRINT
005464  001005                          BNE     6$              ;BRANCH IF PRINT NOT REQUESTED
005466  005737  002414                  TST     QTYBRD          ;SEE IF ANY BOARDS WERE FOUND
005472  001402                          BEQ     6$              ;BRANCH TO RETURN IF NOT - NO BOARD INFO TO PRINT
005474  004737  004630                  JSR     PC,TYPCNF       ;GO TYPE THE BOARD CONFIGURATIONS
005500  000207                  6$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO GENERATE VECTOR ADDRESS TABLE
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE GENERATES THE VECTOR ADDRESS TABLE.
                                ;*
                                ;*******************************************************************
005502  012702  002456          VCTADS: MOV     #VECADR,R2      ;GET LOCATION OF VECTOR TABLE
005506  013700  001460                  MOV     $VECT1,R0       ;COPY BASE VECTOR
005512  042700  177400                  BIC     #177400,R0      ;CLEAR UPPER BYTE
005516  012701  000020                  MOV     #16.,R1         ;DO 16 VECTORS
005522  010022                  1$:     MOV     R0,(R2)+        ;PUT VECTOR ADDRESS IN TABLE
005524  062700  000010                  ADD     #10,R0          ;POINT R0 TO NEXT VECTOR ADDRESS
005530  005301                          DEC     R1              ;DECREMENT LOOP COUNTER
005532  001373                          BNE     1$              ;BRANCH IF NOT ALL DONE YET
005534  000207                          RTS     PC              ;EXIT
```

```
                                        .SBTTL  ROUTINE TO REPORT UNEXPECTED OR ERRONEOUS TRAPS OR INTERRUPTS
                                ;*******************************************************************
                                ;*
                                ;*      THIS IS THE ROUTINE TO REPORT UNEXPECTED OR ERRONEOUS TRAPS OR INTERRUPTS
                                ;*
                                ;*******************************************************************
005536  012737  000340  177776  CATCH:  MOV     #LEVEL7,PSW     ;REESTABLISH CPU PRIORITY AT 7
005544  012637  002542                  MOV     (SP)+,BDVECT    ;GET ADDRESS OF TRAP VECTOR + 4
005550  012637  002674                  MOV     (SP)+,OLDPS1    ;SAVE PS
005554  012637  002676                  MOV     (SP)+,OLDPC2    ;SAVE PC OF ADDRESS OF INSTRUCTION CAUSING TRAP
005560  012637  002700                  MOV     (SP)+,OLDPS2    ;SAVE 2ND PS
005564  162737  000004  002542          SUB     #4,BDVECT       ;ADJUST TO POINT TO TRAP ADDRESS
005572  104050                          ERROR   +50             ;UNEXPECTED TRAP OR INTERRUPT TO TRAP ADDRESS BELOW
005574  013746  002700                  MOV     OLDPS2,-(SP)    ;RESTORE PS RETURN ON STACK
005600  013746  002676                  MOV     OLDPC2,-(SP)    ;RESTORE PC RETURN ON STACK
005604  000002                          RTI                     ;RETURN
```

```
                                        .SBTTL  SUBROUTINE TO PRINT STATE OF A DDW BIT
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE PRINTS THE STATE OF THE BIT IN THE DDW THAT WAS
                                ;*      PRELOADED INTO BITTST.
                                ;*
                                ;*******************************************************************
005606  005046                  PSTATE: CLR     -(SP)           ;SHOW STATE AS ZERO INITIALLY
005610  033737  002710  002720          BIT     BITTST,DDW      ;CHECK STATE OF BIT IN DDW USING BIT SET IN BITTST
005616  001401                          BEQ     1$              ;BRANCH IF NOT SET
005620  005216                          INC     (SP)            ;SHOW A '1' STATE FOR THAT BIT
005622  104403                  1$:     TYPOS                   ;TYPE THE STATE, LEADING ZEROS SUPPRESSED
005624     001     000                  .BYTE   1,0             ;TYPE 1 CHARACTER, SUPPRESS LEADING ZEROS
005626  000207                          RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO PRINT DEVICE PRIORITY
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE PRINTS THE DEVICE PRIORITY
                                ;*
                                ;*******************************************************************
005630  013746  002720          PNTPRI: MOV     DDW,-(SP)       ;PUT DEVICE DESCRIPTOR WORD ON STACK
005634  006216                          ASR     (SP)            ;SHIFT RIGHT STACK LOCATION 5 PLACES
005636  006216                          ASR     (SP)
005640  006216                          ASR     (SP)
005642  006216                          ASR     (SP)
005644  006216                          ASR     (SP)
005646  042716  177770                  BIC     #177770,(SP)    ;MASK TO GET PRIORITY
005652  104403                          TYPOS                   ;TYPE THE DEVICE PRIORITY
005654     001     000                  .BYTE   1,0             ;TYPE 1 CHARACTER, SUPPRESS LEADING ZEROS
005656  000207                          RTS     PC              ;EXIT
```

```
INITIALIZE THE COMMON TAGS SUBROUTINE

                                        .SBTTL  INITIALIZE THE COMMON TAGS SUBROUTINE
                                ;*****************************************************************
                                ;*
                                ;*      THIS SUBROUTINE INITIALIZES THE INTERRUPT VECTORS. USE IS AS FOLLOWS:
                                ;*      MOV     #STACK,SP       ;INITIALIZE THE STACK
                                ;*      JSR     PC,SETTUP       ;CALL THE SUBROUTINE
                                ;*
                                ;*****************************************************************
005660  011637  001360          SETTUP: MOV     (SP),$TMP0      ;SAVE RETURN ADDRESS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
005664  012706  001300                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
005670  005026                  100$:   CLR     (R6)+           ;;CLEAR MEMORY LOCATION
005672  022706  001340                  CMP     #SWR,R6 ;;DONE?
005676  001374                          BNE     100$            ;;LOOP BACK IF NO
005700  012706  001300                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
005704  012737  027636  000020          MOV     #$SCOPE,@#IOTVEC;;IOT VECTOR FOR SCOPE ROUTINE
005712  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
005720  012737  031050  000030          MOV     #$ERROR,@#EMTVEC;;EMT VECTOR FOR ERROR ROUTINE
005726  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
005734  012737  027550  000034          MOV     #$TRAP,@#TRAPVEC;;TRAP VECTOR FOR TRAP CALLS
005742  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
005750  012737  031774  000024          MOV     #$PWRDN,@#PWRVEC;;POWER FAILURE VECTOR
005756  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
005764  013737  024566  024560          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
005772  005037  001376                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
005776  112737  000001  001315          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
006004  012737  011246  001306          MOV     #TST1+2,$LPADR  ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
006012  012737  011246  001310          MOV     #TST1+2,$LPERR  ;;SETUP THE ERROR LOOP ADDRESS
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
006020  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
006024  012737  006060  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
006032  012737  177570  001340          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
006040  012737  177570  001342          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
006046  022777  177777  173264          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
006054  001011                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
006056  000402                          BR      65$             ;;BRANCH IF NO TIMEOUT
006060  062706  000004          64$:    ADD     #4,SP           ;;CLEAN UP STACK AFTER INTERRUPT
006064  012737  000176  001340  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
006072  012737  000174  001342          MOV     #DISPREG,DISPLAY
006100  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
006104  005037  001416                  CLR     $PASS           ;;CLEAR PASS COUNT
006110  132737  000200  001431          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
006116  001403                          BEQ     67$             ;;YES,USE NON-APT SWITCH
006120  012737  001432  001340          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
006126                          67$:
006126  013746  001360                  MOV     $TMP0,-(SP)     ;PUT RETURN ADDRESS ON STACK AND
006132  000207                          RTS     PC              ;RETURN TO THE CALLING ROUTINE
```

```
                                        .SBTTL  MEMORY MANAGEMENT AND LOCATION CHECK SUBROUTINE
                                ;*****************************************************************
                                ;*
                                ;*      THIS SUBROUTINE CHECKS FOR MEMORY MANAGEMENT EXISTENCE AND WHETHER OR
                                ;*      NOT A LOCATION IN UPPER MEMORY EXISTS.
                                ;*
                                ;*****************************************************************
006134  005737  002712          TSTMM:  TST     MEMGMT          ;TEST TO SEE IF MEMORY MANAGEMENT EXISTS
006140  001440                          BEQ     4$              ;BRANCH IF NOT
006142  012737  000401  006152          MOV     #CY+GO,1$       ;SET UP BIT TEST DATA
006150  040227                          BIC     R2,(PC)+        ;TEST TO SEE IF BOTH THE CYCLE AND GO BITS ARE SET
006152  000401                  1$:     .WORD   CY+GO           ;LOCATION TO STORE THE CYCLE AND GO BITS
006154  001032                          BNE     4$              ;KICK OUT IF CYCLE AND/OR GO ARE CLEAR
006156  032737  000060  002572          BIT     #X6+X7,ECSR     ;SEE IF XBA16 OR XBA17 WERE SET IN EXPECTED DATA
006164  001426                          BEQ     4$              ;BRANCH OUT IF NOT
006166  032737  000040  002572          BIT     #X7,ECSR        ;SEE IF XBA17 IS SET
006174  001005                          BNE     2$              ;GO CHECK STATUS OF XBA16 IF SET
006176  132737  000001  002713          BITB    #BIT0,MEMGMT+1  ;SEE IF 200000+NOCARE WAS FOUND TO EXIST - IF NOT,
006204  001420                          BEQ     5$              ;GO SET EXPECTED ERROR AND NEX BITS AND CHECK FOR ERROR
006206  000415                          BR      4$              ;BRANCH OUT IF LOCATION EXISTS
006210  032737  000020  002572  2$:     BIT     #X6,ECSR        ;SEE IF XBA16 IS SET
006216  001005                          BNE     3$              ;BRANCH TO CHECK 600000+NOCARE IF SET
006220  132737  000002  002713          BITB    #BIT1,MEMGMT+1  ;SEE IF 400000+NOCARE WAS FOUND TO EXIST - IF NOT,
006226  001407                          BEQ     5$              ;GO SET EXPECTED ERROR AND NEX BITS AND CHECK FOR ERROR
006230  000404                          BR      4$              :BRANCH OUT IF LOCATION EXISTS
006232  132737  000004  002713  3$:     BITB    #BIT2,MEMGMT+1  ;SEE IF 600000+NOCARE WAS FOUND TO EXIST - IF NOT,
006240  001402                          BEQ     5$              ;GO SET EXPECTED ERROR AND NEX BITS AND CHECK FOR ERROR
006242  062716  000002          4$:     ADD     #2,(SP)         ;CORRECT PC RETURN
006246  000207                  5$:     RTS     PC              ;KICK OUT
```

```
                                        .SBTTL  BIT PATTERN
                                ;*******************************************************************
                                ;*
                                ;*      THIS IS A BIT PATTERN TABLE THAT CAN BE USED TO CHECK ANY LOCATION FOR
                                ;*      ALL COMBINATIONS OF STUCK AND/OR SHORTED BITS.
                                ;*
                                ;*******************************************************************

                                        .RADIX  2               ;THIS ENABLES YOU TO SEE THE PATTERNS IN BINARY
006250  177777          PATRNS: .WORD   1111111111111111  ;ALL SET BITS
006252  000000                  .WORD   0000000000000000  ;ALL CLEAR BITS
006254  052525                  .WORD   0101010101010101  ;EVEN BITS SET, ODD BITS CLEAR
006256  125252                  .WORD   1010101010101010  ;ODD BITS SET, EVEN BITS CLEAR
006260  031463                  .WORD   0011001100110011  ;PAIRS OF BITS SET
006262  007417                  .WORD   0000111100001111  ;GROUPS OF 4 BITS SET
006264  000377                  .WORD   0000000011111111  ;UPPER BYTE CLEAR, LOWER BYTE SET
000010                          .RADIX  8               ;THIS RETURNS MODE BACK TO OCTAL
```

```
                                .SBTTL  EXPECTED DATA TABLE FOR CSR CHECK TEST 30
                        ;*****************************************************************
                        ;*
                        ;*      THE 'EXPAT' TABLE IS USED TO CHECK THE CONTENTS OF THE CSR AFTER SETTING
                        ;*      THE BITS IN THE CSR.
                        ;*
                        ;*****************************************************************

006266                  MAICLR: ; X=     0    1    2    3    4    5    6    7
006266  000200  000000  001202          .WORD   0200,0000,1202,1002,122204,122204,123206,123206  ;CSR 00000X EXPECTED
006306  004210  004010  005212          .WORD   4210,4010,5212,5012,126214,126214,127216,127216  ;CSR 00001X EXPECTED
006326  000220  000020  001222          .WORD   0220,0020,1222,1022,122224,122224,123226,123226  ;CSR 00002X EXPECTED
006346  004230  004030  005232          .WORD   4230,4030,5232,5032,126234,126234,127236,127236  ;CSR 00003X EXPECTED
006366  000240  000040  001242          .WORD   0240,0040,1242,1042,122244,122244,123246,123246  ;CSR 00004X EXPECTED
006406  004250  004050  005252          .WORD   4250,4050,5252,5052,126254,126254,127256,127256  ;CSR 00005X EXPECTED
006426  000260  000060  001262          .WORD   0260,0060,1262,1062,122264,122264,123266,123266  ;CSR 00006X EXPECTED
006446  004270  004070  005272          .WORD   4270,4070,5272,5072,126274,126274,127276,127276  ;CSR 00007X EXPECTED
006466  000300  000100  001302          .WORD   0300,0100,1302,1102,122304,022104,123306,023106  ;CSR 00010X EXPECTED
006506  004310  004110  005312          .WORD   4310,4110,5312,5112,126314,026114,127316,027116  ;CSR 00011X EXPECTED
006526  000320  000120  001322          .WORD   0320,0120,1322,1122,122324,022124,123326,023126  ;CSR 00012X EXPECTED
006546  004330  004130  005332          .WORD   4330,4130,5332,5132,126334,026134,127336,027136  ;CSR 00013X EXPECTED
006566  000340  000140  001342          .WORD   0340,0140,1342,1142,122344,022144,123346,023146  ;CSR 00014X EXPECTED
006606  004350  004150  005352          .WORD   4350,4150,5352,5152,126354,026154,127356,027156  ;CSR 00015X EXPECTED
006626  000360  000160  001362          .WORD   0360,0160,1362,1162,122364,022164,123366,023166  ;CSR 00016X EXPECTED
006646  004370  004170  005372          .WORD   4370,4170,5372,5172,126374,026174,127376,027176  ;CSR 00017X EXPECTED
006666  000600  000200  001602          .WORD   0600,0200,1602,1202,122604,122604,123606,123606  ;CSR 00040X EXPECTED
006706  004610  004210  005612          .WORD   4610,4210,5612,5212,126614,126614,127616,127616  ;CSR 00041X EXPECTED
006726  000620  000220  001622          .WORD   0620,0220,1622,1222,122624,122624,123626,123626  ;CSR 00042X EXPECTED
006746  004630  004230  005632          .WORD   4630,4230,5632,5232,126634,126634,127636,127636  ;CSR 00043X EXPECTED
006766  000640  000240  001642          .WORD   0640,0240,1642,1242,122644,122644,123646,123646  ;CSR 00044X EXPECTED
007006  004650  004250  005652          .WORD   4650,4250,5652,5252,126654,126654,127656,127656  ;CSR 00045X EXPECTED
007026  000660  000260  001662          .WORD   0660,0260,1662,1262,122664,122664,123666,123666  ;CSR 00046X EXPECTED
007046  004670  004270  005672          .WORD   4670,4270,5672,5272,126674,126674,127676,127676  ;CSR 00047X EXPECTED
007066  000700  000300  001702          .WORD   0700,0300,1702,1302,122704,022304,123706,023306  ;CSR 00050X EXPECTED
007106  004710  004310  005712          .WORD   4710,4310,5712,5312,126714,026314,127716,027316  ;CSR 00051X EXPECTED
007126  000720  000320  001722          .WORD   0720,0320,1722,1322,122724,022324,123726,023326  ;CSR 00052X EXPECTED
007146  004730  004330  005732          .WORD   4730,4330,5732,5332,126734,026334,127736,027336  ;CSR 00053X EXPECTED
007166  000740  000340  001742          .WORD   0740,0340,1742,1342,122744,022344,123746,023346  ;CSR 00054X EXPECTED
007206  004750  004350  005752          .WORD   4750,4350,5752,5352,126754,026354,127756,027356  ;CSR 00055X EXPECTED
007226  000760  000360  001762          .WORD   0760,0360,1762,1362,122764,022364,123766,023366  ;CSR 00056X EXPECTED
007246  004770  004370  005772          .WORD   4770,4370,5772,5372,126774,026374,127776,027376  ;CSR 00057X EXPECTED
```

```
                                        .SBTTL  EXPECTED DATA TABLE FOR CSR CHECK TEST 3

                                        ; X=      0      1      2      3      4      5      6      7
007266  010200  010000  011202  MAISET: .WORD   10200,10000,11202,11002,12204,12004,13206,13006  ;CSR 01000X EXPECTED
007306  014210  014010  015212          .WORD   14210,14010,15212,15012,16214,16014,17216,17016  ;CSR 01001X EXPECTED
007326  010220  010020  011222          .WORD   10220,10020,11222,11022,12224,12024,13226,13026  ;CSR 01002X EXPECTED
007346  014230  014030  015232          .WORD   14230,14030,15232,15032,16234,16034,17236,17036  ;CSR 01003X EXPECTED
007366  010240  010040  011242          .WORD   10240,10040,11242,11042,12244,12044,13246,13046  ;CSR 01004X EXPECTED
007406  014250  014050  015252          .WORD   14250,14050,15252,15052,16254,16054,17256,17056  ;CSR 01005X EXPECTED
007426  010260  010060  011262          .WORD   10260,10060,11262,11062,12264,12064,13266,13066  ;CSR 01006X EXPECTED
007446  014270  014070  015272          .WORD   14270,14070,15272,15072,16274,16074,17276,17076  ;CSR 01007X EXPECTED
007466  010300  010100  011302          .WORD   10300,10100,11302,11102,12304,12104,13306,13106  ;CSR 01010X EXPECTED
007506  014310  014110  015312          .WORD   14310,14110,15312,15112,16314,16114,17316,17116  ;CSR 01011X EXPECTED
007526  010320  010120  011322          .WORD   10320,10120,11322,11122,12324,12124,13326,13126  ;CSR 01012X EXPECTED
007546  014330  014130  015332          .WORD   14330,14130,15332,15132,16334,16134,17336,17136  ;CSR 01013X EXPECTED
007566  010340  010140  011342          .WORD   10340,10140,11342,11142,12344,12144,13346,13146  ;CSR 01014X EXPECTED
007606  014350  014150  015352          .WORD   14350,14150,15352,15152,16354,16154,17356,17156  ;CSR 01015X EXPECTED
007626  010360  010160  011362          .WORD   10360,10160,11362,11162,12364,12164,13366,13166  ;CSR 01016X EXPECTED
007646  014370  014170  015372          .WORD   14370,14170,15372,15172,16374,16174,17376,17176  ;CSR 01017X EXPECTED
007666  010600  011202  011602          .WORD   10600,11202,11602,12204,12604,13206,13606,14210  ;CSR 01040X EXPECTED
007706  014610  015212  015612          .WORD   14610,15212,15612,16214,16614,17216,17616,10200  ;CSR 01041X EXPECTED
007726  010620  011222  011622          .WORD   10620,11222,11622,12224,12624,13226,13626,14230  ;CSR 01042X EXPECTED
007746  014630  015232  015632          .WORD   14630,15232,15632,16234,16634,17236,17636,10220  ;CSR 01043X EXPECTED
007766  010640  011242  011642          .WORD   10640,11242,11642,12244,12644,13246,13646,14250  ;CSR 01044X EXPECTED
010006  014650  015252  015652          .WORD   14650,15252,15652,16254,16654,17256,17656,10240  ;CSR 01045X EXPECTED
010026  010660  011262  011662          .WORD   10660,11262,11662,12264,12664,13266,13666,14270  ;CSR 01046X EXPECTED
010046  014670  015272  015672          .WORD   14670,15272,15672,16274,16674,17276,17676,10260  ;CSR 01047X EXPECTED
010066  010700  011302  011702          .WORD   10700,11302,11702,12304,12704,13306,13706,14310  ;CSR 01050X EXPECTED
010106  014710  015312  015712          .WORD   14710,15312,15712,16314,16714,17316,17716,10300  ;CSR 01051X EXPECTED
010126  010720  011322  011722          .WORD   10720,11322,11722,12324,12724,13326,13726,14330  ;CSR 01052X EXPECTED
010146  014730  015332  015732          .WORD   14730,15332,15732,16334,16734,17336,17736,10320  ;CSR 01053X EXPECTED
010166  010740  011342  011742          .WORD   10740,11342,11742,12344,12744,13346,13746,14350  ;CSR 01054X EXPECTED
010206  014750  015352  015752          .WORD   14750,15352,15752,16354,16754,17356,17756,10340  ;CSR 01055X EXPECTED
010226  010760  011362  011762          .WORD   10760,11362,11762,12364,12764,13366,13766,14370  ;CSR 01056X EXPECTED
010246  014770  015372  015772          .WORD   14770,15372,15772,16374,16774,17376,17776,10360  ;CSR 01057X EXPECTED
```

```
                                        .SBTTL  MAIN PROGRAM - INITIALIZATION ROUTINES
                                ;*****************************************************************
                                ;
                                ;MAIN PROGRAM - INITIALIZATION ROUTINES
                                ;
                                ;*****************************************************************

010266  005037  002670          START1: CLR     MANSIZE         ;CLEAR THE MANSIZE SO WE WILL AUTOSIZE
010272  005037  001416          START:  CLR     $PASS           ;CLEAR $PASS
010276  005037  002716                  CLR     $PASS2          ;CLEAR $PASS2
010302  005037  001412                  CLR     $FATAL          ;CLEAR ERROR NO.
010306  005037  001410                  CLR     $MSGTYP         ;CLEAR MESSAGE TYPE
010312  005037  001414                  CLR     $TESTN          ;CLEAR TEST NO.
010316  005037  001420                  CLR     $DEVCT          ;CLEAR DEVICE COUNT
010322  005037  001422                  CLR     $UNIT           ;CLEAR UNIT NUMBER
010326  012737  000006  000004          MOV     #TOVECT+2,TOVECT;INITIALIZE TIMEOUT VECTORS TO 6
010334  012737  000003  000006          MOV     #BPT,TMOPSW     ;CATCHER ROUTINE
010342  012706  001300                  MOV     #STACK,SP       ;INITIALIZE THE STACK
010346  004737  005660                  JSR     PC,SETTUP       ;GO TO THE SETUP ROUTINE TO INITIALIZE VECTORS
010352  005037  001422                  CLR     $UNIT           ;CLEAR $UNIT
010356  005037  001420                  CLR     $DEVCT          ;CLEAR $DEVCT
010362  005037  001414                  CLR     $TESTN          ;CLEAR $TESTN
010366  005037  002706                  CLR     EOPLOC          ;CLEAR EOPLOC
010372  132737  000001  001430          BITB    #BIT0,$ENV      ;CHECK IF ON APT
010400  001404                          BEQ     1$              ;BR IF NOT APT
010402  132737  000200  001431          BITB    #BIT7,$ENVM     ;DID APT SIZE
010410  001007                          BNE     2$              ;BR, IF APT SIZED
010412  022737  177777  002670  1$:     CMP     #-1,MANSIZE     ;WAS CONFIGURATION SET UP IN MULT. BOARD ROUTINE?
010420  001422                          BEQ     BEGIN           ;IF YES, SKIP SELF-SIZING
010422  004737  005274                  JSR     PC,ASIZE        ;AUTOMATICALLY SIZE FOR BOARD CONFIGURATION
010426  000417                          BR      BEGIN           ;BRANCH
010430  005037  002414          2$:     CLR     QTYBRD          ;CLEAR  DEVICE CNT
010434  013702  001466                  MOV     $DEVM,R2        ;MOVE DEVICE MAP TO R2
010440  005702                  3$:     TST     R2              ;TEST MSB OF DEVICE MAP
010442  100002                          BPL     4$              ;BR, IF MSB IS ZERO
010444  005237  002414                  INC     QTYBRD          ;INCREMENT DEVICE COUNT, IF MSB=1
010450  000241                  4$:     CLC                     ;CLEAR THE CARRY BIT FOR THE ROL
010452  006102                          ROL     R2              ;SHIFT NEXT BIT INTO MSB POSITION
010454  001371                          BNE     3$              ;CONTINUE CHECKING $DEVM, IF MORE BITS SET
010456  004737  004344                  JSR     PC,DEVADS       ;GENERATE DEVICE ADDRESS TABLE
010462  004737  005502                  JSR     PC,VCTADS       ;GENERATE VECTOR ADDRESS TABLE
```

```
                                        .SBTTL  DETERMINE MEM MGMT AND UPPER MEMORY EXISTENCE
010466  005737  001416          BEGIN:  TST     $PASS           ;SEE IF THIS IS THE FIRST PASS
010472  001145                          BNE     BEGIN1          ;BRANCH IF NOT
010474  005737  002716                  TST     $PASS2          ;SEE IF UPPER LOCATION HAS BEEN SET
010500  001142                          BNE     BEGIN1          ;BRANCH IF NOT
010502  005037  002712                  CLR     MEMGMT          ;CLEAR THE MEMORY MANAGEMENT FLAG
010506  013737  000004  002672          MOV     TOVECT,OLDPC1   ;SAVE TIMEOUT VECTOR
010514  012737  010750  000004          MOV     #3$,TOVECT      ;TIMEOUT VECTOR TO 3$
010522  013737  000006  002674          MOV     TMOPSW,OLDPS1   ;SAVE TIMEOUT PS
010530  012737  000340  000006          MOV     #LEVEL7,TMOPSW  ;PS TIMEOUT TO PRIORITY 7
010536  005737  177572                  TST     MMR0            ;TEST FOR THE PRESENCE OF MEMORY MANAGEMENT
010542  105237  002712                  INCB    MEMGMT          ;INCREMENT FLAG SHOWING MEMORY MANAGEMENT EXISTS
010546  012737  077406  172304          MOV     #77406,KIPDR2   ;MAKE KIPDR2 RESIDENT
010554  012737  077406  172324          MOV     #77406,KDPDR2   ;MAKE KDPDR2 RESIDENT
010562  013737  000250  002676          MOV     MMVECT,OLDPC2   ;SAVE MEMORY MANAGEMENT VECTOR
010570  012737  010722  000250          MOV     #1$,MMVECT      ;MEMORY MANAGEMENT VECTOR TO 1$
010576  013737  000252  002700          MOV     MMPS,OLDPS2     ;SAVE MEMORY MANAGEMENT PS
010604  012737  000340  000252          MOV     #LEVEL7,MMPS    ;MEMORY MANAGEMENT PS TO PRIORITY 7
010612  005237  177572                  INC     MMR0            ;TURN ON MEMORY MANAGEMENT
010616  012737  002400  172344          MOV     #2400,KIPAR2    ;SET UP KIPAR2 TO ACCESS LOCATION 240000+BITS 12-0 OF NOCARE
010624  012737  002400  172364          MOV     #2400,KDPAR2    ;SET UP KDPAR2 TO ACCESS LOCATION 240000+BITS 12-0 OF NOCARE
010632  005737  053700                  TST     NOCARE          ;SEE IF BITS 12-0 OF NOCARE ADRS +240000 EXISTS
010636  152737  000001  002713          BISB    #BIT0,MEMGMT+1  ;SET BIT 0 OF UPPER BYTE OF MEMGMT IF IT DOES
010644  012737  004400  172344          MOV     #4400,KIPAR2    ;SET UP KIPAR2 TO ACCESS LOCATION 440000+BITS 12-0 OF NOCARE
010652  012737  004400  172364          MOV     #4400,KDPAR2    ;SET UP KDPAR2 TO ACCESS LOCATION 440000+BITS 12-0 OF NOCARE
010660  005737  053700                  TST     NOCARE          ;SEE IF BITS 12-0 OF NOCARE ADRS +440000 EXISTS
010664  152737  000002  002713          BISB    #BIT1,MEMGMT+1  ;SET BIT 1 OF UPPER BYTE OF MEMGMT IF IT DOES
010672  012737  006400  172344          MOV     #6400,KIPAR2    ;SET UP KIPAR2 TO ACCESS LOCATION 640000+BITS 12-0 OF NOCARE
010700  012737  006400  172364          MOV     #6400,KDPAR2    ;SET UP KDPAR2 TO ACCESS LOCATION 640000+BITS 12-0 OF NOCARE
010706  005737  053700                  TST     NOCARE          ;SEE IF BITS 12-0 OF NOCARE ADRS +640000 EXISTS
010712  152737  000004  002713          BISB    #BIT2,MEMGMT+1  ;SET BIT 2 OF UPPER BYTE OF MEMGMT IF IT DOES
010720  000402                          BR      2$              ;BRANCH OVER STACK CORRECTION
010722  062706  000004          1$:     ADD     #4,SP           ;CORRECT STACK AFTER MM TRAP
010726  005037  177572          2$:     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
010732  013737  002700  000252          MOV     OLDPS2,MMPS     ;RESTORE MEMORY MANAGEMENT PS
010740  013737  002676  000250          MOV     OLDPC2,MMVECT   ;RESTORE MEMORY MANAGEMENT VECTOR
010746  000402                          BR      4$              ;BRANCH OVER STACK CORRECTION
010750  062706  000004          3$:     ADD     #4,SP           ;CORRECT STACK AFTER TIMEOUT
010754  013737  002674  000006  4$:     MOV     OLDPS1,TMOPSW   ;RESTORE TIMEOUT PS
010762  013737  002672  000004          MOV     OLDPC1,TOVECT   ;RESTORE TIMEOUT VECTOR
010770  104401  037170                  TYPE    ,M1             ;TYPE:  '(^X) INHIBITS EOP'S, (^Y) FOR ERROR SUMMARY'
                                                                ;       'UNIBUS HANG?  RESTART AT ADDRESS '
010774  012746  053622                  MOV     #UBHANG,-(SP)   ;MOVE ADDRESS OF HANG ROUTINE TO STACK
011000  104402                          TYPOC                   ;GO TYPE THE ADDRESS IN OCTAL
011002  104401  037307                  TYPE    ,M1A            ;       'LZDRLCO DR11 GEN NPR INTFC LOGIC TEST'
```

```
                                                .SBTTL  PREPARE ADDRESSES AND VECTORS FOR UUT
011006  012737  000001  002544  BEGIN1: MOV     #BIT0,DEVMSK    ;SET UP BIT MASK TO TEST $DEVM FOR DEVICES
011014  005037  002546                 CLR     TABINX          ;CLEAR LOCATION TO STORE TABLE OFFSETS
011020  033737  002544  001466  TSTDEV: BIT     DEVMSK,$DEVM    ;CHECK TO SEE IF DEVICE IS TO BE TESTED
011026  001026                         BNE     2$              ;BR, IF YES
011030  005737  002544                 TST     DEVMSK          ;SEE IF BIT 15 IS SET
011034  100013                         BPL     1$              ;BRANCH TO CONTINUE IF NOT SET
011036  005737  001416                 TST     $PASS           ;SEE IF THIS IS THE FIRST PASS
011042  001361                         BNE     BEGIN1          ;BRANCH TO REINITIALIZE THE DEVMSK LOCATION IF NOT
011044  005737  002716                 TST     $PASS2          ;SEE IF THIS IS THE FIRST PASS
011050  001356                         BNE     BEGIN1          ;BRANCH TO REINITIALIZE THE DEVMSK LOCATION IF NOT
011052  104401  037037                 TYPE    ,NODVPR         ;TYPE:  'NO DEVICES RECOGNIZED - DIAGNOSTIC CANNOT BE RUN'
                                                                ;       'RESTART AT 204 IF A DEVICE IS PRESENT'
011056  000000                         HALT                    ;FATAL ERROR - HALT HERE
011060  000137  010266                 JMP     START1          ;JUMP TO START1 TO CHECK AGAIN FOR A MODULE
011064  006337  002544          1$:    ASL     DEVMSK          ;SHIFT MASK TO CHECK NEXT $DEVM BIT
011070  062737  000002  002546         ADD     #2,TABINX       ;INCREMENT TABLE INDEX
011076  005237  001422                 INC     $UNIT           ;INCREMENT UNIT NUMBER
011102  000746                         BR      TSTDEV          ;GO TEST NEXT BIT OF DEVICE MAP

011104  006337  002544          2$:    ASL     DEVMSK          ;UPDATE DEVICE MAP TEST MASK
011110  013702  002546                 MOV     TABINX,R2       ;MOVE TABLE OFFSET TO R2
011114  062737  000002  002546         ADD     #2,TABINX       ;UPDATE TABLE OFFSET FOR NEXT DEVICE
011122  016200  002416                 MOV     REGADR(R2),R0   ;PUT UUT ADDRESS INTO R0
011126  012701  002516                 MOV     #WCR,R1         ;POINT R1 TO STORAGE AREA FOR UUT ADDRESSES
011132  012703  000004                 MOV     #4,R3           ;MOVE 4 ADDRESSES
011136  010021          3$:    MOV     R0,(R1)+        ;TRANSFER UUT ADDRESS
011140  062700  000002                 ADD     #2,R0           ;POINT TO NEXT UUT REGISTER
011144  005303                         DEC     R3              ;DECREMENT THE LOOP COUNTER AND
011146  001373                         BNE     3$              ;BRANCH IF NOT DONE TRANSFERING
011150  016200  002456                 MOV     VECADR(R2),R0   ;PUT UUT VECTOR INTO R0
011154  010021                         MOV     R0,(R1)+        ;TRANSFER UUT VECTORS TO ACTIVE TABLE AREA
011156  062700  000002                 ADD     #2,R0           ;POINT TO NEXT VECTOR
011162  010011                         MOV     R0,(R1)         ;TRANSFER VECTOR TO TABLE AREA
011164  016237  001474  002720         MOV     $DDW0(R2),DDW   ;SET UP DDW TO PROPER DEVICE DESCRIPTOR WORD
011172  013737  002720  002552         MOV     DDW,DRLEV       ;MOVE THE WORD TO THE DRLEV LOCATION
011200  042737  177437  002552         BIC     #177437,DRLEV   ;STRIP ALL BITS EXCEPT BR LEVEL
011206  105037  030707          REINIT: CLRB    CHARCT          ;CLEAR THE CHARACTER LOCATION OF ANY CHARACTER
011212  004737  004374                 JSR     PC,BPINIT       ;GO RESET THE ".+2" AND "BPT" LOCATIONS
011216  004737  004036                 JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
011222  105737  002706                 TSTB    EOPLOC          ;SEE IF ^X IS ENABLED (IS THE PRINTER DISABLED)
011226  001006                         BNE     TST1            ;GO DO TEST IF NOT
                                ;*****************************************************************
                                ;*      *DO*NOT*REMOVE*THE*WAIT*LOOP*ROUTINE*BELOW*.  BECAUSE OF THE SPEED OF THIS DIAG-
                                ;*      NOSTIC (.125 SECOND FOR 1 PASS, NO ERRORS), SOME VIDEO TERMINALS PRINT ERRONEOUS
                                ;*      CHARACTER(S) WITH THE EOP MESSAGE DUE TO THE RESET EXECUTED IN TEST 4.  THIS WAIT
                                ;*      LOOP ENABLES THOSE TERMINALS TO 'CATCH UP' BEFORE ITS EXECUTION.
                                ;*****************************************************************
011230  012737  010000  011240         MOV     #10000,2$       ;REESTABLISH THE WAIT LOOP COUNTER
011236  005327          1$:    DEC     (PC)+           ;DECREMENT THE LOCATION TO KILL TIME
011240  010000          2$:    .WORD   10000           ;LOCATION TO BE DECREMENTED
011242  001375                 BNE     1$              ;BRANCH BACK UNTIL ZERO
                                ;*****************************************************************
                                ;
                                ;MAIN PROGRAM - DEVICE TESTS
                                ;
                                ;*****************************************************************
```

```
                        .SBTTL  TEST #1 - CAN ALL DR11 REG BE ADDRESSED WITHOUT ERROR?
                        ;*********************************************************************
                        ;*TEST 1        CAN ALL DR11 REG BE ADDRESSED WITHOUT ERROR?
                        ;*
                        ;*      THIS TEST INSURES THAT THE CSR, BAR, BDR AND WCR REGISTERS CAN BE
                        ;*      ACCESSED FOR THIS DEVICE.  IF NOT, THE REST OF THE DIAGNOSTIC CANNOT
                        ;*      BE RUN.
                        ;*
                        ;*********************************************************************
011244  032777  004000  170066  TST1:   BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
011252  001407                          BEQ     1001$           ;&& BRANCH IF NOT
011254  104401  011262                  TYPE    ,1000$          ;&& TYPE: "T # 1"
011260  000404                          BR      1001$           ;&& GET OVER ASCII
011262     124     040     043  1000$:  .ASCIZ  /T # 1/<CRLF>   ;&& THE ASCII MESSAGE
                                        .EVEN
011272                          1001$:
011272  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
011274  012737  011302  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2478 011302  005037  001312  999$:     CLR     $ERTTL          ;CLEAR THE ERROR TOTAL - NEW PASS
2479 011306  012737  011410  000004    MOV     #1$,BUSERR      ;CPU ERROR VECTOR TO 1$
2480 011314  013737  002516  002550    MOV     WCR,DREG        ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
2481 011322  005777  171170            TST     @WCR            ;ACCESS REGISTER
2482 011326  013737  002520  002550    MOV     BAR,DREG        ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
2483 011334  005777  171160            TST     @BAR            ;ACCESS REGISTER
2484 011340  013737  002522  002550    MOV     CSR,DREG        ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
2485 011346  005777  171150            TST     @CSR            ;ACCESS REGISTER
2486 011352  013737  002524  002550    MOV     BDR,DREG        ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
2487 011360  005777  171140            TST     @BDR            ;ACCESS REGISTER
2488 011364  013737  002526  002550    MOV     DRINV,DREG      ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
2489 011372  005777  171130            TST     @DRINV          ;ACCESS REGISTER
2490 011376  005737  001312            TST     $ERTTL          ;SEE IF THERE WERE ANY ERRORS
2491 011402  001414                    BEQ     2$              ;BRANCH TO CONTINUE IF NONE
2492 011404  000137  024360            JMP     ENDEV           ;GO TO END OF DEVICE ROUTINE - FATAL ERRORS
2493 011410  012637  002672    1$:     MOV     (SP)+,OLDPC1    ;SAVE PC OF TRAP FOR ERROR PRINTOUT
2494 011414  012637  002674            MOV     (SP)+,OLDPS1    ;SAVE PS FOR RESTORATION AFTER ERROR CALL
2495 011420  104001                    ERROR   +1              ;CANNOT ACCESS DR11 REGISTER
2496 011422  013746  002674            MOV     OLDPS1,-(SP)    ;PUT PS BACK ON STACK
2497 011426  013746  002672            MOV     OLDPC1,-(SP)    ;PUT PC BACK ON STACK
2498 011432  000002                    RTI                     ;RETURN TO PROGRAM
2499 011434  012737  000006  000004  2$:     MOV     #6,BUSERR       ;RESTORE #6 TO BUS ERROR
```

```
                                        .SBTTL  TEST #2 - CHECK B OR W STATUS IS AS EXPECTED
                                        ;*********************************************************************
                                        ;*TEST 2        CHECK B OR W STATUS IS AS EXPECTED
                                        ;*
                                        ;*      THIS TEST INSURES THAT THE B OR W STATUS IN THE DEVICE DESCRIPTOR
                                        ;*      WORD MATCHES WHAT THE EIR SAYS THE MODULE IS.
                                        ;*
                                        ;*********************************************************************
011442  032777  004000  167670  TST2:   BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
011450  001407                          BEQ     1001$           ;&& BRANCH IF NOT
011452  104401  011460                  TYPE    ,1000$          ;&& TYPE: "T # 2"
011456  000404                          BR      1001$           ;&& GET OVER ASCII
011460     124     040     043  1000$:  .ASCIZ  /T # 2/<CRLF>   ;&& THE ASCII MESSAGE
                                        .EVEN
011470                               1001$:
011470  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
011472  012737  011500  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2507 011500  005077  171016       999$: CLR     @CSR            ;GO TO CSR
2508 011504  012777  100000  171010    MOV     #EIR,@CSR       ;FORCE TO BE EIR
2509 011512  017737  171004  002610    MOV     @CSR,BORW       ;ATTEMPT EIR READ
2510 011520  042737  177776  002610    BIC     #CBIT0,BORW     ;MASK OFF ALL BITS EXCEPT BIT 0
2511 011526  013737  002720  001362    MOV     DDW,$TMP1       ;GET DEVICE DESCRIPTOR WORD
2512 011534  042737  177776  001362    BIC     #CBIT0,$TMP1    ;MASK OFF ALL BUT B OR W BIT
2513 011542  001403                    BEQ     1$              ;BRANCH IF IT IS CLEAR
2514 011544  005037  001362            CLR     $TMP1           ;CLEAR THE BIT
2515 011550  000403                    BR      2$              ;GO TEST THE BIT
2516 011552  012737  000001  001362 1$:  MOV     #1,$TMP1        ;SET THE BIT
2517 011560  023737  002610  001362 2$:  CMP     BORW,$TMP1      ;B OR W STATE AS EXPECTED?
2518 011566  001401                    BEQ     TST3            ;BRANCH IF OK
2519 011570  104002                    ERROR   +2              ;DR11-B OR W MODE INCORRECT (0=B, 1-W)
```

```
2527                                    .SBTTL  TEST #3 - CHECK CSR BIT PATTERNS WITH MAINT BIT SET
                                        ;**********************************************************************
                                        ;*TEST 3        CHECK CSR BIT PATTERNS WITH MAINT BIT SET
                                        ;*
                                        ;*      THIS TEST SETS ALL POSSIBLE COMBINATIONS OF SET BITS IN THE CSR WITH
                                        ;*      THE MAINTENANCE BIT SET, AND COMPARES THE RECEIVED CSR CONTENTS WITH
                                        ;*      THAT OF THE EXPECTED PATTERNS IN THE 'MAISET' TABLE.
                                        ;*
                                        ;**********************************************************************
011572  032777  004000  167540  TST3:   BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
011600  001407                          BEQ     1001$           ;&& BRANCH IF NOT
011602  104401  011610                  TYPE    ,1000$          ;&& TYPE: "T # 3"
011606  000404                          BR      1001$           ;&& GET OVER ASCII
011610     124     040     043  1000$:  .ASCIZ  /T # 3/<CRLF>   ;&& THE ASCII MESSAGE
                                        .EVEN
011620                               1001$:
011620  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
011622  012737  011704  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2528 011630  004737  004036            JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2529 011634  012737  000200  001362    MOV     #RY,$TMP1       ;MOVE READY BIT TO $TMP1
2530 011642  032737  000004  002720    BIT     #BIT2,DDW       ;TEST TO SEE IF CABLE IS IN
2531 011650  001003                    BNE     1$              ;BRANCH AROUND NON-CABLE SETUP IF IN
2532 011652  052737  127000  001362    BIS     #127000,$TMP1   ;SET THE BITS TO BE EXPECTED IN $TMP1
2533 011660  012701  007266          1$: MOV     #MAISET,R1      ;MOVE ADDRESS OF EXPECTED PATTERNS TO R1
2534 011664  012702  010000            MOV     #MA,R2          ;START WITH JUST THE MAINTENANCE BIT
2535 011670  005037  002714            CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +202
2536 011674  012700  000002            MOV     #2,R0           ;DO 2 SETS OF 200 PATTERNS
2537 011700  012703  000200          2$: MOV     #200,R3         ;MOVE 200 TO THE LOOP COUNTER
2538 011704  052777  010000  170610  999$: BIS     #MA,@CSR        ;SET MAINTENANCE AND
2539 011712  005077  170604            CLR     @CSR            ;CLEAR TO DO AN INIT
2540 011716  017737  170600  002560    MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2541 011724  023737  001362  002560    CMP     $TMP1,RCSR      ;MAKE SURE EXPECTED DATA CAME UP
2542 011732  001404                    BEQ     3$              ;BRANCH IF SO
2543 011734  013737  001362  002572    MOV     $TMP1,ECSR      ;MOVE EXPECTED DATA TO ECSR
2544 011742  104032                    ERROR   +32             ;CSR IS WRONG
2545 011744  012777  177777  170544  3$: MOV     #-1,@WCR        ;MOVE 1 WORD COUNT TO WCR IN CASE OF IE ENABLED
2546 011752  012777  053700  170540    MOV     #NOCARE,@BAR    ;MOVE A NOT-CARE ADDRESS TO BAR FOR SAME REASON
2547 011760  010277  170536            MOV     R2,@CSR         ;SET THE PARTICULAR FUNCTION BITS IN CSR
2548 011764  017737  170532  002560    MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2549 011772  011137  002572            MOV     (R1),ECSR       ;MOVE EXPECTED DATA TO ECSR
2550 011776  023737  002572  002560    CMP     ECSR,RCSR       ;COMPARE EXPECTED WITH RECEIVED
2551 012004  001425                    BEQ     7$              ;BRANCH IF OK
2552 012006  012737  000401  012016    MOV     #CY+GO,4$       ;REESTABLISH TEST PATTERN
2553 012014  040227                    BIC     R2,(PC)+        ;SEE IF BOTH CYCLE AND GO WERE SET
2554 012016  000401                  4$: .WORD   CY+GO           ;LOCATION TO HOLD BOTH CYCLE AND GO BITS
2555 012020  001013                    BNE     6$              ;BRANCH TO ERROR ONLY IF CYCLE AND GO WERE SET
2556 012022  032737  000060  002572    BIT     #X6+X7,ECSR     ;SEE IF EITHER XBA16 OR XBA17 ARE SET
2557 012030  001407                    BEQ     6$              ;BRANCH TO ERROR IF BOTH ARE CLEAR
2558 012032  052737  140000  002572  5$: BIS     #ER+NX,ECSR     ;SET THE ERROR AND NEX BITS - EXPECT THEM TO SET
2559 012040  023737  002560  002572    CMP     RCSR,ECSR       ;NOW SEE IF DATA MATCHES
2560 012046  001415                    BEQ     10$             ;BRANCH AROUND ERROR IF IT DOES
2561 012050  010237  002536          6$: MOV     R2,BUT          ;MOVE THE BITS SET INTO CSR TO THE BUT LOCATION
2562 012054  104202                    ERROR   +202            ;CSR PATTERN NOT CORRECT
2563 012056  000411                    BR      10$             ;BRANCH AROUND MM TESTS
2564 012060  012737  000401  012070  7$: MOV     #CY+GO,8$       ;REESTABLISH TEST PATTERN
2565 012066  040227                    BIC     R2,(PC)+        ;SEE IF BOTH CYCLE AND GO WERE SET
2566 012070  000401                  8$: .WORD   CY+GO           ;LOCATION TO HOLD BOTH CYCLE AND GO BITS
```

```
2567 012072  001003                      BNE     10$             ;BRANCH AROUND MEM MGMT TEST IF EITHER OR BOTH WERE CLEAR
2568 012074  004737  006134      9$:     JSR     PC,TSTMM        ;GO CHECK FOR MEMORY MANAGEMENT EXISTENCE
2569 012100  000754                      BR      5$              ;IF RETURN IS HERE, GO BACK TO SET EXPECTED DATA
2570 012102  062701  000002      10$:    ADD     #2,R1           ;INCREMENT R1 TO NEXT EXPECTED PATTERN
2571 012106  005202                      INC     R2              ;INCREMENT THE PATTERN
2572 012110  005303                      DEC     R3              ;DECREMENT THE LOOP COUNTER
2573 012112  001274                      BNE     999$            ;BRANCH BACK IF NOT DONE
2574 012114  062702  000200              ADD     #200,R2         ;ADD 200 TO PATTERN LOCATION
2575 012120  005300                      DEC     R0              ;DECREMENT THE LOOP COUNTER AND
2576 012122  001266                      BNE     2$              ;BRANCH BACK IF 2ND OCTAL GROUP NOT DONE
```

```
                                .SBTTL  TEST #4 - CHECK WCR, BAR & BDR, & RESET CLRS 4 DEV REGS
                                ;*********************************************************************
                                ;*TEST 4         CHECK WCR, BAR & BDR, & RESET CLRS 4 DEV REGS
                                ;*
                                ;*      THIS TEST INSURES THAT THE WCR, BAR AND BDR REGISTER BITS CAN ALL BE
                                ;*      SET, AND THAT A RESET CLEARS ALL 3 PLUS THE CSR REGISTER.
                                ;*
                                ;*********************************************************************
012124  032777  004000  167206  TST4:   BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
012132  001407                        BEQ     1001$           ;&& BRANCH IF NOT
012134  104401  012142                TYPE    ,1000$          ;&& TYPE: "T # 4"
012140  000404                        BR      1001$           ;&& GET OVER ASCII
012142     124     040     043  1000$:  .ASCIZ  /T # 4/<CRLF>   ;&& THE ASCII MESSAGE
                                        .EVEN
012152                          1001$:
012152  000004                        SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
012154  012737  012162  001310        MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2584 012162  012777  010000  170332  999$:   MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
2585 012170  005077  170326            CLR     @CSR            ;CLEAR TO DO AN INIT
2586 012174  012777  010000  170320        MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT TO DO THIS TEST IN MAINT MODE
2587 012202  012777  177777  170306        MOV     #-1,@WCR        ;ALL ONES TO WCR
2588 012210  017737  170302  002570        MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
2589 012216  022737  177777  002570        CMP     #-1,RWCR        ;SEE IF DATA WAS LOADED PROPERLY
2590 012224  001423                    BEQ     4$              ;BRANCH IF OK
2591 012226  012737  012236  001310        MOV     #1$,$LPERR      ;MOVE NEW LOOP ON ERROR LOCATION TO $LPERR
2592 012234  000412                    BR      2$              ;BRANCH OVER LOOP SETUP
2593 012236  012777  177777  170252  1$:     MOV     #-1,@WCR        ;ALL ONES TO WCR
2594 012244  017737  170246  002570        MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
2595 012252  022737  177777  002570        CMP     #-1,RWCR        ;SEE IF DATA WAS LOADED PROPERLY
2596 012260  001401                    BEQ     3$              ;BRANCH IF OK
2597 012262  104010                2$:     ERROR   +10             ;ATTEMPT TO SET ALL WCR BITS FAILED
2598 012264  032777  001000  167046  3$:     BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
2599 012272  001361                    BNE     1$              ;BRANCH BACK IF SO
2600 012274  012777  177777  170216  4$:     MOV     #-1,@BAR        ;ALL ONES TO BAR
2601 012302  017737  170212  002566        MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
2602 012310  022737  177776  002566        CMP     #-2,RBAR        ;SEE IF ALL BITS WERE SET (DON'T EXPECT BIT 0 TO SET)
2603 012316  001426                    BEQ     8$              ;BRANCH IF OK
2604 012320  012737  012336  001310        MOV     #5$,$LPERR      ;MOVE NEW LOOP ON ERROR LOCATION TO $LPERR
2605 012326  012737  177776  002600        MOV     #-2,EBAR        ;MOVE EXPECTED DATA TO EBAR
2606 012334  000412                    BR      6$              ;BRANCH OVER LOOP SETUP
2607 012336  012777  177777  170154  5$:     MOV     #-1,@BAR        ;ALL ONES TO BAR
2608 012344  017737  170150  002566        MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
2609 012352  022737  177776  002566        CMP     #-2,RBAR        ;SEE IF ALL BITS WERE SET (DON'T EXPECT BIT 0 TO SET)
2610 012360  001401                    BEQ     7$              ;BRANCH IF OK
2611 012362  104012                6$:     ERROR   +12             ;ATTEMPT TO SET ALL BAR BITS TO 1 FAILED
2612 012364  032777  001000  166746  7$:     BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
2613 012372  001361                    BNE     5$              ;BRANCH BACK IF SO
2614 012374  052777  000001  170120  8$:     BIS     #GO,@CSR        ;SET GO TO CLEAR READY TO PREPARE TO SET BAR BIT 0
2615 012402  017737  170114  002560        MOV     @CSR,RCSR       ;READ CSR TO SET BIT 0 OF BAR
2616 012410  012777  177777  170102        MOV     #-1,@BAR        ;ALL ONES TO BAR
2617 012416  017737  170076  002566        MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
2618 012424  022737  177777  002566        CMP     #-1,RBAR        ;SEE IF ALL BITS WERE SET (*DO* EXPECT BIT 0 TO SET)
2619 012432  001431                    BEQ     12$             ;BRANCH IF OK
2620 012434  012737  012452  001310        MOV     #9$,$LPERR      ;MOVE NEW LOOP ON ERROR LOCATION TO $LPERR
2621 012442  012737  177777  002600        MOV     #-1,EBAR        ;MOVE EXPECTED DATA TO EBAR
2622 012450  000415                    BR      10$             ;BRANCH OVER LOOP SETUP
2623 012452  017737  170044  002560  9$:     MOV     @CSR,RCSR       ;ACCESS CSR TO SET BIT 0 OF BAR
```

```
2624 012460  012777  177777  170032          MOV     #-1,@BAR        ;ALL ONES TO BAR
2625 012466  017737  170026  002566          MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
2626 012474  022737  177777  002566          CMP     #-1,RBAR        ;SEE IF ALL BITS WERE SET (*DO* EXPECT BIT 0 TO SET)
2627 012502  001401                          BEQ     11$             ;BRANCH IF OK
2628 012504  104012                  10$:    ERROR   +12             ;ATTEMPT TO SET ALL BAR BITS TO 1 FAILED
2629 012506  032777  001000  166624  11$:    BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
2630 012514  001356                          BNE     9$              ;BRANCH BACK IF SO
2631 012516  012777  177777  170000  12$:    MOV     #-1,@BDR        ;ALL ONES TO BDR
2632 012524  017737  167774  002564          MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
2633 012532  022737  177777  002564          CMP     #-1,RBDR        ;SEE IF DATA WAS LOADED PROPERLY
2634 012540  001423                          BEQ     16$             ;BRANCH IF OK
2635 012542  012737  012552  001310          MOV     #13$,$LPERR     ;MOVE NEW LOOP ON ERROR LOCATION TO $LPERR
2636 012550  000412                          BR      14$             ;BRANCH OVER LOOP SETUP
2637 012552  012777  177777  167744  13$:    MOV     #-1,@BDR        ;ALL ONES TO BDR
2638 012560  017737  167740  002564          MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
2639 012566  022737  177777  002564          CMP     #-1,RBDR        ;SEE IF DATA WAS LOADED PROPERLY
2640 012574  001401                          BEQ     15$             ;BRANCH IF OK
2641 012576  104027                  14$:    ERROR   +27             ;ALL BDR BITS ARE NOT SET
2642 012600  032777  001000  166532  15$:    BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
2643 012606  001361                          BNE     13$             ;BRANCH BACK IF SO
2644 012610  012777  010576  167704  16$:    MOV     #10576,@CSR     ;SET ALL CSR WRITEABLE BITS
2645 012616  000005                          RESET                   ;RESET THE WORLD OF ITS TROUBLES - HOPEFULLY
2646 012620  012737  012162  001310          MOV     #999$,$LPERR    ;RESET THE LOOP ON ERROR LOCATION
2647 012626  017737  167664  002570          MOV     @WCR,RWCR       ;WAS WCR CLEARED?
2648 012634  001401                          BEQ     17$             ;BRANCH IF WCR WAS CLEARED
2649 012636  104007                          ERROR   +7              ;RESET FAILED TO CLEAR WCR
2650 012640  017737  167654  002566  17$:    MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
2651 012646  001403                          BEQ     18$             ;BRANCH IF BAR WAS CLEARED
2652 012650  005037  002600                  CLR     EBAR            ;CLEAR EXPECTED LOCATION
2653 012654  104011                          ERROR   +11             ;RESET FAILED TO CLEAR BAR
2654 012656  017737  167640  002560  18$:    MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2655 012664  012737  000200  002572          MOV     #RY,ECSR        ;MOVE EXPECTED DATA TO ECSR
2656 012672  004737  004060                  JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
2657 012676  023737  002572  002560          CMP     ECSR,RCSR       ;SEE IF EXPECTED DATA WAS RECEIVED
2658 012704  001401                          BEQ     19$             ;BRANCH IF IT WAS
2659 012706  104032                          ERROR   +32             ;READY IS NOT THE ONLY BIT SET
2660 012710  052777  010000  167604  19$:    BIS     #MA,@CSR        ;MAINT MODE (SO THAT IDR GETS ODR CONTENTS)
2661 012716  017737  167602  002564          MOV     @BDR,RBDR       ;MOVE CONTENTS OF BDR TO RBDR
2662 012724  001401                          BEQ     TST5            ;BRANCH IF IT CORRECTLY REMAINS ZERO
2663 012726  104026                          ERROR   +26             ;BDR IS NOT CLEAR
```

```
                                        .SBTTL  TEST #5 - DEVICE INIT CLEARS CSR, WCR, BDR AND BAR
                                        ;**********************************************************************
                                        ;*TEST 5        DEVICE INIT CLEARS CSR, WCR, BDR AND BAR
                                        ;*
                                        ;*      THIS TEST INSURES THAT DEVICE INIT CLEARS THE CSR, WCR, BDR AND BAR.
                                        ;*
                                        ;**********************************************************************
     012730  032777  004000  166402  TST5:   BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     012736  001407                          BEQ     1001$           ;&& BRANCH IF NOT
     012740  104401  012746                  TYPE    ,1000$          ;&& TYPE: "T # 5"
     012744  000404                          BR      1001$           ;&& GET OVER ASCII
     012746     124     040     043  1000$:  .ASCIZ  /T # 5/<CRLF>   ;&& THE ASCII MESSAGE
                                             .EVEN
     012756  000004                  1001$:  SCOPE
2670 012760  005077  167536          999$:   CLR     @CSR            ;FORCE ACCESS TO CSR
2671 012764  012777  177777  167524          MOV     #-1,@WCR        ;ALL ONES TO WCR
2672 012772  012777  177777  167524          MOV     #-1,@BDR        ;ALL ONES TO BDR
2673 013000  012777  177777  167512          MOV     #-1,@BAR        ;ALL ONES TO BAR
2674 013006  012777  010576  167506          MOV     #10576,@CSR     ;SET ALL WRITEABLE BITS IN THE CSR
2675 013014  042777  010000  167500          BIC     #MA,@CSR        ;CLEAR THE MAINT BIT TO DO AN INIT
2676 013022  017737  167470  002570          MOV     @WCR,RWCR       ;MOVE RECEIVED CONTENTS TO RWCR
2677 013030  001401                          BEQ     1$              ;BRANCH IF WCR WAS CLEARED
2678 013032  104003                          ERROR   +3              ;INIT FAILED TO CLEAR WCR
2679 013034  017737  167460  002566  1$:     MOV     @BAR,RBAR       ;MOVE RECEIVED CONTENTS TO RBAR
2680 013042  001403                          BEQ     2$              ;BRANCH IF BAR WAS CLEARED
2681 013044  005037  002600                  CLR     EBAR            ;CLEAR EXPECTED LOCATION
2682 013050  104004                          ERROR   +4              ;INIT FAILED TO CLEAR BAR
2683 013052  017737  167444  002560  2$:     MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2684 013060  012737  000200  002572          MOV     #RY,ECSR        ;EXPECT READY BIT ONLY TO BE SET
2685 013066  004737  004060                  JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
2686 013072  023737  002572  002560          CMP     ECSR,RCSR       ;SEE IF EXPECTED DATA WAS RECEIVED
2687 013100  001401                          BEQ     3$              ;BRANCH IF THEY WERE ALL CLEAR
2688 013102  104006                          ERROR   +6              ;INIT FAILED TO CLEAR ALL CSR R-W BITS
2689 013104  012777  010000  167410  3$:     MOV     #MA,@CSR        ;GO BACK INTO MAINT MODE (SO THAT IDR GETS ODR CONTENTS)
2690 013112  017737  167406  002564          MOV     @BDR,RBDR       ;MOVE RECEIVED CONTENTS TO RBDR
2691 013120  001401                          BEQ     TST6            ;BRANCH IF IT WAS CLEARED
2692 013122  104005                          ERROR   +5              ;INIT FAILED TO CLEAR BDR
```

```
2702                                    .SBTTL  TEST #6 - BIT PATTERN TEST OF WCR, BDR AND BAR REGISTERS
                                        ;**********************************************************************
                                        ;*TEST 6        BIT PATTERN TEST OF WCR, BDR AND BAR REGISTERS
                                        ;*
                                        ;*      THIS TEST RUNS 7 BIT PATTERNS THROUGH THE WCR, BDR AND BAR TO CHECK FOR
                                        ;*      ANY STUCK OR SHORTED PINS.  LOCATION $LPERR IS NOT SET UP AT THE START
                                        ;*      SINCE A DIFFERENT METHOD OF ERROR LOOPING IS DONE.  WHEN AN ERROR IS
                                        ;*      DETERMINED TO EXIST, THE $LPERR IS INITIALIZED TO A ROUTINE SPECIFI-
                                        ;*      CALLY WRITTEN FOR THAT PARTICULAR ERROR TO CREATE A VERY TIGHT LOOP.
                                        ;*
                                        ;**********************************************************************
     013124  032777  004000  166206 TST6:   BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     013132  001407                         BEQ     1001$           ;&& BRANCH IF NOT
     013134  104401  013142                 TYPE    ,1000$          ;&& TYPE: "T # 6"
     013140  000404                         BR      1001$           ;&& GET OVER ASCII
     013142     124     040     043 1000$:  .ASCIZ   /T # 6/<CRLF>   ;&& THE ASCII MESSAGE
                                            .EVEN
     013152  000004                 1001$:  SCOPE
2703 013154  004737  004036                 JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2704 013160  005037  002714                 CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +204
2705 013164  012701  006250                 MOV     #PATRNS,R1      ;MOVE ADDRESS OF BIT PATTERNS TO R1
2706 013170  012702  000007                 MOV     #7,R2           ;DO 7 PATTERNS
2707 013174  012777  010000  167320 1$:     MOV     #MA,@CSR        ;GO TO MAINTENANCE MODE & SET GO TO DISABLE BAR BIT 0
2708 013202  011177  167316                 MOV     (R1),@BDR       ;MOVE THE DATA TO BDR
2709 013206  017737  167312  002564         MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
2710 013214  021137  002564                 CMP     (R1),RBDR       ;SEE IF EXPECTED DATA WAS RECEIVED
2711 013220  001423                         BEQ     5$              ;BRANCH IF SO
2712 013222  012737  013236  001310         MOV     #2$,$LPERR      ;SET UP LOOP ON ERROR LOCATION
2713 013230  011137  002576                 MOV     (R1),EBDR       ;MOVE EXPECTED DATA TO EBDR
2714 013234  000410                         BR      3$              ;SKIP OVER LOOP ON ERROR SETUP
2715 013236  011177  167262         2$:     MOV     (R1),@BDR       ;LOAD BIT PATTERN TO BDR
2716 013242  017737  167256  002564         MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
2717 013250  021137  002564                 CMP     (R1),RBDR       ;SEE IF DATA IS OK NOW
2718 013254  001401                         BEQ     4$              ;BRANCH OUT IF SO - OK NOW
2719 013256  104205                 3$:     ERROR   +205            ;BDR PATTERN NOT CORRECT
2720 013260  032777  001000  166052 4$:     BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
2721 013266  001363                         BNE     2$              ;BRANCH BACK IF SO
2722 013270  005077  167226         5$:     CLR     @CSR            ;CLEAR CSR TO DO AN INIT - WCR & BAR DON'T NEED MAINT MODE
2723 013274  011177  167216                 MOV     (R1),@WCR       ;MOVE THE DATA TO WCR
2724 013300  017737  167212  002570         MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
2725 013306  021137  002570                 CMP     (R1),RWCR       ;SEE IF EXPECTED DATA WAS RECEIVED
2726 013312  001423                         BEQ     9$              ;BRANCH IF SO
2727 013314  012737  013330  001310         MOV     #6$,$LPERR      ;SET UP LOOP ON ERROR LOCATION
2728 013322  011137  002602                 MOV     (R1),EWCR       ;MOVE EXPECTED DATA TO EWCR
2729 013326  000410                         BR      7$              ;SKIP OVER LOOP ON ERROR SETUP
2730 013330  011177  167162         6$:     MOV     (R1),@WCR       ;LOAD BIT PATTERN TO WCR
2731 013334  017737  167156  002570         MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
2732 013342  021137  002570                 CMP     (R1),RWCR       ;SEE IF DATA IS OK NOW
2733 013346  001401                         BEQ     8$              ;BRANCH OUT IF SO - OK NOW
2734 013350  104204                 7$:     ERROR   +204            ;WCR DATA PATTERN NOT CORRECT
2735 013352  032777  001000  165760 8$:     BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
2736 013360  001363                         BNE     6$              ;BRANCH BACK IF SO
2737 013362  011177  167132         9$:     MOV     (R1),@BAR       ;MOVE THE DATA TO BAR
2738 013366  017737  167126  002566         MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
2739 013374  011137  002600                 MOV     (R1),EBAR       ;MOVE EXPECTED DATA TO EBAR
2740 013400  042737  000001  002600         BIC     #BIT0,EBAR      ;DO NOT EXPECT BIT 0 TO BE READ
2741 013406  023737  002600  002566         CMP     EBAR,RBAR       ;SEE IF EXPECTED DATA WAS RECEIVED
```

```
2742 013414  001421                        BEQ     13$             ;BRANCH IF SO
2743 013416  012737  013426  001310        MOV     #10$,$LPERR     ;SET UP LOOP ON ERROR LOCATION
2744 013424  000410                        BR      11$             ;SKIP OVER LOOP ON ERROR SETUP
2745 013426  011177  167066          10$:  MOV     (R1),@BAR       ;LOAD BIT PATTERN TO BAR
2746 013432  017737  167062  002566        MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
2747 013440  021137  002566                CMP     (R1),RBAR       ;SEE IF DATA IS OK NOW
2748 013444  001401                        BEQ     12$             ;BRANCH OUT IF SO - OK NOW
2749 013446  104203                  11$:  ERROR   +203            ;BAR DATA PATTERN NOT CORRECT
2750 013450  032777  001000  165662  12$:  BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
2751 013456  001363                        BNE     10$             ;BRANCH BACK IF SO
2752 013460  005721                  13$:  TST     (R1)+           ;GO TO NEXT PATTERN
2753 013462  005302                        DEC     R2              ;DECREMENT THE LOOP COUNTER AND
2754 013464  001243                        BNE     1$              ;BRANCH BACK IF NOT DONE
2755 013466  004737  004036                JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
```

```
                                        .SBTTL  TEST #7 - TEST CSR AND EIR BIT0
                                        ;*****************************************************************
                                        ;*TEST 7         TEST CSR AND EIR BIT0
                                        ;*
                                        ;*      THIS TEST INSURES THAT BIT 0 OF THE CSR IS CLEAR WHEN IN CSR MODE (BIT
                                        ;*      15 CLEAR), AND SET WHEN IN EIR MODE (BIT 15 SET).
                                        ;*
                                        ;*****************************************************************
013472  032777  004000  165640  TST7:   BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
013500  001407                          BEQ     1001$           ;&& BRANCH IF NOT
013502  104401  013510                  TYPE    ,1000$          ;&& TYPE: "T # 7"
013506  000404                          BR      1001$           ;&& GET OVER ASCII
013510     124     040     043  1000$:  .ASCIZ  /T # 7/<CRLF>   ;&& THE ASCII MESSAGE
                                        .EVEN
013520                                  1001$:
013520  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
013522  012737  013530  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2763 013530  032737  000001  002610  999$:  BIT     #BIT0,BORW      ;TEST TO SEE IF WE ARE TESTING A DR11-W
2764 013536  001444                          BEQ     TST10           ;BRANCH TO NEXT TEST IF A DR11-B
2765 013540  005077  166756                  CLR     @CSR            ;FORCE ACCESS TO CSR
2766 013544  012737  000001  002536          MOV     #BIT0,BUT       ;MOVE BIT 0 INDICATOR TO BIT UNDER TEST LOCATION
2767 013552  017737  166744  002560          MOV     @CSR,RCSR       ;MOVE CSR CONTENTS TO RCSR
2768 013560  032737  000001  002560          BIT     #BIT0,RCSR      ;CLEAR ALL BUT BIT 0
2769 013566  001407                          BEQ     1$              ;BRANCH IF A ZERO
2770 013570  013737  002560  002572          MOV     RCSR,ECSR       ;MOVE CSR TO EXPECTED DATA, ECSR AND
2771 013576  042737  000001  002572          BIC     #BIT0,ECSR      ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
2772 013604  104014                          ERROR   +14             ;CSR BIT TEST FAILED
2773 013606  012777  100000  166706  1$:     MOV     #EIR,@CSR       ;GO TO EIR MODE
2774 013614  017737  166702  002562          MOV     @CSR,REIR       ;MOVE CSR CONTENTS TO RCSR
2775 013622  032737  000001  002562          BIT     #BIT0,REIR      ;CLEAR ALL BUT BIT 0
2776 013630  001007                          BNE     TST10           ;BRANCH IF NOT A ZERO
2777 013632  013737  002562  002574          MOV     REIR,EEIR       ;MOVE CONTENTS TO ECSR ALSO AND
2778 013640  052737  000001  002574          BIS     #BIT0,EEIR      ;SET THE 0 BIT - EXPECTED IT TO BE 1
2779 013646  104015                          ERROR   +15             ;EIR BIT TEST FAILED
```

```
                                .SBTTL  TEST #10 - ATTN CAN BE SET VIA FNCT2 & ERROR BIT SETS
                                ;*********************************************************************
                                ;*TEST 10       ATTN CAN BE SET VIA FNCT2 & ERROR BIT SETS
                                ;*
                                ;*      THIS TEST INSURES THAT THE ATTN BIT (BIT 13) SETS VIA FNCT2 AND ERROR
                                ;*      BIT SETS.
                                ;*
                                ;*********************************************************************
013650  032777  004000  165462  TST10:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
013656  001407                         BEQ     1001$           ;&& BRANCH IF NOT
013660  104401  013666                 TYPE    ,1000$          ;&& TYPE: "T # 10"
013664  000404                         BR      1001$           ;&& GET OVER ASCII
013666     124     040     043  1000$:  .ASCIZ  /T # 10/<CRLF>  ;&& THE ASCII MESSAGE
                                       .EVEN
013676                          1001$:
013676  000004                         SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
013700  012737  013716  001310         MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2787 013706  032737  000001  002720    BIT     #BIT0,DDW       ;TEST TO SEE IF WE ARE TESTING A DR11-W
2788 013714  001073                    BNE     TST11           ;BRANCH TO NEXT TEST IF A DR11-B
2789 013716  005077  166600  999$:     CLR     @CSR            ;FORCE ACCESS TO CSR
2790 013722  012777  010000  166572    MOV     #MA,@CSR        ;MAINT
2791 013730  017737  166566  002560    MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2792 013736  022737  010200  002560    CMP     #MA+RY,RCSR     ;SEE IF EXPECTED DATA WAS RECEIVED
2793 013744  001404                    BEQ     1$              ;BRANCH IF THEY ARE
2794 013746  012737  010200  002572    MOV     #MA+RY,ECSR     ;MOVE EXPECTED DATA TO ECSR
2795 013754  104016                    ERROR   +16             ;READY AND MAINTENANCE ARE NOT THE ONLY BITS SET IN CSR
2796 013756  112777  000004  166536  1$:      MOVB    #F2,@CSR        ;SET FNCT2
2797 013764  017737  166532  002560    MOV     @CSR,RCSR       ;MOVE THE CONTENTS TO RCSR
2798 013772  013701  002560            MOV     RCSR,R1         ;MOVE CONTENTS TO R1 FOR BIT TEST
2799 013776  042701  057777            BIC     #CB1513,R1      ;CLEAR ALL BUT BITS ERROR & ATTN FOR TEST
2800 014002  022701  120000            CMP     #ER+AT,R1       ;TEST TO SEE IF ERROR AND ATTN ARE SET
2801 014006  001411                    BEQ     2$              ;BRANCH IF IT IS PROPERLY SET
2802 014010  013737  002560  002572    MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
2803 014016  004737  004060            JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
2804 014022  052737  120000  002572    BIS     #ER+AT,ECSR     ;SET THE BITS THAT SHOULD HAVE BEEN SET
2805 014030  104017                    ERROR   +17             ;ATTN AND ERROR FAILED TO SET PROPERLY
2806 014032  042777  020004  166462  2$:      BIC     #AT+F2,@CSR     ;CLEAR ATTN & FNCT2
2807 014040  017737  166456  002560    MOV     @CSR,RCSR       ;MOVE CSR DATA TO RCSR
2808 014046  032737  120000  002560    BIT     #ER+AT,RCSR     ;BIT TEST ATTN AND ERROR BITS TO SEE IF THEY ARE CLEAR
2809 014054  001411                    BEQ     3$              ;BRANCH IF ATTN IS CLEAR
2810 014056  013737  002560  002572    MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
2811 014064  042737  120000  002572    BIC     #ER+AT,ECSR     ;CLEAR THE BITS THAT WERE SUPPOSED TO BE CLEAR
2812 014072  004737  004060            JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
2813 014076  104020                    ERROR   +20             ;ATTN AND ERROR FAILED TO CLEAR PROPERLY
2814 014100  005077  166416  3$:      CLR     @CSR            ;RETURN TO CSR
```

```
                                .SBTTL  TEST #11 - FNCT BIT 1 CONTROLS DSTAT BIT 9
                                ;*********************************************************************
                                ;*TEST 11        FNCT BIT 1 CONTROLS DSTAT BIT 9
                                ;*
                                ;*      THIS TEST INSURES THAT FNCT BIT 1 CONTROLS DSTAT BIT 9.
                                ;*
                                ;*********************************************************************
     014104  032777  004000  165226  TST11:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     014112  001407                          BEQ     1001$           ;&& BRANCH IF NOT
     014114  104401  014122                  TYPE    ,1000$          ;&& TYPE: "T # 11"
     014120  000404                          BR      1001$           ;&& GET OVER ASCII
     014122     124     040     043  1000$:  .ASCIZ   /T # 11/<CRLF> ;&& THE ASCII MESSAGE
                                             .EVEN
     014132                          1001$:
     014132  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     014134  012737  014142  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2821 014142  005077  166354          999$:   CLR     @CSR            ;CLR FUNCT BITS AND FORCE ACCESS TO CSR
2822 014146  012777  010000  166346          MOV     #MA,@CSR        ;MAINT MODE
2823 014154  017737  166342  002560          MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
2824 014162  013737  002560  002572          MOV     RCSR,ECSR       ;MOVE EXPECTED TO ECSR
2825 014170  013701  002572                  MOV     ECSR,R1         ;MOVE CONTENTS TO R1 FOR TESTING
2826 014174  042701  177761                  BIC     #CFNC,R1        ;CLEAR ALL BUT THE FNCT BITS
2827 014200  001404                          BEQ     1$              ;BRANCH IF THE FUNCTION BITS ARE CLEAR
2828 014202  042737  000016  002572          BIC     #FNC,ECSR       ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
2829 014210  104024                          ERROR   +24             ;FUNCTION BIT(S) ARE NOT CLEAR
2830 014212  052777  000002  166302  1$:     BIS     #F1,@CSR        ;SET FNCT1
2831 014220  017737  166276  002560          MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
2832 014226  013737  002560  002572          MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
2833 014234  013701  002572                  MOV     ECSR,R1         ;MOVE CONTENTS TO R1 FOR TEST
2834 014240  042701  170777                  BIC     #CDST,R1        ;CLEAR ALL BUT BITS 9, 10 & 11
2835 014244  022701  001000                  CMP     #DSC,R1         ;SEE IF DSTAT A AND B ARE CLEAR & C IS SET
2836 014250  001407                          BEQ     TST12           ;BRANCH TO NEXT TEST IF ALL CLEAR
2837 014252  042737  006000  002572          BIC     #DAB,ECSR       ;CLEAR THE DSTAT A & B BITS THAT SHOULD HAVE BEEN CLEAR
2838 014260  052737  001000  002572          BIS     #DSC,ECSR       ;SET THE DSTAT C BIT THAT SHOULD HAVE BEEN SET
2839 014266  104025                          ERROR   +25             ;DSTAT A, B OR C ARE NOT AS EXPECTED
```

```
                                        .SBTTL  TEST #12 - FNCT BIT 2 CONTROLS DSTAT BIT 10
                                        ;**********************************************************************
                                        ;*TEST 12       FNCT BIT 2 CONTROLS DSTAT BIT 10
                                        ;*
                                        ;*      THIS TEST INSURES THAT FNCT BIT 2 CONTROLS DSTAT BIT 10.
                                        ;*
                                        ;**********************************************************************
     014270  032777  004000  165042  TST12:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     014276  001407                          BEQ     1001$           ;&& BRANCH IF NOT
     014300  104401  014306                  TYPE    ,1000$          ;&& TYPE: "T # 12"
     014304  000404                          BR      1001$           ;&& GET OVER ASCII
     014306     124     040     043  1000$:  .ASCIZ   /T # 12/<CRLF>  ;&& THE ASCII MESSAGE
                                             .EVEN
     014316                          1001$:
     014316  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     014320  012737  014326  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2846 014326  005077  166170          999$:   CLR     @CSR            ;CLR FUNCT BITS AND FORCE ACCESS TO CSR
2847 014332  012777  010000  166162          MOV     #MA,@CSR        ;MAINT MODE
2848 014340  017737  166156  002560          MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
2849 014346  013737  002560  002572          MOV     RCSR,ECSR       ;MOVE EXPECTED TO ECSR
2850 014354  013701  002572                  MOV     ECSR,R1         ;MOVE CONTENTS TO R1 FOR TESTING
2851 014360  042701  177761                  BIC     #CFNC,R1        ;CLEAR ALL BUT THE FNCT BITS
2852 014364  001404                          BEQ     1$              ;BRANCH IF THE FUNCTION BITS ARE CLEAR
2853 014366  042737  000016  002572          BIC     #FNC,ECSR       ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
2854 014374  104024                          ERROR   +24             ;FUNCTION BIT(S) ARE NOT CLEAR
2855 014376  052777  000004  166116  1$:     BIS     #F2,@CSR        ;SET FNCT2
2856 014404  017737  166112  002560          MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
2857 014412  013737  002560  002572          MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
2858 014420  013701  002572                  MOV     ECSR,R1         ;MOVE CONTENTS TO R1 FOR TEST
2859 014424  042701  170777                  BIC     #CDST,R1        ;CLEAR ALL BUT THE DSTAT BITS
2860 014430  022701  002000                  CMP     #DSB,R1         ;IF DSTAT A AND C ARE CLEAR & B IS SET
2861 014434  001407                          BEQ     TST13           ;BRANCH TO NEXT TEST IF AS EXPECTED
2862 014436  042737  005000  002572          BIC     #DAC,ECSR       ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
2863 014444  052737  002000  002572          BIS     #DSB,ECSR       ;SET THE BIT THAT SHOULD HAVE BEEN SET
2864 014452  104025                          ERROR   +25             ;DSTAT A, B OR C ARE NOT AS EXPECTED
```

```
                                        .SBTTL  TEST #13 - FNCT BIT 3 CONTROLS DSTAT BIT 11
                                        ;**************************************************************
                                        ;*TEST 13       FNCT BIT 3 CONTROLS DSTAT BIT 11
                                        ;*
                                        ;*      THIS TEST INSURES THAT FNCT BIT 3 CONTROLS DSTAT BIT 11.
                                        ;*
                                        ;**************************************************************
     014454  032777  004000  164656  TST13: BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     014462  001407                        BEQ     1001$           ;&& BRANCH IF NOT
     014464  104401  014472                TYPE    ,1000$          ;&& TYPE: "T # 13"
     014470  000404                        BR      1001$           ;&& GET OVER ASCII
     014472     124     040     043  1000$: .ASCIZ  /T # 13/<CRLF>  ;&& THE ASCII MESSAGE
                                            .EVEN
     014502                             1001$:
     014502  000004                        SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     014504  012737  014512  001310        MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2871 014512  005077  166004          999$: CLR     @CSR            ;CLR FUNCT BITS AND FORCE ACCESS TO CSR
2872 014516  012777  010000  165776        MOV     #MA,@CSR        ;MAINT MODE
2873 014524  017737  165772  002560        MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
2874 014532  013737  002560  002572        MOV     RCSR,ECSR       ;MOVE EXPECTED TO ECSR
2875 014540  013701  002572                MOV     ECSR,R1         ;MOVE CONTENTS TO R1 FOR TESTING
2876 014544  042701  177761                BIC     #CFNC,R1        ;CLEAR ALL BUT THE FNCT BITS
2877 014550  001404                        BEQ     1$              ;BRANCH IF THE FUNCTION BITS ARE CLEAR
2878 014552  042737  000016  002572        BIC     #FNC,ECSR       ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
2879 014560  104024                        ERROR   +24             ;FUNCTION BIT(S) ARE NOT CLEAR
2880 014562  052777  000010  165732  1$:   BIS     #F3,@CSR        ;SET FNCT3
2881 014570  017737  165726  002560        MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
2882 014576  013737  002560  002572        MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
2883 014604  013701  002572                MOV     ECSR,R1         ;MOVE CONTENTS TO R1 FOR TEST
2884 014610  042701  170777                BIC     #CDST,R1        ;CLEAR ALL BUT DSTAT BITS
2885 014614  022701  004000                CMP     #DSA,R1         ;SEE IF DSTAT B AND C ARE CLEAR & A IS SET
2886 014620  001407                        BEQ     TST14           ;BRANCH TO NEXT TEST IF DATA OK
2887 014622  042737  003000  002572        BIC     #DBC,ECSR       ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
2888 014630  052737  004000  002572        BIS     #DSA,ECSR       ;SET THE BIT THAT SHOULD HAVE BEEN SET
2889 014636  104025                        ERROR   +25             ;DSTAT A, B OR C ARE NOT AS EXPECTED
```

```
                                .SBTTL  TEST #14 - EIR BLOCKS DATA XFERS FROM ODR TO IDR
                                ;*****************************************************************
                                ;*TEST 14       EIR BLOCKS DATA XFERS FROM ODR TO IDR
                                ;*
                                ;*      THIS TEST INSURES THAT GOING TO EIR MODE BLOCKS DATA TRANSFERS FROM
                                ;*      ODR TO IDR (ODR RECEIVES DATA WHEN WRITING TO THE BDR, AND WHEN READING
                                ;*      THE BDR, THE IDR IS READ).
                                ;*
                                ;*****************************************************************
 014640  032777  004000  164472  TST14:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
 014646  001407                        BEQ     1001$           ;&& BRANCH IF NOT
 014650  104401  014656                TYPE    ,1000$          ;&& TYPE: "T # 14"
 014654  000404                        BR      1001$           ;&& GET OVER ASCII
 014656     124     040     043  1000$:  .ASCIZ  /T # 14/<CRLF>  ;&& THE ASCII MESSAGE
                                        .EVEN
 014666                         1001$:
 014666  000004                        SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
 014670  012737  014706  001310        MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2898 014676  032737  000001  002610    BIT     #BIT0,BCRW      ;TEST TO SEE IF WE ARE TESTING A DR11-W
2899 014704  001451                    BEQ     TST15           ;BRANCH TO NEXT TEST IF A DR11-B
2900 014706  005077  165610     999$:  CLR     @CSR            ;FORCE ACCESS TO CSR
2901 014712  012777  010000  165602    MOV     #MA,@CSR        ;SET MAINT MODE (SO THAT IDR GETS ODR CONTENTS)
2902 014720  012777  052525  165576    MOV     #52525,@BDR     ;SET ALT 0'S AND 1'S TO BDR
2903 014726  017737  165572  002564    MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
2904 014734  022737  052525  002564    CMP     #52525,RBDR     ;SEE IF DATA WAS LOADED PROPERLY
2905 014742  001404                    BEQ     1$              ;BRANCH IF IT WAS
2906 014744  012737  052525  002576    MOV     #52525,EBDR     ;MOVE EXPECTED DATA TO EBDR
2907 014752  104031                    ERROR   +31             ;BDR PATTERN NOT CORRECT
2908 014754  052777  100000  165540  1$:     BIS     #EIR,@CSR       ;GO TO EIR
2909 014762  012737  052525  002576    MOV     #52525,EBDR     ;MOVE EXPECTED DATA TO EBDR
2910 014770  012777  125252  165526    MOV     #125252,@BDR    ;SET ALT 1'S AND 0'S TO BDR
2911 014776  017737  165522  002564    MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
2912 015004  022737  052525  002564    CMP     #52525,RBDR     ;TEST FOR OLD PATTERN
2913 015012  001404                    BEQ     2$              ;BRANCH IF ORIGINAL PATTERN STILL THERE
2914 015014  012737  052525  002576    MOV     #5, 25,EBDR     ;MOVE EXPECTED DATA TO EBDR
2915 015022  104030                    ERROR   +30             ;BDR SHOULD NOT HAVE BEEN LOADED WITH NEW PATTERN
2916 015024  004737  004036     2$:    JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
```

```
2923                                    .SBTTL  TEST #15 - DR11 INTERRUPTS WITH CPU AT LEVEL 3
                                        ;**********************************************************************
                                        ;*TEST 15       DR11 INTERRUPTS WITH CPU AT LEVEL 3
                                        ;*
                                        ;*      THIS TEST INSURES THAT THE DR11 WILL INTERRUPT WITH THE CPU PRIORITY
                                        ;*      AT LEVEL 3.
                                        ;*
                                        ;**********************************************************************
     015030  032777  004000  164302 TST15:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     015036  001407                         BEQ     1001$           ;&& BRANCH IF NOT
     015040  104401  015046                 TYPE    ,1000$          ;&& TYPE: "T # 15"
     015044  000404                         BR      1001$           ;&& GET OVER ASCII
     015046     124     040     043 1000$:  .ASCIZ   /T # 15/<CRLF>  ;&& THE ASCII MESSAGE
                                            .EVEN
     015056                             1001$:
     015056  000004                         SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     015060  012737  015066  001310         MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2924 015066  012777  010000  165426 999$:   MOV     #MA,@CSR        ;SET MAINTENANCE BIT AND
2925 015074  005077  165422                 CLR     @CSR            ;CLEAR CSR TO DO AN INIT
2926 015100  012737  000140  177776         MOV     #LEVEL3,PSW     ;STATUS AT LEVEL 3
2927 015106  017737  165410  002560         MOV     @CSR,RCSR       ;MOVE CSR CONTENTS TO RCSR
2928 015114  105737  002560                 TSTB    RCSR            ;SEE IF READY BIT (BIT 7) IS SET
2929 015120  100406                         BMI     1$              ;BRANCH IF IT IS
2930 015122  012737  000200  002572         MOV     #RY,ECSR        ;SET THE BIT THAT SHOULD HAVE BEEN SET
2931 015130  004737  004060                 JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
2932 015134  104022                         ERROR   +22             ;READY OF CSR WAS NOT SET
2933 015136  017737  165364  002532 1$:     MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2934 015144  017737  165360  002534         MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2935 015152  012777  015242  165346         MOV     #3$,@DRINV      ;SET UP INTERRUPT VECTOR
2936 015160  012777  010000  165334         MOV     #MA,@CSR        ;MAINT MODE
2937 015166  012737  001000  002660         MOV     #1000,TIME      ;SET THE TIME COUNTER
2938 015174  052777  000105  165320         BIS     #IE+F2+GO,@CSR  ;IE, FNCT2 AND GO
2939 015202  005337  002660          2$:    DEC     TIME            ;DECREMENT DOWN TO ZERO
2940 015206  001375                         BNE     2$              ;BRANCH IF NOT THERE YET
2941 015210  017737  165306  002560         MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2942 015216  013777  002532  165302         MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2943 015224  013777  002534  165276         MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2944 015232  104035                         ERROR   +35             ;DR11 FAILED TO INTEPRUPT
2945 015234  005077  165262                 CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
2946 015240  000410                         BR      TST16           ;BRANCH TO THE NEXT TEST
2947 015242  062706  000004          3$:    ADD     #4,SP           ;CLEAN THE STACK AFTER THE INTERRUPT
2948 015246  013777  002532  165252         MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2949 015254  013777  002534  165246         MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
```

```
2956                                    .SBTTL  TEST #16 - DR11 FAILS TO INTERRUPT WITH CPU AT LEVEL 7
                                        ;*****************************************************************
                                        ;*TEST 16       DR11 FAILS TO INTERRUPT WITH CPU AT LEVEL 7
                                        ;*
                                        ;*      THIS TEST INSURES THAT THE DR11 FAILS TO INTERRUPT WITH THE CPU PRIORITY
                                        ;*      AT LEVEL 7.
                                        ;*
                                        ;*****************************************************************
 015262  032777  004000  164050 TST16:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
 015270  001407                         BEQ     1001$           ;&& BRANCH IF NOT
 015272  104401  015300                 TYPE    ,1000$          ;&& TYPE: "T # 16"
 015276  000404                         BR      1001$           ;&& GET OVER ASCII
 015300     124     040     043 1000$:  .ASCIZ   /T # 16/<CRLF>  ;&& THE ASCII MESSAGE
                                        .EVEN
 015310                             1001$:
 015310  000004                         SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
 015312  012737  015320  001310         MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2957 015320  004737  004036     999$:  JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2958 015324  017737  165172  002560    MOV     @CSR,RCSR       ;MOVE CSR DATA TO RCSR
2959 015332  105737  002560            TSTB    RCSR            ;CLEAR ALL BUT THE READY BIT (BIT 7)
2960 015336  100411                    BMI     1$              ;BRANCH IF IT IS SET
2961 015340  012737  000200  002572    MOV     #RY,ECSR        ;MOVE EXPECTED DATA TO ECSR
2962 015346  004737  004060            JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
2963 015352  052737  000200  002572    BIS     #RY,ECSR        ;SET THE BIT THAT SHOULD HAVE BEEN SET
2964 015360  104022                    ERROR   +22             ;READY OF CSR WAS NOT SET
2965 015362  017737  165140  002532 1$:     MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2966 015370  017737  165134  002534        MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2967 015376  012777  015464  165122        MOV     #3$,@DRINV      ;SET UP INT VECTOR
2968 015404  012777  000340  165116        MOV     #LEVEL7,@DRVS
2969 015412  012737  001000  002660        MOV     #1000,TIME      ;SET TIME DELAY COUNTER
2970 015420  012777  010000  165074        MOV     #MA,@CSR        ;MAINT MODE
2971 015426  052777  000105  165066        BIS     #IE+F2+GO,@CSR  ;IE, FNCT2 AND GO
2972 015434  005337  002660         2$:    DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
2973 015440  001375                    BNE     2$              ;BRANCH BACK IF NOT ZERO YET
2974 015442  005077  165054            CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
2975 015446  013777  002532  165052    MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2976 015454  013777  002534  165046    MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2977 015462  000416                    BR      TST17           ;BRANCH TO THE NEXT TEST
2978 015464  062706  000004         3$:    ADD     #4,SP           ;RESTORE STACK
2979 015470  017737  165026  002560        MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2980 015476  005077  165020            CLR     @CSR            ;CLEAR IE
2981 015502  013777  002532  165016    MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2982 015510  013777  002534  165012    MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2983 015516  104036                    ERROR   +36             ;DR11 INTERRUPTED, BUT IT SHOULDN'T HAVE
```

```
                                .SBTTL  TEST #17 - DR11 INTERRUPTS AT CORRECT BR LEVEL
                                ;*****************************************************************
                                ;*TEST 17       DR11 INTERRUPTS AT CORRECT BR LEVEL
                                ;*
                                ;*      THIS TEST INSURES THAT THE DR11 WILL INTERRUPT AT THE CORRECT LEVEL AS
                                ;*      DEFINED IN THE DEVICE DESCRIPTOR WORD.
                                ;*
                                ;*****************************************************************
 015520  032777  004000  163612 TST17:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
 015526  001407                        BEQ     1001$           ;&& BRANCH IF NOT
 015530  104401  015536                TYPE    ,1000$          ;&& TYPE: "T # 17"
 015534  000404                        BR      1001$           ;&& GET OVER ASCII
 015536     124     040     043 1000$: .ASCIZ  /T # 17/<CRLF>  ;&& THE ASCII MESSAGE
                                        .EVEN
 015546                         1001$:
 015546  000004                        SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
 015550  012737  015632  001310        MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2991 015556  004737  004036            JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2992 015562  013737  002720  001362    MOV     DDW,$TMP1       ;MOVE DEVICE DESCRIPTOR WORD TO $TMP1
2993 015570  006237  001362            ASR     $TMP1           ;SHIFT THE LEVEL TO THE RIGHT 5 PLACES
2994 015574  006237  001362            ASR     $TMP1
2995 015600  006237  001362            ASR     $TMP1
2996 015604  006237  001362            ASR     $TMP1
2997 015610  006237  001362            ASR     $TMP1
2998 015614  042737  177770  001362    BIC     #177770,$TMP1   ;CLEAR ALL BUT THE PRIORITY
2999 015622  012700  000003        1$: MOV     #3,R0           ;DO 3 PRIORITY LEVELS
3000 015626  012701  005264            MOV     #LEVELS,R1      ;MOVE ADDRESS OF CPU PRIORITIES TO R1
3001 015632  012777  010000  164662 999$: MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
3002 015640  005077  164656            CLR     @CSR            ;CLEAR TO DO AN INIT
3003 015644  011137  177776            MOV     (R1),PSW        ;PUT PRIORITY INTO PSW
3004 015650  017737  164646  002560    MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3005 015656  012737  000200  002572    MOV     #RY,ECSR        ;MOVE READY BIT TO ECSR
3006 015664  004737  004060            JSP     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
3007 015670  023737  002560  002572    CMP     RCSR,ECSR       ;SEE IF RECEIVED DATA MATCHES EXPECTED
3008 015676  001412                    BEQ     2$              ;BRANCH IF OK
3009 015700  012737  015622  001310    MOV     #1$,$LPERR      ;SET UP FOR POSSIBLE LOOP ON ERROR FOR THIS ERROR ONLY
3010 015706  012737  000200  002572    MOV     #RY,ECSR        ;MOVE EXPECTED DATA TO ECSR
3011 015714  104022                    ERROR   +22             ;READY OF CSR WAS NOT SET
3012 015716  012737  015632  001310    MOV     #999$,$LPERR    ;RETURN ORIGINAL LOOP ON ERROR ADDRESS - DID NOT LOOP
3013 015724  017737  164576  002532 2$: MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3014 015732  017737  164572  002534    MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3015 015740  012777  016052  164560    MOV     #4$,@DRINV      ;SET UP INTERRUPT VECTOR
3016 015746  012777  000340  164554    MOV     #LEVEL7,@DRVS   ;SET UP INTERRUPT PS
3017 015754  012777  010000  164540    MOV     #MA,@CSR        ;MAINT MODE
3018 015762  012737  000400  002660    MOV     #400,TIME       ;SET DELAY COUNTER
3019 015770  052777  000105  164524    BIS     #IE+F2+GO,@CSR  ;IE, FNCT2 AND GO
3020 015776  005337  002660        3$: DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
3021 016002  001375                    BNE     3$              ;BRANCH BACK IF NOT ZERO YET
3022 016004  005077  164512            CLR     @CSR            ;CLEAR CSR TO DO AN INIT
3023 016010  013777  002532  164510    MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3024 016016  013777  002534  164504    MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3025 016024  013737  177776  002540    MOV     PSW,LEVEL       ;SAVE OLD STATUS LEVEL
3026 016032  005721                    TST     (R1)+           ;INCREMENT R1 TO POINT TO NEXT PRIORITY LEVEL
3027 016034  005300                    DEC     R0              ;DECREMENT LOOP COUNTER AND
3028 016036  001275                    BNE     999$            ;BRANCH BACK FOR ANOTHER TRY IF NOT DONE
3029 016040  104053                    ERROR   +53             ;DR11 FAILED TO INTERRUPT
3030 016042  013737  002552  002540    MOV     DRLEV,LEVEL     ;SET LEVEL TO CONTAIN THE ANTICIPATED LEVEL
```

```
3031 016050  000422                        BR      TST20           ;BRANCH TO THE NEXT TEST
3032 016052  062706  000004          4$:   ADD     #4,SP           ;RESTORE STACK
3033 016056  005077  164440                CLR     @CSR            ;CLEAR IE
3034 016062  013777  002532  164436        MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3035 016070  013777  002534  164432        MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3036 016076  042737  177437  002540        BIC     #177437,LEVEL   ;CLEAR ALL BITS BUT THE BR LEVEL
3037 016104  023737  002540  002552        CMP     LEVEL,DRLEV     ;SEE IF LEVEL INTERRUPTED MATCHES EXPECTED
3038 016112  001401                        BEQ     TST20           ;BRANCH AROUND ERROR CALL IF IT IS AS EXPECTED
3039 016114  104052                        ERROR   +52             ;DR11 INTERRUPTED AT WRONG LEVEL
```

```
                                        .SBTTL  TEST #20 - A GO WITHOUT CLEARING ERROR CAUSES INTRPT
                                        ;*****************************************************************
                                        ;*TEST 20       A GO WITHOUT CLEARING ERROR CAUSES INTRPT
                                        ;*
                                        ;*      THIS TEST INSURES THAT SETTING THE GO BIT WITHOUT CLEARING THE ERROR
                                        ;*      BIT CAUSES AN INTERRUPT.
                                        ;*
                                        ;*****************************************************************
     016116  032777  004000  163214  TST20:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     016124  001407                          BEQ     1001$           ;&& BRANCH IF NOT
     016126  104401  016134                  TYPE    ,1000$          ;&& TYPE: "T # 20"
     016132  000404                          BR      1001$           ;&& GET OVER ASCII
     016134     124     040     043  1000$:  .ASCIZ   /T # 20/<CRLF>  ;&& THE ASCII MESSAGE
                                             .EVEN
     016144                               1001$:
     016144  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     016146  012737  016154  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3047 016154  004737  004036           999$:  JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3048 016160  017737  164342  002532          MOV     @DRINV,$DRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3049 016166  017737  164336  002534          MOV     @DRVS,$DRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3050 016174  012777  016276  164324          MOV     #2$,@DRINV      ;INTERRUPT VECTOR TO 3$
3051 016202  012777  000140  164320          MOV     #LEVEL3,@DRVS   ;INTERRUPT STATUS TO LEVEL 3
3052 016210  005037  177776                  CLR     PSW             ;LET THE DR11 INTERRUPT
3053 016214  012737  001000  002660          MOV     #1000,TIME      ;MOVE DELAY COUNTER TO LOCATION
3054 016222  012777  010101  164272          MOV     #MA+IE+GO,@CSR  ;SET MAINT, IE AND GO
3055 016230  052777  020004  164264          BIS     #AT+F2,@CSR     ;SET ATTN AND FNCT2
3056 016236  005337  002660           1$:    DEC     TIME            ;DECREMENT UNTIL WE REACH ZERO
3057 016242  001375                          BNE     1$              ;BRANCH IF NOT ZERO YET
3058 016244  013777  002532  164254          MOV     $DRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3059 016252  013777  002534  164250          MOV     $DRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3060 016260  017737  164236  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3061 016266  104035                          ERROR   +35             ;DR11 FAILED TO INTERRUPT
3062 016270  005077  164226                  CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
3063 016274  000512                          BR      TST21           ;BRANCH TO THE NEXT TEST
3064 016276  062706  000004           2$:    ADD     #4,SP           ;READJUST STACK AFTER THE INTERRUPT
3065 016302  013777  002532  164216          MOV     $DRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3066 016310  013777  002534  164212          MOV     $DRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3067 016316  017737  164200  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3068 016324  100407                          BMI     3$              ;BRANCH IF ERROR IS SET
3069 016326  013737  002560  002572          MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
3070 016334  052737  100000  002572          BIS     #ER,ECSR        ;SET THE BIT THAT SHOULD HAVE BEEN SET
3071 016342  104037                          ERROR   +37             ;ERROR BIT SHOULD NOT BE CLEAR
3072 016344  017737  164156  002532  3$:     MOV     @DRINV,$DRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3073 016352  017737  164152  002534          MOV     @DRVS,$DRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3074 016360  012777  016454  164140          MOV     #5$,@DRINV      ;INTERRUPT VECTOR TO 6$
3075 016366  005077  164126                  CLR     @BAR            ;PREVENT CAUSING ANOTHER ERROR
3076 016372  012777  177777  164116          MOV     #-1,@WCR        ;SET-UP WCR
3077 016400  012737  001000  002660          MOV     #1000,TIME      ;LOAD 1000 IN LOCATION TIME FOR WAIT LOOP
3078 016406  052777  000001  164106          BIS     #GO,@CSR        ;SET 'GO' IN CSR
3079 016414  005237  002660           4$:    INC     TIME            ;DELAY - WAIT FOR INTERRUPT
3080 016420  001375                          BNE     4$              ;BRANCH BACK IF NOT ZERO
3081 016422  013777  002532  164076          MOV     $DRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3082 016430  013777  002534  164072          MOV     $DRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3083 016436  017737  164060  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3084 016444  104035                          ERROR   +35             ;DR11 FAILED TO INTERRUPT
3085 016446  005077  164050                  CLR     @CSR            ;CLEAR CSR TO DO AN INIT
3086 016452  000423                          BR      TST21           ;BRANCH TO THE NEXT TEST
```

```
3087 016454  062706  000004          5$:     ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
3088 016460  013777  002532  164040          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3089 016466  013777  002534  164034          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3090 016474  017737  164022  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR - IS ERROR CLEAR
3091 016502  100007                          BPL     TST21           ;BRANCH TO NEXT TEST IF IT IS
3092 016504  013737  002560  002572          MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
3093 016512  042737  100000  002572          BIC     #ER,ECSR        ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
3094 016520  104021                          ERROR   +21             ;ERROR BIT SHOULD HAVE BEEN CLEAR
```

```
3101                                    .SBTTL  TEST #21 - FUNCTION BITS INC WITH MAINT MODE TRANSFERS
                                        ;*****************************************************************
                                        ;*TEST 21       FUNCTION BITS INC WITH MAINT MODE TRANSFERS
                                        ;*
                                        ;*      THIS TEST INSURES THAT THE FUNCTION BITS INCREMENT WITH MAINTENANCE
                                        ;*      MODE TRANSFERS.
                                        ;*
                                        ;*****************************************************************
     016522  032777  004000  162610  TST21:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     016530  001407                          BEQ     1001$           ;&& BRANCH IF NOT
     016532  104401  016540                  TYPE    ,1000$          ;&& TYPE: "T # 21"
     016536  000404                          BR      1001$           ;&& GET OVER ASCII
     016540     124     040     043  1000$:  .ASCIZ   /T # 21/<CRLF>  ;&& THE ASCII MESSAGE
                                             .EVEN
     016550                          1001$:
     016550  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     016552  012737  016574  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3102 016560  012737  000016  001364          MOV     #16,$TMP2       ;SET UP FUNCTION COUNT COMPARE
3103 016566  012737  177771  001360          MOV     #-7,$TMP0       ;SET UP WCR LOAD VARIABLE
3104 016574  004737  004036          999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3105 016600  013777  002616  163712          MOV     INBUF,@BAR      ;SET-UP BAR
3106 016606  017737  163714  002532  1$:     MOV     @DRINV,$DRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3107 016614  017737  163710  002534          MOV     @DRVS,$DRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3108 016622  012777  016732  163676          MOV     #3$,@DRINV      ;INTERRUPT VECTOR
3109 016630  013777  002540  163672          MOV     LEVEL,@DRVS     ;INTERRUPT VECTOR PRIORITY TO LEVEL OF DEVICE
3110 016636  013777  001360  163652          MOV     $TMP0,@WCR      ;SET UP FOR NUMBER OF TRANSFERS IN $TMP0
3111 016644  005037  177776                  CLR     PSW             ;LET THE DR11 INTERRUPT
3112 016650  012777  010000  163644          MOV     #MA,@CSR        ;MAINT MODE
3113 016656  012737  001000  002660          MOV     #1000,TIME      ;MOVE WAIT COUNTER TO LOCATION TIME
3114 016664  052777  000501  163630          BIS     #IE+CY+GO,@CSR  ;IE, CYCLE & GO
3115 016672  005337  002660          2$:     DEC     TIME            ;DECREMENT UNTIL ZERO
3116 016676  001375                          BNE     2$              ;BRANCH BACK IF NOT
3117 016700  017737  163616  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3118 016706  013777  002532  163612          MOV     $DRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3119 016714  013777  002534  163606          MOV     $DRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3120 016722  104035                          ERROR   +35             ;DR11 FAILED TO INTERRUPT
3121 016724  005077  163572                  CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
3122 016730  000442                          BR      TST22           ;BRANCH TO NEXT TEST
3123 016732  062706  000004          3$:     ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
3124 016736  013777  002532  163562          MOV     $DRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3125 016744  013777  002534  163556          MOV     $DRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3126 016752  017737  163544  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3127 016760  013701  002560                  MOV     RCSR,R1         ;MOVE RECEIVED DATA TO R1 ALSO AND
3128 016764  042701  177761                  BIC     #CFNC,R1        ;CLEAR ALL BUT THE FUNCTION BITS
3129 016770  020137  001364                  CMP     R1,$TMP2        ;SEE IF FUNCTION BIT(S) HAD INCREMENTED PROPERLY
3130 016774  001412                          BEQ     4$              ;BRANCH IF THEY HAD
3131 016776  013737  002560  002572          MOV     RCSR,ECSR       ;MOVE RECEIVED DATA TO EXPECTED LOCATION
3132 017004  042737  000016  002572          BIC     #FNC,ECSR       ;CLEAR THE FUNCTION BIT(S) THAT WERE THERE AND
3133 017012  053737  001364  002572          BIS     $TMP2,ECSR      ;PUT FUNCTION BIT(S) EXPECTED IN THEIR PLACE
3134 017020  104212                          ERROR   +212            ;FUNCTION BITS DIDN'T INCREMENT IN MAINT MODE
3135 017022  005237  001360          4$:     INC     $TMP0           ;ADJUST WCR LOAD LOCATION
3136 017026  162737  000002  001364          SUB     #2,$TMP2        ;SUBTRACT 2 FROM FUNCTION COUNT TEST LOCATION
3137 017034  001264                          BNE     1$              ;BRANCH BACK FOR ANOTHER TRY
```

```
                                        .SBTTL  TEST #22 - TEST FOR 10 MAINT MODE TRANSFERS
                                        ;*****************************************************************
                                        ;*TEST 22       TEST FOR 10 MAINT MODE TRANSFERS
                                        ;*
                                        ;*      THIS TEST CHECKS IF 10 MAINTENANCE MODE TRANSFERS CAN BE DONE.
                                        ;*
                                        ;*****************************************************************
017036  032777  004000  162274  TST22:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
017044  001407                          BEQ     1001$           ;&& BRANCH IF NOT
017046  104401  017054                  TYPE    ,1000$          ;&& TYPE: "T # 22"
017052  000404                          BR      1001$           ;&& GET OVER ASCII
017054     124     040     043  1000$:  .ASCIZ  /T # 22/<CRLF>  ;&& THE ASCII MESSAGE
                                        .EVEN
017064                          1001$:
017064  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
017066  012737  017074  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3144 017074  005077  163422  999$:     CLR     @CSR            ;FORCE ACCESS TO CSR
3145 017100  012737  000010  002622          MOV     #10,BUFLEN      ;BUFLEN=10
3146 017106  013737  002622  002630          MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
3147 017114  005437  002630                  NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
3148 017120  004737  003472                  JSR     PC,LODBUF       ;LOAD IN BUFFER WITH INCREMENTING PATTERN
3149 017124  004737  003520                  JSR     PC,CHKBFF       ;LOAD CHECK BUFFER WITH MODIFIED INCREMENTING PATTERN
3150 017130  013777  002630  163360          MOV     WCLEN,@WCR      ;SET UP WCR
3151 017136  013777  002616  163354          MOV     INBUF,@BAR      ;SET UP BAR
3152 017144  012777  177777  163352          MOV     #-1,@BDR        ;MAINT AIDE
3153 017152  017737  163350  002532          MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3154 017160  017737  163344  002534          MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3155 017166  012777  017270  163332          MOV     #2$,@DRINV      ;INTERRUPT VECTOR
3156 017174  013777  002540  163326          MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
3157 017202  005037  177776                  CLR     PSW             ;LET DR11 INTERRUPT
3158 017206  012777  010000  163306          MOV     #MA,@CSR        ;MAINT MODE
3159 017214  052777  000501  163300          BIS     #IE+CY+GO,@CSR  ;IE, CYCLE & GO
3160 017222  012737  001000  002660          MOV     #1000,TIME      ;SET LOOP COUNTER FOR WAIT
3161 017230  005337  002660          1$:     DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
3162 017234  001375                          BNE     1$              ;BRANCH BACK IF NOT ZERO
3163 017236  013777  002532  163262          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3164 017244  013777  002534  163256          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3165 017252  017737  163244  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3166 017260  104035                          ERROR   +35             ;DR11 FAILED TO INTERRUPT
3167 017262  005077  163234                  CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
3168 017266  000402                          BR      3$              ;BRANCH AROUND THE STACK CLEANUP
3169 017270  062706  000004          2$:     ADD     #4,SP           ;CLEAN UP STACK AFTER THE INTERRUPT
3170 017274  013777  002532  163224  3$:     MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3171 017302  013777  002534  163220          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3172 017310  004737  003546                  JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
3173 017314  104051                          ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
3174 017316  005037  002714                  CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
3175 017322  004737  003716                  JSR     PC,DATCHK       ;CHECK INBUF AFTER A MAINTENANCE MODE OPERATION
3176 017326  104201                          ERROR   +201            ;BUFFER DATA NOT CORRECT
3177 017330  004737  004014                  JSR     PC,DATCH2       ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATCHK
3178
```

```
                                                .SBTTL  TEST #23 - TEST 10 MAINTENANCE MODE XFERS
                                                ;*****************************************************************
                                                ;*TEST 23       TEST 10 MAINTENANCE MODE XFERS
                                                ;*
                                                ;*      THIS TEST CHECKS THAT 10 MAINTENANCE MODE TRANSFERS, ATTEMPTED BEFORE
                                                ;*      SERVICING A PENDING INTERRUPT OF A PREVIOUS TRANSFER, ARE UNSUCCESSFUL.
                                                ;*
                                                ;*****************************************************************
     017334  032777  004000  161776  TST23:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     017342  001407                           BEQ     1001$           ;&& BRANCH IF NOT
     017344  104401  017352                   TYPE    ,1000$          ;&& TYPE: "T # 23"
     017350  000404                           BR      1001$           ;&& GET OVER ASCII
     017352     124     040     043  1000$:  .ASCIZ  /T # 23/<CRLF>  ;&& THE ASCII MESSAGE
                                             .EVEN
     017362                                  1001$:
     017362  000004                           SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     017364  012737  017372  001310           MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3186 017372  004737  004036          999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3187 017376  012737  001000  002660          MOV     #1000,TIME      ;SET DELAY
3188 017404  012737  000010  002622          MOV     #10,BUFLEN      ;BUFLEN=10
3189 017412  013737  002622  002630          MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
3190 017420  005437  002630                  NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
3191 017424  004737  003472                  JSR     PC,LODBUF       ;LOAD IN BUFFER WITH INCREMENTING PATTERN
3192 017430  004737  003520                  JSR     PC,CHKBFF       ;LOAD CHECK BUFFER WITH MODIFIED INCREMENTING PATTERN
3193 017434  013777  002630  163054          MOV     WCLEN,@WCR      ;SET UP WCR
3194 017442  013777  002616  163050          MOV     INBUF,@BAR      ;SET UP BAR
3195 017450  012777  177777  163046          MOV     #-1,@BDR        ;MAINT AIDE
3196 017456  017737  163044  002532          MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3197 017464  017737  163040  002534          MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3198 017472  012777  017572  163026          MOV     #2$,@DRINV      ;INTERRUPT VECTOR
3199 017500  013777  002540  163022          MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
3200 017506  012777  010000  163006          MOV     #MA,@CSR        ;MAINT MODE
3201 017514  052777  000501  163000          BIS     #IE+CY+GO,@CSR  ;IE, CYCLE & GO
3202 017522  005337  002660          1$:     DEC     TIME            ;WAIT FOR TRANSFERS TO COMPLETE
3203 017526  001375                          BNE     1$              ;BRANCH BACK IF WE ARE STILL WAITING
3204 017530  013777  002532  162770          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3205 017536  013777  002534  162764          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3206 017544  004737  003546                  JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
3207 017550  104051                          ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
3208 017552  005037  002714                  CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
3209 017556  004737  003716                  JSR     PC,DATCHK       ;CHECK INBUF AFTER A MAINTENANCE MODE OPERATION
3210 017562  104201                          ERROR   +201            ;BUFFER DATA NOT CORRECT
3211 017564  004737  004014                  JSR     PC,DATCH2       ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATCHK
3212 017570  000415                          BR      3$              ;BRANCH TO CONTINUE
3213 017572  062706  000004          2$:     ADD     #4,SP           ;CLEAN UP THE STACK FROM THIS INTERRUPT
3214 017576  013777  002532  162722          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3215 017604  013777  002534  162716          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3216 017612  017737  162704  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3217 017620  104036                          ERROR   +36             ;DR11 INTERRUPTED, BUT IT SHOULDN'T HAVE
3218 017622  000523                          BR      TST24           ;BRANCH TO NEXT TEST
3219 017624  012777  010000  162670  3$:     MOV     #MA,@CSR        ;MAINT MODE
3220 017632  012737  001000  002660          MOV     #1000,TIME      ;SET TIME LOOP COUNTER
3221 017640  013777  002630  162650          MOV     WCLEN,@WCR      ;MOVE WCLEN TO WCR
3222 017646  013777  002616  162644          MOV     INBUF,@BAR      ;MOVE INBUF TO BAR
3223 017654  017737  162646  002532          MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3224 017662  017737  162642  002534          MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3225 017670  012777  020042  162630          MOV     #8$,@DRINV      ;SET UP INTERRUPT VECTOR
```

```
3226 017676  012777  010000  162616          MOV     #MA,@CSR        ;MAINT MODE
3227 017704  052777  000501  162610          BIS     #IE+CY+GO,@CSR  ;IE, CYCLE & GO
3228 017712  005337  002660          4$:     DEC     TIME            ;DECREMENT TO ZERO WHILE WAITING
3229 017716  001375                          BNE     4$              ;BRANCH BACK IF NOT ZERO
3230 017720  013777  002532  162600          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3231 017726  013777  002534  162574          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3232 017734  017737  162562  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3233 017742  022737  010700  002560          CMP     #10700,RCSR     ;SEE IF ONLY READY, MAINT, IE & CYCLE ARE SET
3234 017750  001404                          BEQ     5$              ;BRANCH IF THEY ARE
3235 017752  012737  010700  002572          MOV     #10700,ECSR     ;MOVE EXPECTED DATA TO ECSR
3236 017760  104040                          ERROR   +40             ;CSR IS WRONG
3237 017762  017737  162532  002566  5$:     MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
3238 017770  022777  177770  162520          CMP     #-10,@WCR       ;CHECK THAT NO TRANSFERS WERE MADE
3239 017776  001004                          BNE     6$              ;BRANCH TO ERROR IF THERE WERE
3240 020000  023737  002616  002566          CMP     INBUF,RBAR      ;CHECK THAT NO TRANSFERS WERE MADE
3241 020006  001412                          BEQ     7$              ;BRANCH AROUND ERROR IF NONE
3242 020010  017737  162502  002570  6$:     MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
3243 020016  012737  177770  002602          MOV     #-10,EWCR       ;MOVE EXPECTED DATA TO EWCR
3244 020024  013737  002616  002600          MOV     INBUF,EBAR      ;MOVE EXPECTED DATA TO EBAR
3245 020032  104041                          ERROR   +41             ;TRANSFERS SHOULD HAVE BEEN INHIBITED
3246 020034  005077  162462          7$:     CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
3247 020040  000414                          BR      TST24           ;BRANCH TO NEXT TEST
3248 020042  062706  000004          8$:     ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
3249 020046  013777  002532  162452          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3250 020054  013777  002534  162446          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3251 020062  017737  162434  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3252 020070  104042                          ERROR   +42             ;DR11 SHOULD NOT HAVE INTERRUPTED A 2ND TIME
```

```
                                .SBTTL  TEST #24 - TEST FOR 200 NPR TRANSFERS IN MAINT MODE
                                ;**********************************************************************
                                ;*TEST 24        TEST FOR 200 NPR TRANSFERS IN MAINT MODE
                                ;*
                                ;*      THIS TEST CHECKS FOR 200 NPR TRANSFERS IN MAINTENANCE MODE.
                                ;*
                                ;**********************************************************************
 020072  032777  004000  161240 TST24:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
 020100  001407                         BEQ     1001$           ;&& BRANCH IF NOT
 020102  104401  020110                 TYPE    ,1000$          ;&& TYPE: "T # 24"
 020106  000404                         BR      1001$           ;&& GET OVER ASCII
 020110     124     040     043 1000$:  .ASCIZ   /T # 24/<CRLF>  ;&& THE ASCII MESSAGE
                                        .EVEN
 020120                         1001$:
 020120  000004                         SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
 020122  012737  020130  001310         MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3259 020130  004737  004036  999$:     JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3260 020134  005077  162362             CLR     @CSR            ;FORCE ACCESS TO CSR
3261 020140  012737  000200  002622     MOV     #200,BUFLEN     ;LENGTH OF BUFFER = 200
3262 020146  013737  002622  002630     MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
3263 020154  005437  002630             NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
3264 020160  004737  003472             JSR     PC,LODBUF       ;LOAD INBUF WITH INCREMENTING PATTERN
3265 020164  004737  003520             JSR     PC,CHKBFF       ;LOAD CHKBUF WITH MODIFIED INCREMENTED PATTERN
3266 020170  013777  002630  162320     MOV     WCLEN,@WCR      ;SET UP WCR
3267 020176  013777  002616  162314     MOV     INBUF,@BAR      ;SET UP BAR
3268 020204  012777  177777  162312     MOV     #-1,@BDR        ;MAINT AIDE
3269 020212  017737  162310  002532     MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3270 020220  017737  162304  002534     MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3271 020226  012777  020330  162272     MOV     #2$,@DRINV      ;INT VECTOR
3272 020234  013777  002540  162266     MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
3273 020242  005037  177776             CLR     PSW             ;LET THE DR11 INTERRUPT
3274 020246  012777  010000  162246     MOV     #MA,@CSR        ;MAINT MODE
3275 020254  012737  001000  002660     MOV     #1000,TIME      ;SET WAIT LOOP COUNTER
3276 020262  052777  000501  162232     BIS     #IE+CY+GO,@CSR  ;IE, CYCLE & GO
3277 020270  005337  002660         1$: DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
3278 020274  001375                     BNE     1$              ;BRANCH BACK IF NOT ZERO
3279 020276  017737  162220  002560     MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3280 020304  017777  002532  162214     MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3281 020312  013777  002534  162210     MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3282 020320  104043                     ERROR   +43             ;EXPECTED INTERRUPT DID NOT OCCUR
3283 020322  005077  162174             CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
3284 020326  000402                     BR      3$              ;BRANCH AROUND THE STACK CLEANUP
3285 020330  062706  000004         2$: ADD     #4,SP           ;CLEAN UP THE STACK AFTER INTERRUPT
3286 020334  013777  002532  162164 3$: MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3287 020342  013777  002534  162160     MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3288 020350  004737  003546             JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
3289 020354  104051                     ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORRECT
3290 020356  005037  002714             CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
3291 020362  004737  003716             JSR     PC,DATCHK       ;CHECK INBUF AFTER A MAINTENANCE MODE OPERATION
3292 020366  104201                     ERROR   +201            ;BUFFER DATA NOT CORRECT
3293 020370  004737  004014             JSR     PC,DATCH2       ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATCHK
```

```
3300                                    .SBTTL  TEST #25 - DOING DATO TO DIODE MEMORY CAUSES NEX
                                        ;**********************************************************************
                                        ;*TEST 25       DOING DATO TO DIODE MEMORY CAUSES NEX
                                        ;*
                                        ;*      THIS TEST INSURES THAT DOING A DATO TO DIODE MEMORY CAUSES THE NEX BIT
                                        ;*      (BIT 14) TO SET.
                                        ;*
                                        ;**********************************************************************
     020374  032777  004000  160736  TST25:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     020402  001407                          BEQ     1001$           ;&& BRANCH IF NOT
     020404  104401  020412                  TYPE    ,1000$          ;&& TYPE: "T # 25"
     020410  000404                          BR      1001$           ;&& GET OVER ASCII
     020412     124     040     043  1000$:  .ASCIZ  /T # 25/<CRLF>  ;&& THE ASCII MESSAGE
                                              .EVEN
     020422                               1001$:
     020422  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     020424  012737  020432  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3301 020432  004737  004036          999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3302 020436  005077  162060                  CLR     @CSR            ;FORCE ACCESS TO CSR
3303 020442  012777  177776  162046          MOV     #-2,@WCR        ;SET UP WCR
3304 020450  013777  002614  162042          MOV     DIOMEM,@BAR     ;SET UP BAR
3305 020456  017737  162044  002532          MOV     @DRINV,$DRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3306 020464  017737  162040  002534          MOV     @DRVS,$DRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3307 020472  012777  020602  162026          MOV     #2$,@DRINV      ;INTERRUPT VECTOR TO 3$
3308 020500  013777  002540  162022          MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
3309 020506  005037  177776                  CLR     PSW             ;LET THE DR11 INTERRUPT
3310 020512  012777  010000  162002          MOV     #MA,@CSR        ;MAINT MODE
3311 020520  052777  000062  161774          BIS     #F1+X6+X7,@CSR  ;SET FNCT1, XBA16, AND XBA17
3312 020526  052777  000501  161766          BIS     #IE+CY+GO,@CSR  ;SET IE, CYCLE, AND GO
3313 020534  012737  001000  002660          MOV     #1000,TIME      ;SET DELAY COUNTER
3314 020542  005337  002660          1$:     DEC     TIME            ;DECREMENT UNTIL ZERO
3315 020546  001375                          BNE     1$              ;BRANCH BACK IF NOT
3316 020550  013777  002532  161750          MOV     $DRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3317 020556  013777  002534  161744          MOV     $DRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3318 020564  017737  161730  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3319 020572  104035                          ERROR   +35             ;DR11 FAILED TO INTERRUPT
3320 020574  005077  161722                  CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
3321 020600  000431                          BR      TST26           ;BRANCH TO THE NEXT TEST
3322 020602  062706  000004          2$:     ADD     #4,SP           ;RESTORE THE STACK
3323 020606  013777  002532  161712          MOV     $DRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3324 020614  013777  002534  161706          MOV     $DRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3325 020622  017737  161674  002560          MOV     @CSR,RCSR       ;MOVE CSR DATA TO RCSR
3326 020630  013701  002560                  MOV     RCSR,R1         ;MOVE DATA TO R1 FOR CHECKING
3327 020634  042701  037577                  BIC     #37577,R1       ;CLEAR ALL BUT ERROR, NEX AND READY BITS
3328 020640  022701  140200                  CMP     #ER+NX+RY,R1    ;SEE IF ALL THESE BITS ARE SET
3329 020644  001407                          BEQ     TST26           ;BRANCH TO THE NEXT TEST IF THEY ARE ALL SET
3330 020646  013737  002560  002572          MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
3331 020654  052737  140200  002572          BIS     #ER+NX+RY,ECSR  ;SET THE BIT THAT SHOULD HAVE BEEN SET
3332 020662  104040                          ERROR   +40             ;CSR IS WRONG
```

```
3339                                    .SBTTL  TEST #26 - CROSSING 32K DOESN'T CAUSE BAOF OR FORCE ERROR
                                        ;*********************************************************************
                                        ;*TEST 26       CROSSING 32K DOESN'T CAUSE BAOF OR FORCE ERROR
                                        ;*
                                        ;*      THIS TEST INSURES THAT CROSSING THE 32K BOUNDARY DOES NOT CAUSE A BAOF
                                        ;*      OR FORCE ERROR.
                                        ;*
                                        ;*********************************************************************
     020664  032777  004000  160446  TST26:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     020672  001407                    BEQ     1001$           ;&& BRANCH IF NOT
     020674  104401  020702            TYPE    ,1000$          ;&& TYPE: "T # 26"
     020700  000404                    BR      1001$           ;&& GET OVER ASCII
     020702     124     040     043  1000$:  .ASCIZ  /T # 26/<CRLF>  ;&& THE ASCII MESSAGE
                                        .EVEN
     020712                          1001$:
     020712  000004                    SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     020714  012737  020722  001310        MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3340 020722  004737  004036        999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3341 020726  012777  177760  161562        MOV     #-20,@WCR       ;SET UP WCR
3342 020734  012777  177776  161556        MOV     #-2,@BAR        ;SET UP BAR FOR PROC STATUS ADDRESS
3343 020742  017737  161560  002532        MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3344 020750  017737  161554  002534        MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3345 020756  012777  021054  161542        MOV     #2$,@DRINV      ;INTERRUPT VECTOR TO 3$
3346 020764  013777  002540  161536        MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
3347 020772  012737  001000  002660        MOV     #1000,TIME      ;SET WAIT LOOP COUNTER
3348 021000  005037  177776                CLR     PSW             ;LET THE DR11 INTERRUPT
3349 021004  012777  010000  161510        MOV     #MA,@CSR        ;MAINT MODE
3350 021012  052777  000563  161502        BIS     #563,@CSR       ;CYCLE, IE, FNCT1, XBA17, XBA16, AND GO TO CSR
3351 021020  005337  002660          1$:   DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
3352 021024  001375                      BNE     1$              ;BRANCH BACK IF NOT ZERO
3353 021026  013777  002532  161472        MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3354 021034  013777  002534  161466        MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3355 021042  017737  161454  002560        MOV     @CSR,RCSR       ;MOVE CSR CONTENTS TO RCSR
3356 021050  104035                        ERROR   +35             ;DR11 FAILED TO INTERRUPT
3357 021052  000433                        BR      TST27           ;BRANCH TO NEXT TEST
3358 021054  062706  000004          2$:   ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
3359 021060  013777  002532  161440        MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3360 021066  013777  002534  161434        MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3361 021074  017737  161422  002560        MOV     @CSR,RCSR       ;MOVE CSR CONTENTS TO RCSR
3362 021102  013701  002560                MOV     RCSR,R1         ;MOVE CONTENTS TO R1 FOR TESTING
3363 021106  042701  037577                BIC     #37577,R1       ;CLEAR ALL BUT THE ERROR AND READY BITS
3364 021112  022701  140200                CMP     #ER+RY+NX,R1    ;SEE IF ERROR, READY AND NEX ARE SET
3365 021116  001411                        BEQ     TST27           ;BRANCH TO NEXT TEST IF THEY ARE
3366 021120  013737  002560  002572        MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
3367 021126  052737  140200  002572        BIS     #ER+RY+NX,ECSR  ;SET THE BITS THAT SHOULD HAVE BEEN SET
3368 021134  104040                        ERROR   +40             ;CSR IS WRONG
3369 021136  005077  161360                CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
```

```
                                .SBTTL  TEST #27 - CHECK ACTUAL POSITION OF 2-N BURST SWITCH
                                ;*****************************************************************
                                ;*TEST 27       CHECK ACTUAL POSITION OF 2-N BURST SWITCH
                                ;*
                                ;*      THIS TEST INSURES THAT THE 2-N BURST SWITCH IS IN THE POSITION THAT
                                ;*      THE DEVICE DESCRIPTOR WORD SAYS IT SHOULD BE.
                                ;*
                                ;*****************************************************************
021142  032777  004000  160170  TST27:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
021150  001407                          BEQ     1001$           ;&& BRANCH IF NOT
021152  104401  021160                  TYPE    ,1000$          ;&& TYPE: "T # 27"
021156  000404                          BR      1001$           ;&& GET OVER ASCII
021160     124     040     043  1000$:  .ASCIZ  /T # 27/<CRLF>  ;&& THE ASCII MESSAGE
                                        .EVEN
021170                          1001$:
021170  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
021172  012737  021210  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3377 021200  032737  000001  002610     BIT     #BIT0,BORW      ;TESTING A "B" OR A "W"?
3378 021206  001456                     BEQ     INOOUT          ;BRANCH TO INOOUT IF DR11-B
3379 021210  004737  004036          999$:  JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3380 021214  012777  100000  161300     MOV     #EIR,@CSR       ;GO TO EIR MODE
3381 021222  017737  161274  002562     MOV     @CSR,REIR       ;MOVE EIR DATA TO REIR
3382 021230  005077  161266             CLR     @CSR            ;GO BACK TO CSR
3383 021234  013701  002562             MOV     REIR,R1         ;MOVE DATA TO R1 ALSO
3384 021240  000301                     SWAB    R1              ;GET BIT 8 INTO BIT 0 BY SWAPPING BYTES
3385 021242  006301                     ASL     R1              ;MOVE BIT 0 INTO BIT 1
3386 021244  042701  177775             BIC     #CBIT1,R1       ;CLEAR ALL BUT BIT 1
3387 021250  013702  002720             MOV     DDW,R2          ;PUT DEVICE DESCRIPTOR WORD IN R2
3388 021254  042702  177775             BIC     #CBIT1,R2       ;CLEAR ALL BUT BIT 1
3389 021260  001402                     BEQ     1$              ;BRANCH IF IT IS CLEAR
3390 021262  005002                     CLR     R2              ;CLEAR THE BIT
3391 021264  000402                     BR      2$              ;GO TEST THE BIT
3392 021266  012702  000002          1$:    MOV     #BIT1,R2        ;SET THE BIT
3393 021272  020102                  2$:    CMP     R1,R2           ;SEE IF RECEIVED MATCHES EXPECTED
3394 021274  001017                     BNE     5$              ;BRANCH TO CHECK FOR LOOP ON TEST
3395 021276  013737  002562  002574     MOV     REIR,EEIR       ;MOVE EXPECTED DATA TO EEIR
3396 021304  032737  000400  002574     BIT     #BIT8,EEIR      ;TEST STATE OF BIT 8
3397 021312  001404                     BEQ     3$              ;BRANCH IF IT IS CLEAR
3398 021314  042737  000400  002574     BIC     #BIT8,EEIR      ;REVERSE STATE - EXPECTED CLEAR
3399 021322  000403                     BR      4$              ;GO CALL ERROR
3400 021324  052737  000400  002574  3$:    BIS     #BIT8,EEIR      ;REVERSE STATE - EXPECTED SET
3401 021332  104054                  4$:    ERROR   +54             ;2-N CYCLE BURST SWITCH IN WRONG POSITION
3402 021334  032777  040400  157776  5$:    BIT     #BIT14+BIT8,@SWR;SEE IF WE SHOULD LOOP ON THIS TEST
3403 021342  001322                     BNE     999$            ;BRANCH BACK IF SO
```

```
3404                                            .SBTTL  CODE TO CHECK CABLE STATUS FOR EXECUTION OF CABLE TESTS
3405                                    ;               CABLE MODE TESTING (WRAP-AROUND CABLE IN USER SLOTS)
3406
3407                                    ;       TESTS 30 THRU 37 ARE PERFORMED IF BIT 2 OF DEVICE DESCRIPTOR WORD IS
3408                                    ;       SET, INDICATING CABLE IS IN.
3409
3410 021344  032737  000004  002720  INOOUT: BIT     #BIT2,DDW       ;SEE IF CABLE IS IN
3411 021352  001002                          BNE     TST30           ;BRANCH TO NEXT TEST IF CABLE IS IN
3412 021354  000137  024360                  JMP     ENDEV           ;JUMP TO ENDEV - TESTS ARE NOT TO BE DONE
```

```
                                        .SBTTL  TEST #30 - CHECK CSR BIT PATTERNS WITH MAINT BIT CLEAR
                                        ;*********************************************************************
                                        ;*TEST 30       CHECK CSR BIT PATTERNS WITH MAINT BIT CLEAR
                                        ;*
                                        ;*      THIS TEST SETS ALL POSSIBLE COMBINATIONS OF SET BITS IN THE CSR WITH
                                        ;*      THE MAINTENANCE BIT CLEAR, AND COMPARES THE RECEIVED CSR CONTENTS WITH
                                        ;*      THAT OF THE EXPECTED PATTERNS IN 'MAICLR'.
                                        ;*
                                        ;*********************************************************************
 021360  032777  004000  157752  TST30:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
 021366  001407                         BEQ     1001$           ;&& BRANCH IF NOT
 021370  104401  021376                 TYPE    ,1000$          ;&& TYPE: "T # 30"
 021374  000404                         BR      1001$           ;&& GET OVER ASCII
 021376     124     040     043  1000$: .ASCIZ  /T # 30/<CRLF>  ;&& THE ASCII MESSAGE
                                        .EVEN
 021406                               1001$:
 021406  000004                         SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
 021410  012737  021444  001310         MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3421 021416  004737  004036             JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3422 021422  005037  002714             CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +202
3423 021426  012700  000002             MOV     #2,R0           ;DO 2 SETS OF 200 PATTERNS
3424 021432  012701  006266             MOV     #MAICLR,R1      ;MOVE ADDRESS OF EXPECTED PATTERNS TO R1
3425 021436  005002                     CLR     R2              ;START WITH PATTERN ZERO
3426 021440  012703  000200         1$: MOV     #200,R3         ;MOVE 200 TO THE LOOP COUNTER
3427 021444  052777  010000  161050 999$: BIS     #MA,@CSR        ;SET MAINTENANCE AND
3428 021452  005077  161044             CLR     @CSR            ;CLEAR TO DO AN INIT
3429 021456  017737  161040  002560     MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3430 021464  022737  000200  002560     CMP     #RY,RCSR        ;MAKE SURE READY BIT IS THE ONLY BIT SET
3431 021472  001404                     BEQ     2$              ;BRANCH IF SO
3432 021474  012737  000200  002572     MOV     #RY,ECSR        ;MOVE EXPECTED DATA TO ECSR
3433 021502  104032                     ERROR   +32             ;READY IS NOT THE ONLY BIT SET
3434 021504  012777  177777  161004  2$: MOV     #-1,@WCR        ;MOVE 1 WORD COUNT TO WCR IN CASE OF IE ENABLED
3435 021512  012777  053700  161000     MOV     #NOCARE,@BAR    ;MOVE A NOT-CARE ADDRESS TO BAR FOR SAME REASON
3436 021520  010277  160776             MOV     R2,@CSR         ;SET THE PARTICULAR FUNCTION BITS IN CSR
3437 021524  017737  160772  002560     MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3438 021532  011137  002572             MOV     (R1),ECSR       ;MOVE EXPECTED DATA TO ECSR
3439 021536  023737  002572  002560     CMP     ECSR,RCSR       ;COMPARE EXPECTED WITH RECEIVED
3440 021544  001427                     BEQ     6$              ;BRANCH IF OK
3441 021546  012737  000401  021556     MOV     #CY+GO,3$       ;REESTABLISH BIT PATTERN
3442 021554  040227                     BIC     R2,(PC)+        ;SEE IF BOTH CYCLE AND GO WERE SET
3443 021556  000401                 3$: .WORD   CY+GO           ;LOCATION TO HOLD BOTH CYCLE AND GO BITS
3444 021560  001016                     BNE     5$              ;BRANCH TO ERROR ONLY IF EITHER OR BOTH BITS WERE CLEAR
3445 021562  005737  002712             TST     MEMGMT          ;SEE IF MEMORY MANAGEMENT IS OUT THERE
3446 021566  001404                     BEQ     4$              ;BRANCH IF NOT
3447 021570  032737  000060  002572     BIT     #X6+X7,ECSR     ;SEE IF EITHER XBA16 OR XBA17 ARE SET
3448 021576  001407                     BEQ     5$              ;BRANCH TO ERROR IF BOTH ARE CLEAR
3449 021600  052737  140000  002572  4$: BIS     #ER+NX,ECSR     ;SET THE ERROR AND NEX BITS - EXPECT THEM TO SET
3450 021606  023737  002572  002560     CMP     ECSR,RCSR       ;NOW SEE IF DATA MATCHES
3451 021614  001403                     BEQ     6$              ;BRANCH AROUND ERROR IF IT DOES
3452 021616  010237  002536         5$: MOV     R2,BUT          ;MOVE THE BITS SET INTO CSR TO THE BUT LOCATION
3453 021622  104202                     ERROR   +202            ;CSR PATTERN NOT CORRECT
3454 021624  062701  000002         6$: ADD     #2,R1           ;INCREMENT R1 TO NEXT EXPECTED PATTERN
3455 021630  005202                     INC     R2              ;INCREMENT THE PATTERN
3456 021632  005303                     DEC     R3              ;DECREMENT THE LOOP COUNTER
3457 021634  001303                     BNE     999$            ;BRANCH BACK IF NOT DONE
3458 021636  062702  000200             ADD     #200,R2         ;ADD 200 TO PATTERN LOCATION
3459 021642  005300                     DEC     R0              ;DECREMENT THE LOOP COUNTER AND
```

```
   3460 021644  001275                    BNE     1$              ;BRANCH BACK IF 2ND OCTAL GROUP NOT DONE
```

```
                                        .SBTTL  TEST #31 - CHECK BAR WITH CSR CLEAR
                                        ;*********************************************************************
                                        ;*TEST 31       CHECK BAR WITH CSR CLEAR
                                        ;*
                                        ;*      THIS TEST CHECKS THAT BAR BIT 0 IS CLEAR WITH CSR CLEAR (CSR=0).
                                        ;*
                                        ;*********************************************************************
021646  032777  004000  157464  TST31:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
021654  001407                          BEQ     1001$           ;&& BRANCH IF NOT
021656  104401  021664                  TYPE    ,1000$          ;&& TYPE: "T # 31"
021662  000404                          BR      1001$           ;&& GET OVER ASCII
021664     124     040     043  1000$:  .ASCIZ  /T # 31/<CRLF>  ;&& THE ASCII MESSAGE
                                        .EVEN
021674                          1001$:
021674  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
021676  012737  021704  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3467 021704  004737  004036  999$:     JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3468 021710  012777  001360  160602    MOV     #$TMP0,@BAR     ;PUT AN ADDRESS IN THE BAR
3469 021716  012777  000001  160576    MOV     #GO,@CSR        ;SET JUST THE GO BIT TO CLEAR THE READY BIT
3470 021724  017737  160572  002560    MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3471 021732  001403                    BEQ     1$              ;BRANCH AROUND ERROR IF EQUAL TO ZERO
3472 021734  005037  002572            CLR     ECSR            ;MOVE EXPECTED DATA TO ECSR
3473 021740  104040                    ERROR   +40             ;CSR IS WRONG
3474 021742  017737  160552  002566  1$:     MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
3475 021750  032737  000001  002566        BIT     #BIT0,RBAR      ;SEE IF THIS BIT IS CLEAR
3476 021756  001407                        BEQ     TST32           ;BRANCH TO NEXT TEST IF IT WAS
3477 021760  013737  002566  002600        MOV     RBAR,EBAR       ;MOVE EXPECTED DATA TO EBAR
3478 021766  042737  000001  002600        BIC     #BIT0,EBAR      ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
3479 021774  104044                        ERROR   +44             ;BAR IS WRONG
```

```
                                        .SBTTL  TEST #32 - TEST 7 SINGLE DATI NON BURST MODE TRANSFERS
                                        ;**********************************************************************
                                        ;*TEST 32       TEST 7 SINGLE DATI NON BURST MODE TRANSFERS
                                        ;*
                                        ;*      THIS TEST DOES 7 BIT PATTERNS OF SINGLE DATI NON BURST MODE TRANSFERS,
                                        ;*      AND THAT THEY ARE DONE PROPERLY.
                                        ;*
                                        ;**********************************************************************
021776  032777  004000  157334  TST32:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
022004  001407                          BEQ     1001$           ;&& BRANCH IF NOT
022006  104401  022014                  TYPE    ,1000$          ;&& TYPE: "T # 32"
022012  000404                          BR      1001$           ;&& GET OVER ASCII
022014     124     040     043  1000$:  .ASCIZ  /T # 32/<CRLF>  ;&& THE ASCII MESSAGE
                                        .EVEN
022024                          1001$:
022024  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
022026  012737  022050  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3487 022034  012702  000007      1$:    MOV     #7,R2           ;SET UP LOOP COUNTER - DO 7 BIT PATTERNS
3488 022040  005037  002714              CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
3489 022044  012703  006250              MOV     #PATRNS,R3      ;MOVE ADDRESS OF PATTERNS TO R3
3490 022050  017737  160452  002532  999$:  MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3491 022056  017737  160446  002534          MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3492 022064  012777  022226  160434          MOV     #3$,@DRINV      ;INTERRUPT VECTOR TO 4$
3493 022072  013777  002540  160430          MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
3494 022100  005037  177776                  CLR     PSW             ;LET THE DR11 INTERRUPT
3495 022104  012777  010000  160410          MOV     #MA,@CSR        ;DO AN INIT BY SETTING AND
3496 022112  005077  160404                  CLR     @CSR            ;CLEARING THE CSR MAINTENANCE BIT
3497 022116  012777  177777  160372          MOV     #-1,@WCR        ;SET UP FOR 1 TRANSFER
3498 022124  012777  002612  160366          MOV     #NPR1,@BAR      ;TRANSFER FROM BUS ADDRESS IN NPR1
3499 022132  005077  160366                  CLR     @BDR            ;GET READY TO RECEIVE DATA
3500 022136  011337  002612                  MOV     (R3),NPR1       ;SET UP TRANSFER DATA
3501 022142  012777  000110  160352          MOV     #F3+IE,@CSR     ;SET THE NON BURST (FNCT3) AND IE
3502 022150  052777  000401  160344          BIS     #CY+GO,@CSR     ;SET THE CYCLE AND GO BITS
3503 022156  005037  002660                  CLR     TIME            ;CLEAR THE TIME LOCATION FOR WAIT LOOP
3504 022162  005237  002660          2$:     INC     TIME            ;INCREMENT UNTIL WE GET TO ZERO AGAIN
3505 022166  001375                          BNE     2$              ;BRANCH BACK IF WE AREN'T THERE YET
3506 022170  013777  002532  160330          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3507 022176  013777  002534  160324          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3508 022204  017737  160312  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3509 022212  011337  002604                  MOV     (R3),ENPR1      ;MOVE PATTERN TO ENPR1
3510 022216  104035                          ERROR   +35             ;DR11 FAILED TO INTERRUPT
3511 022220  005077  160276                  CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
3512 022224  000450                          BR      5$              ;BRANCH TO SEE IF THERE ARE ANY MORE PATTERNS TO CHECK
3513 022226  062706  000004          3$:     ADD     #4,SP           ;CLEAN STACK AFTER INTERRUPT
3514 022232  013777  002532  160266          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3515 022240  013777  002534  160262          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3516 022246  004737  004254                  JSR     PC,ERRCHK       ;CLEAR IE, CHECK FOR ERROR
3517 022252  104021                          ERROR   +21             ;ERROR BIT SHOULD HAVE BEEN CLEAR
3518 022254  017737  160240  002566          MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
3519 022262  012737  002615  002600          MOV     #NPR1+3,EBAR    ;MOVE EXPECTED DATA TO EBAR
3520 022270  017737  160230  002564          MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
3521 022276  011337  002576                  MOV     (R3),EBDR       ;MOVE EXPECTED DATA TO EBDR
3522 022302  017737  160210  002570          MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
3523 022310  005037  002602                  CLR     EWCR            ;MOVE EXPECTED DATA TO EWCR
3524 022314  023737  002566  002600          CMP     RBAR,EBAR       ;COMPARE RECEIVED WITH EXPECTED
3525 022322  001010                          BNE     4$              ;BRANCH IF WRONG
3526 022324  023737  002564  002576          CMP     RBDR,EBDR       ;COMPARE RECEIVED WITH EXPECTED
```

```
3527 022332  001004                        BNE     4$              ;BRANCH IF WRONG
3528 022334  023737  002570  002602        CMP     RWCR,EWCR       ;COMPARE RECEIVED WITH EXPECTED
3529 022342  001401                        BEQ     5$              ;BRANCH IF OK
3530 022344  104211                4$:     ERROR   +211            ;CSR AND-OR WCR AND-OR BAR ARE INCORRECT
3531 022346  062703  000002        5$:     ADD     #2,R3           ;INCREMENT TO NEXT PATTERN
3532 022352  005302                        DEC     R2              ;DECREMENT THE LOOP COUNTER
3533 022354  001235                        BNE     999$            ;BRANCH BACK IF NOT ZERO YET
```

```
3539                                    .SBTTL  TEST #33 - TEST STRING OF 200 DATIS BURST MODE XFERS
                                        ;**********************************************************************
                                        ;*TEST 33       TEST STRING OF 200 DATIS BURST MODE XFERS
                                        ;*
                                        ;*      THIS TEST DOES 200 DATI TRANSFERS IN BURST MODE.
                                        ;*
                                        ;**********************************************************************
     022356  032777  004000  156754  TST33:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     022364  001407                          BEQ     1001$           ;&& BRANCH IF NOT
     022366  104401  022374                  TYPE    ,1000$          ;&& TYPE: "T # 33"
     022372  000404                          BR      1001$           ;&& GET OVER ASCII
     022374     124     040     043  1000$:  .ASCIZ  /T # 33/<CRLF>  ;&& THE ASCII MESSAGE
                                             .EVEN
     022404                          1001$:
     022404  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     022406  012737  022414  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3540 022414  004737  004036          999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3541 022420  012737  000200  002622      .   MOV     #200,BUFLEN     ;LENGTH OF BUFFER=200
3542 022426  004737  003472                  JSR     PC,LODBUF       ;LOAD THE BUFFER WITH INCREMENTING PATTERN
3543 022432  012737  177600  002630          MOV     #-200,WCLEN     ;PREPARE NUMBER FOR WCR
3544 022440  013777  002630  160050          MOV     WCLEN,@WCR      ;SET UP WCR
3545 022446  013777  002616  160044          MOV     INBUF,@BAR      ;SET UP BAR
3546 022454  012777  177777  160042          MOV     #-1,@BDR        ;MAINT AIDE
3547 022462  017737  160040  002532          MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3548 022470  017737  160034  002534          MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3549 022476  012777  022606  160022          MOV     #2$,@DRINV      ;INT VECTOR
3550 022504  013777  002540  160016          MOV     LEVEL,@DRVS     ;INTERRUPT STATUS TO LEVEL OF DEVICE
3551 022512  005037  177776                  CLR     PSW             ;LET THE DR11 INTERRUPT
3552 022516  012777  000100  157776          MOV     #IE,@CSR        ;SET INTERRUPT ENABLE
3553 022524  052777  000401  157770          BIS     #CY+GO,@CSR     ;CYCLE, GO
3554 022532  012737  001000  002660          MOV     #1000,TIME      ;WAIT FOR INTERRUPT
3555 022540  005337  002660          1$:     DEC     TIME            ;DECREMENT UNTIL WE REACH ZERO
3556 022544  001375                          BNE     1$              ;BRANCH BACK IF NOT ZERO
3557 022546  013777  002532  157752          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3558 022554  013777  002534  157746          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3559 022562  017737  157734  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3560 022570  104035                          ERROR   +35             ;DR11 FAILED TO INTERRUPT
3561 022572  012777  010000  157722          MOV     #MA,@CSR         SET THE MAINTENANCE BIT AND
3562 022600  005077  157716                  CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
3563 022604  000426                          BR      TST34           ;BRANCH TO NEXT TEST
3564 022606  062706  000004          2$:     ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
3565 022612  013777  002532  157706          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3566 022620  013777  002534  157702          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3567 022626  004737  003546                  JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
3568 022632  104051                          ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
3569 022634  017737  157664  002564          MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
3570 022642  022737  000177  002564          CMP     #177,RBDR       ;CHECK THAT WORD #200 OF INBUF IS IN BDR
3571 022650  001404                          BEQ     TST34           ;BRANCH TO NEXT TEST IF IT IS
3572 022652  012737  000177  002576          MOV     #177,EBDR       ;MOVE EXPECTED DATA TO EBDR
3573 022660  104045                          ERROR   +45             ;BAD DATA IN BDR
```

```
                                        .SBTTL  TEST #34 - TEST 7 SINGLE DATO NON BURST MODE TRANSFERS
                                        ;*****************************************************************
                                        ;*TEST 34       TEST 7 SINGLE DATO NON BURST MODE TRANSFERS
                                        ;*
                                        ;*      THIS TEST DOES 7 PATTERNS OF SINGLE DATO NON BURST MODE TRANSFERS, AND
                                        ;*      THAT THEY ARE DONE PROPERLY.
                                        ;*
                                        ;*****************************************************************
     022662  032777  004000  156450  TST34:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     022670  001407                           BEQ     1001$           ;&& BRANCH IF NOT
     022672  104401  022700                   TYPE    ,1000$          ;&& TYPE: "T # 34"
     022676  000404                           BR      1001$           ;&& GET OVER ASCII
     022700     124     040     043  1000$:  .ASCIZ  /T # 34/<CRLF>  ;&& THE ASCII MESSAGE
                                              .EVEN
     022710                                  1001$:
     022710  000004                           SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     022712  012737  022734  001310           MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3581 022720  012702  000007          1$:     MOV     #7,R2           ;DO 7 BIT PATTERNS
3582 022724  005037  002714                   CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
3583 022730  012703  006250                   MOV     #PATRNS,R3      ;MOVE ADDRESS OF PATTERNS TO R3
3584 022734  017737  157566  002532  999$:    MOV     @DRINV,$DRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3585 022742  017737  157562  002534           MOV     @DRVS,$DRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3586 022750  012777  023122  157550           MOV     #3$,@DRINV      ;INTERRUPT VECTOR TO 3$
3587 022756  013777  002540  157544           MOV     LEVEL,@DRVS     ;INTERRUPT STATUS TO LEVEL OF DEVICE
3588 022764  005037  177776                   CLR     PSW             ;LET DR11 INTERRUPT
3589 022770  012777  010000  157524           MOV     #MA,@CSR        ;DO AN INIT
3590 022776  005077  157520                   CLR     @CSR            ;FORCE ACCESS TO CSR
3591 023002  012777  177777  157506           MOV     #-1,@WCR        ;SET UP FOR 1 TRANSFER
3592 023010  012777  002612  157502           MOV     #NPR1,@BAR      ;TRANSFER TO BUS ADDRESS IN NPR1
3593 023016  005037  002612                   CLR     NPR1            ;GET READY TO RECEIVE DATA
3594 023022  011377  157476                   MOV     (R3),@BDR       ;SET UP TO TRANSFER DATA
3595 023026  012777  000112  157466           MOV     #F1+F3+IE,@CSR  ;DATO (FNCT1), FNCT3, IE
3596 023034  052777  000401  157460           BIS     #CY+GO,@CSR     ;CYCLE, GO
3597 023042  012737  001000  002660           MOV     #1000,TIME      ;CLEAR THE TIME LOCATION FOR WAIT LOOP
3598 023050  005337  002660          2$:     DEC     TIME            ;DECREMENT UNTIL WE GET BACK TO ZERO
3599 023054  001375                           BNE     2$              ;BRANCH BACK IF NOT ZERO
3600 023056  013777  002532  157442           MOV     $DRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3601 023064  013777  002534  157436           MOV     $DRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3602 023072  017737  157424  002560           MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3603 023100  011337  001360                   MOV     (R3),$TMP0      ;MOVE PATTERN TO $TMP0
3604 023104  104035                           ERROR   +35             ;DR11 FAILED TO INTERRUPT
3605 023106  012777  010000  157406           MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
3606 023114  005077  157402                   CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
3607 023120  000445                           BR      6$              ;BRANCH TO DO NEXT PATTERN
3608 023122  062706  000004          3$:     ADD     #4,SP           ;CLEAN UP STACK FROM INTERRUPT
3609 023126  013777  002532  157372           MOV     $DRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3610 023134  013777  002534  157366           MOV     $DRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3611 023142  004737  004254                   JSR     PC,ERRCHK       ;CLEAR IE, CHECK FOR ERROR
3612 023146  104021                           ERROR   +21             ;ERROR BIT SHOULD HAVE BEEN CLEAR
3613 023150  017737  157342  002570           MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
3614 023156  001403                           BEQ     4$              ;BRANCH IF IT IS EQUAL TO ZERO
3615 023160  011337  001362                   MOV     (R3),$TMP1      ;MOVE PATTERN TO $TMP0
3616 023164  104206                           ERROR   +206            ;WCR NOT EQUAL TO ZERO
3617 023166  017737  157326  002566  4$:     MOV     @BAR,RBAR       ;MOVE RECEIVED ADDRESS TO RBAR
3618 023174  022737  002615  002566           CMP     #NPR1+3,RBAR    ;COMPARE CORRECT BAR WITH BAR CABLE MODE TESTING LEAVES
3619                                                                  ;BIT 0 OF BAR SET. THEREFORE MUST CHECK FOR ODD ADDRESS
3620 023202  001406                           BEQ     5$              ;BRANCH IF IT IS OK
```

```
 3621 023204  012737  002615  002600          MOV     #NPR1+3,EBAR    ;MOVE EXPECTED DATA TO EBAR
 3622 023212  011337  001362                  MOV     (R3),$TMP1      ;MOVE PATTERN TO $TMP0
 3623 023216  104207                          ERROR   +207            ;BAR IS WRONG
 3624 023220  021337  002612          5$:     CMP     (R3),NPR1       ;CHECK FOR CORRECT DATA
 3625 023224  001403                          BEQ     6$              ;BRANCH IF CORRECT DATA WAS TRANSFERRED
 3626 023226  011337  002604                  MOV     (R3),ENPR1      ;MOVE EXPECTED DATA TO ENPR1
 3627 023232  104210                          ERROR   +210            ;DATA NOT TRANSFERED CORRECTLY
 3628 023234  062703  000002          6$:     ADD     #2,R3           ;POINT TO NEXT BIT PATTERN
 3629 023240  005302                          DEC     R2              ;COUNT 1 PATTERN DONE
 3630 023242  001234                          BNE     999$            ;BRANCH BACK IF NOT DONE
```

```
3636                                    .SBTTL  TEST #35 - TEST STRING OR 200 DATOS BURST MODE XFERS
                                        ;*****************************************************************
                                        ;*TEST 35       TEST STRING OR 200 DATOS BURST MODE XFERS
                                        ;*
                                        ;*      THIS TEST CHECKS 200 DATO TRANSFERS IN BURST MODE.
                                        ;*
                                        ;*****************************************************************
     023244  032777  004000  156066  TST35:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     023252  001407                          BEQ     1001$           ;&& BRANCH IF NOT
     023254  104401  023262                  TYPE    ,1000$          ;&& TYPE: "T # 35"
     023260  000404                          BR      1001$           ;&& GET OVER ASCII
     023262     124     040     043  1000$:  .ASCIZ  /T # 35/<CRLF>  ;&& THE ASCII MESSAGE
                                             .EVEN
     023272                          1001$:
     023272  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     023274  012737  023302  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3637 023302  005077  157214          999$:   CLR     @CSR            ;FORCE ACCESS TO CSR
3638 023306  012737  000200  002622          MOV     #200,BUFLEN     ;LENGTH OF BUFFER=200
3639 023314  004737  003472                  JSR     PC,LODBUF       ;LOAD THE BUFFER WITH INCREMENTING PATTERN
3640 023320  013737  002622  002630          MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
3641 023326  005437  002630                  NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
3642 023332  013777  002630  157156          MOV     WCLEN,@WCR      ;SET UP WCR
3643 023340  013777  002616  157152          MOV     INBUF,@BAR      ;SET UP BAR
3644 023346  012777  052525  157150          MOV     #52525,@BDR     ;SET UP BDR
3645 023354  017737  157146  002532          MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3646 023362  017737  157142  002534          MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3647 023370  012777  023500  157130          MOV     #2$,@DRINV      ;INTERRUPT VECTOR
3648 023376  013777  002540  157124          MOV     LEVEL,@DRVS     ;INTERRUPT STATUS TO LEVEL OF DEVICE
3649 023404  005037  177776                  CLR     PSW             ;LET THE DR11 INTERRUPT
3650 023410  012777  000102  157104          MOV     #IE+F1,@CSR     ;IE, FNCT1
3651 023416  052777  000401  157076          BIS     #CY+GO,@CSR     ;CYCLE, GO
3652 023424  012737  001000  002660          MOV     #1000,TIME      ;MOVE WAIT LOOP VALUE TO TIME LOCATION
3653 023432  005337  002660          1$:     DEC     TIME            ;DECREMENT UNTIL WE REACH ZERO
3654 023436  001375                          BNE     1$              ;BRANCH BACK IF NOT ZERO
3655 023440  013777  002532  157060          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3656 023446  013777  002534  157054          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3657 023454  017737  157042  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3658 023462  104035                          ERROR   +35             ;DR11 FAILED TO INTERRUPT
3659 023464  012777  010000  157030          MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
3660 023472  005077  157024                  CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
3661 023476  000421                          BR      TST36           ;BRANCH TO NEXT TEST
3662 023500  062706  000004          2$:     ADD     #4,SP           ;CLEAN STACK AFTER INTERRUPT
3663 023504  013777  002532  157014          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3664 023512  013777  002534  157010          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3665 023520  004737  003546                  JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
3666 023524  104051                          ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
3667 023526  004737  004106                  JSR     PC,DATOCK       ;CHECK INBUF
3668 023532  104046                          ERROR   +46             ;BUFFER DATA NOT CORRECT
3669 023534  004737  004170                  JSR     PC,DATOC2       ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATOCK
3670 023540  104047                          ERROR   +47             ;TOO MANY WORDS WERE TRANSFERED
```

```
3676                                    .SBTTL  TEST #36 - TEST STRING OF 200 DATIS NON-BURST MODE
                                        ;*********************************************************************
                                        ;*TEST 36       TEST STRING OF 200 DATIS NON-BURST MODE
                                        ;*
                                        ;*      THIS TEST DOES 200 DATI TRANSFERS IN NON BURST MODE.
                                        ;*
                                        ;*********************************************************************
     023542  032777  004000  155570  TST36:  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     023550  001407                          BEQ     1001$           ;&& BRANCH IF NOT
     023552  104401  023560                  TYPE    ,1000$          ;&& TYPE: "T # 36"
     023556  000404                          BR      1001$           ;&& GET OVER ASCII
     023560     124     040     043  1000$:  .ASCIZ   /T # 36/<CRLF> ;&& THE ASCII MESSAGE
                                             .EVEN
     023570                               1001$:
     023570  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     023572  012737  023600  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3677 023600  004737  004036          999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3678 023604  005077  156712                  CLR     @CSR            ;FORCE ACCESS TO CSR
3679 023610  012737  000200  002622          MOV     #200,BUFLEN     ;LENGTH OF BUFFER=200
3680 023616  004737  003472                  JSR     PC,LODBUF       ;LOAD THE BUFFER WITH INCREMENTING PATTERN
3681 023622  013737  002622  002630          MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
3682 023630  005437  002630                  NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
3683 023634  013777  002630  156654          MOV     WCLEN,@WCR      ;SET-UP WCR
3684 023642  013777  002616  156650          MOV     INBUF,@BAR      ;SET-UP BAR
3685 023650  012777  177777  156646          MOV     #-1,@BDR        ;MAINT AIDE
3686 023656  017737  156644  002532          MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3687 023664  017737  156640  002534          MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3688 023672  012777  024002  156626          MOV     #2$,@DRINV      ;INT VECTOR
3689 023700  013777  002540  156622          MOV     LEVEL,@DRVS     ;INTERRUPT STATUS TO LEVEL OF DEVICE
3690 023706  005037  177776                  CLR     PSW             ;LET THE DR11 INTERRUPT
3691 023712  012777  000110  156602          MOV     #F3+IE,@CSR     ;FNCT3, IE
3692 023720  052777  000401  156574          BIS     #CY+GO,@CSR     ;CYCLE, GO
3693 023726  012737  001000  002660          MOV     #1000,TIME      ;SET WAIT LOOP COUNTER
3694 023734  005337  002660          1$:     DEC     TIME            ;DECREMENT UNTIL WE REACH ZERO
3695 023740  001375                          BNE     1$              ;BRANCH BACK IF NOT ZERO
3696 023742  013777  002532  156556          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3697 023750  013777  002534  156552          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3698 023756  017737  156540  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3699 023764  104035                          ERROR   +35             ;DR11 FAILED TO INTERRUPT
3700 023766  012777  010000  156526          MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
3701 023774  005077  156522                  CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
3702 024000  000426                          BR      TST37           ;BRANCH TO NEXT TEST
3703 024002  062706  000004          2$:     ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
3704 024006  013777  002532  156512          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3705 024014  013777  002534  156506          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3706 024022  004737  003546                  JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
3707 024026  104051                          ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
3708 024030  017737  156470  002564          MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
3709 024036  022737  000177  002564          CMP     #177,RBDR       ;CHECK THAT WORD #200 OF INBUF IS IN BDR
3710 024044  001404                          BEQ     TST37           ;BRANCH TO NEXT TEST IF OK
3711 024046  012737  000177  002576          MOV     #177,EBDR       ;MOVE EXPECTED DATA TO EBDR
3712 024054  104045                          ERROR   +45             ;BAD DATA IN BDR
```

```
                                        .SBTTL  TEST #37 - TEST STRING OF 200 DATOS NON-BURST MODE
                                        ;*****************************************************************
                                        ;*TEST 37       TEST STRING OF 200 DATOS NON-BURST MODE
                                        ;*
                                        ;*      THIS TEST DOES 200 DATOS IN NON BURST MODE.
                                        ;*
                                        ;*****************************************************************
 024056  032777  004000  155254  TST37: BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
 024064  001407                        BEQ     1001$           ;&& BRANCH IF NOT
 024066  104401  024074                TYPE    ,1000$          ;&& TYPE: "T # 37"
 024072  000404                        BR      1001$           ;&& GET OVER ASCII
 024074     124     040     043  1000$: .ASCIZ  /T # 37/<CRLF>  ;&& THE ASCII MESSAGE
                                        .EVEN
 024104                         1001$:
 024104  000004                        SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
 024106  012737  024114  001310        MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
3719 024114  004737  004036    999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3720 024120  005077  156376            CLR     @CSR            ;FORCE ACCESS TO CSR
3721 024124  012737  000200  002622    MOV     #200,BUFLEN     ;LENGTH OF BUFFER=200
3722 024132  004737  003472            JSR     PC,LODBUF       ;LOAD THE BUFFER WITH INCREMENTING PATTERN
3723 024136  013737  002622  002630    MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
3724 024144  005437  002630            NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
3725 024150  013777  002630  156340    MOV     WCLEN,@WCR      ;SET UP WCR
3726 024156  013777  002616  156334    MOV     INBUF,@BAR      ;SET UP BAR
3727 024164  012777  052525  156332    MOV     #52525,@BDR     ;SET UP BDR
3728 024172  017737  156330  002532    MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3729 024200  017737  156324  002534    MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3730 024206  012777  024316  156312    MOV     #2$,@DRINV      ;INTERRUPT VECTOR
3731 024214  013777  002540  156306    MOV     LEVEL,@DRVS     ;INTERRUPT STATUS TO LEVEL OF DEVICE
3732 024222  005037  177776            CLR     PSW             ;LET THE DR11 INTERRUPT
3733 024226  012777  000112  156266    MOV     #F1+F3+IE,@CSR  ;FNCT1, FNCT3, IE
3734 024234  052777  000401  156260    BIS     #CY+GO,@CSR     ;CYCLE, GO
3735 024242  012737  001000  002660    MOV     #1000,TIME      ;SET WAIT LOOP COUNTER
3736 024250  005337  002660      1$:   DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
3737 024254  001375                    BNE     1$              ;BRANCH BACK IF NOT ZERO
3738 024256  013777  002532  156242    MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3739 024264  013777  002534  156236    MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3740 024272  017737  156224  002560    MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
3741 024300  104035                    ERROR   +35             ;DR11 FAILED TO INTERRUPT
3742 024302  012777  010000  156212    MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
3743 024310  005077  156206            CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
3744 024314  000421                    BR      TST40           ;BRANCH TO NEXT TEST
3745 024316  062706  000004      2$:   ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
3746 024322  013777  002532  156176    MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3747 024330  013777  002534  156172    MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
3748 024336  004737  003546            JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
3749 024342  104051                    ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
3750 024344  004737  004106            JSR     PC,DATOCK       ;CHECK INBUF
3751 024350  104046                    ERROR   +46             ;BUFFER DATA NOT CORRECT
3752 024352  004737  004170            JSR     PC,DATOC2       ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATOCK
3753 024356  104047                    ERROR   +47             ;TOO MANY WORDS WERE TRANSFERED
```

```
                                        .SBTTL  END OF DEVICE PASS ROUTINE
                                        ;*************************************************************************
                                ENDEV:  SCOPE                           ;FOR POSSIBLE LOOP ON TEST
                                        CLR     $TSTNM                  ;CLEAR TEST NO. COUNT FOR SCOPE ROUTINE
                                        CMP     #1,QTYBRD               ;IS THERE MORE THAN 1 BOARD UNDER TEST?
                                        BEQ     5$                      ;BRANCH IF NO
                                        TST     $ERTTL                  ;SEE IF THERE WERE ANY ERRORS
                                        BNE     2$                      ;BRANCH AROUND EOP TEST IF SO
                                        TSTB    EOPLOC                  ;SEE IF EOP MESSAGES ARE TO PRINT
                                        BEQ     3$                      ;BRANCH IF SO
                                        TSTB    EOPLOC+1                ;SEE IF EOP WAS REQUESTED BEFORE
                                        BNE     1$                      ;BRANCH TO PRINT MESSAGE IF SO
                                        TSTB    CHARCT                  ;SEE IF EOD REQUESTED
                                        BEQ     5$                      ;BRANCH AROUND MESSAGE PRINT IF NOT
                                1$:     INCB    EOPLOC+1                ;INCREMENT UPPER BYTE OF EOPLOC TO CALL EOP MESSAGE
                                        BR      3$                      ;BRANCH IF NOT - NO ERRORS TO SPACE FROM
                                2$:     TYPE    ,$CRLF                  ;TYPE A <CRLF>
                                3$:     TYPE    ,BOARD                  ;TYPE:  'BOARD # '
                                        MOV     $UNIT,-(SP)             ;,SAVE $UNIT FOR TYPEOUT
                                        TYPDS                           ;;GO TYPE--DECIMAL ASCII WITH SIGN
                                        TYPE    ,TSTCOM                 ;TYPE:  ' TESTING COMPLETE'
                                        TST     $ERTTL                  ;SEE IF ANY ERRORS THIS DEVICE
                                        BEQ     4$                      ;BRANCH AROUND TOTAL ERRORS MESSAGE IF NONE
                                        TYPE    ,ETDEV                  ;TYPE:  ' - TOTAL ERRORS THIS DEVICE = '
                                        MOV     $ERTTL,-(SP)            ;MOVE NUMBER OF ERRORS TO THE STACK
                                        TYPDS                           ;GO TYPE THE NUMBER
                                        JSR     PC,ERCAPT               ;GO LOG THE UNIT #, PASS # & # OF ERRORS THIS DEVICE
                                4$:     TYPE    ,$CRLF                  ;TYPE A <CRLF>
                                5$:     INC     $DEVCT                  ;INCREMENT DEVICE COUNTER
                                        CMP     QTYBRD,$DEVCT           ;ALL DEVICES TESTED?
                                        BEQ     $EOP                    ;GO TO END OF PASS ROUTINE IF SO
                                        INC     $UNIT                   ;INCREMENT THE UNIT NUMBER
                                        JMP     TSTDEV                  ;GO TEST NEXT DEVICE
```

```
3786                                    .SBTTL  END OF PASS ROUTINE
                                        ;*********************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO GOAGIN
024532                                  $EOP:
024532  000004                                  SCOPE
024534  005037  001302                         CLR     $TSTNM          ;;ZERO THE TEST NUMBER
024540  005237  001416                         INC     $PASS           ;;INCREMENT THE PASS NUMBER
024544  100004                                 BPL     10$             ;&& BRANCH IF STILL POSITIVE
024546  005037  001416                         CLR     $PASS           ;&& CLEAR THE PASS LOCATION AND
024552  005237  002716                         INC     $PASS2          ;&& INCREMENT $PASS2 TO SHOW 1 OVERFLOW
024556  005327                          10$:    DEC     (PC)+           ;;LOOP?
024560  000001                          $EOPCT: .WORD   1
024562  003076                                 BGT     $DOAGN          ;;YES
024564  012737                                 MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
024566  000001                          $ENDCT: .WORD   1
024570  024560                                 $EOPCT
024572  105737  002706                         TSTB    EOPLOC          ;&& SEE IF EOP MESSAGES ARE TO PRINT
024576  001420                                 BEQ     1$              ;&& BRANCH IF THEY ARE
024600  005737  001312                         TST     $ERTTL          ;&& SEE IF ANY ERRORS THIS PASS
024604  001020                                 BNE     2$              ;&& BRANCH IF THERE ARE TO PRINT EOP
024606  105737  002707                         TSTB    EOPLOC+1        ;&& SEE IF ERROR OCCURED IN ANOTHER DEVICE
024612  001012                                 BNE     1$              ;&& BRANCH IF ANY TO PRINT EOP
024614  105737  030707                         TSTB    CHARCT          ;&& SEE IF ANY CHARACTER WAS INPUTED
024620  001447                                 BEQ     $GET42          ;&& BRANCH IF NOT
024622  122737  000007  030707                 CMPB    #7,CHARCT       ;&& SEE IF OTHER THAT (^G) TYPED, REQUESTING EOP
024630  001443                                 BEQ     $GET42          ;&& BRANCH IF A (^G) - THIS NOT MEANT FOR EOP ROUTINE
024632  105037  030707                         CLRB    CHARCT          ;&& CLEAR THE LOCATION
024636  000405                                 BR      3$              ;&& GET OVER $ERTTL TEST AND <CRLF> PRINT
024640  005737  001312                  1$:    TST     $ERTTL          ;&& SEE IF AN EXTRA <CRLF> NEEDS PRINTING
024644  001402                                 BEQ     3$              ;&& BRANCH IF NOT
024646  104401  001405                  2$:    TYPE    ,$CRLF          ;&& TYPE A <CRLF> TO SPACE EOP FROM ERROR
024652  105037  002707                  3$:    CLRB    EOPLOC+1        ;&& CLEAR THE UPPER BYTE OF EOPLOC
024656  104401  024767                         TYPE    ,$ENDMG         ;;TYPE "END PASS #"
024662  013746  002716                         MOV     $PASS2,-(SP)    ;&& MOVE OVERFLOW TO STACK FOR PRINTING
024666  013746  001416                         MOV     $PASS,-(SP)     ;;MOVE $PASS TO THE STACK FOR PRINTING
024672  104406                                 TYPDE                   ;&& TYPE THE NUMBER IN EXTENDED DECIMAL
024674  104401  024764                         TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
024700  005737  001312                         TST     $ERTTL          ;&& SEE IF ANY ERRORS THIS PASS
024704  001413                                 BEQ     4$              ;&& BRANCH AROUND NUMBER OF ERRORS MESSAGE IF NONE
024706  022737  000001  002414                 CMP     #1,QTYBRD       ;&& SEE IF ONLY 1 BOARD IS BEING TESTED
024714  001007                                 BNE     4$              ;&& BRANCH IF NOT - 'TOTAL ERRORS' IS MEANINGLESS
024716  104401  030710                         TYPE    ,TESLR          ;&& TYPE:   '  TOTAL ERRORS SINCE LAST REPORT '
024722  013746  001312                         MOV     $ERTTL,-(SP)    ;&& PUT $ERTTL ON STACK FOR PRINTING
024726  104405                                 TYPDS                   ;&& TYPE THE NUMBER OF ERRORS IN DECIMAL
024730  004737  003126                         JSR     PC,ERCAPT       ;&& GO CAPTURE THE DEVICE, PASS & # OF ERRORS
024734  104401  001405                  4$:    TYPE    ,$CRLF          ;&& TYPE A <CRLF>
024740  013700  000042                  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
024744  001405                                 BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
024746  000005                                 RESET                   ;;CLEAR THE WORLD
024750  004710                          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
024752  000240                                 NOP                     ;;SAVE ROOM
024754  000240                                 NOP                     ;;FOR
024756  000240                                 NOP                     ;;ACT11
024760                                  $DOAGN:
```

```
        024760  000137                          JMP     @(PC)+          ;;RETURN
        024762  025002                  $RTNAD: .WORD   GOAGIN
        024764     377      377     000 $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
        024767     105      116     104 $ENDMG: .ASCIZ  /END PASS #/
   3787 025002  005037  001420          GOAGIN: CLR     $DEVCT          ;CLEAR DEVICE COUNT
   3788 025006  022737  000001  002414          CMP     #1,QTYBRD       ;IS THERE ONLY ONE DEVICE UNDER TEST?
   3789 025014  001002                          BNE     RSTRT           ;BR, IF NOT
   3790 025016  000137  011206                  JMP     REINIT          ;GO DO ANOTHER PASS
   3791 025022  005037  001422          RSTRT:  CLR     $UNIT           ;CLEAR UNIT NUMBER
   3792 025026  000137  011006                  JMP     BEGIN1          ;GO BEGIN TEST OF NEXT DEVICE
```

```
 3793                                   .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
                                        ;*
 025032  105737  001357         $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
 025036  100002                         BPL     1$              ;;BR IF YES
 025040  000000                         HALT                    ;;HALT HERE IF NO TERMINAL
 025042  000430                         BR      3$              ;;LEAVE
 025044  010046                 1$:     MOV     R0,-(SP)        ;;SAVE R0
 025046  017600  000002                 MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
 025052  122737  000001  001430         CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
 025060  001011                         BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
 025062  132737  000100  001431         BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
 025070  001405                         BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
 025072  010037  025102                 MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
 025076  004737  031534                 JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
 025102  000000                 61$:    .WORD   0               ;;MESSAGE ADDRESS
 025104  132737  000040  001431 62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
 025112  001003                         BNE     60$             ;;YES,SKIP TYPE OUT
 025114  112046                 2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
 025116  001005                         BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
 025120  005726                         TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
 025122  012600                 60$:    MOV     (SP)+,R0        ;;RESTORE R0
 025124  062716  000002         3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
 025130  000002                         RTI                     ;;RETURN
 025132  122716  000011         4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
 025136  001430                         BEQ     8$
 025140  122716  000200                 CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
 025144  001006                         BNE     5$
 025146  005726                         TST     (SP)+           ;;POP  <CR><LF> EQUIV
 025150  104401                         TYPE                    ;;TYPE A CR AND LF
 025152  001405                         $CRLF
 025154  105037  025374                 CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
 025160  000755                         BR      2$              ;;GET NEXT CHARACTER
 025162  004737  025244         5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
 025166  123726  001356         6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
 025172  001350                         BNE     2$              ;;IF NO GO GET NEXT CHAR.
 025174  013746  001354                 MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                 ;;AND THE NULL CHAR.
 025200  105366  000001         7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
 025204  002770                         BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
 025206  004737  025244                 JSR     PC,$TYPEC       ;;GO TYPE A NULL
 025212  105337  025374                 DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
 025216  000770                         BR      7$              ;;LOOP
                                        ;HORIZONTAL TAB PROCESSOR
 025220  112716  000040         8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
```

```
025224  004737  025244          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
025230  132737  000007  025374          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
025236  001372                          BNE     9$              ;;TAB STOP
025240  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
025242  000724                          BR      2$              ;;GET NEXT CHARACTER
025244  105777  154100          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
025250  100375                          BPL     $TYPEC
025252  116677  000002  154072          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
025260  105777  154060                  TSTB    @$TKS           ;;SEE IF KEYBOARD IS TALKING.
025264  100027                          BPL     2$              ;;BRANCH IF IT ISN'T.
025266  117737  154054  030707          MOVB    @$TKB,CHARCT    ;&& PUT CHARACTER IN CHARCT
025274  142737  000200  030707          BICB    #200,CHARCT     ;&& BIT CLEAR PARITY BIT.
025302  122737  000023  030707          CMPB    #23,CHARCT      ;&& SEE IF THIS IS A ^S.
025310  001015                          BNE     2$              ;;BRANCH TO CONTINUE IF IT ISN'T.
025312  105777  154026          3$:     TSTB    @$TKS           ;;WAIT FOR ANOTHER INPUT.
025316  100375                          BPL     3$              ;;BRANCH BACK IF NOT READY.
025320  117737  154022  030707          MOVB    @$TKB,CHARCT    ;&& PUT CHARACTER IN CHARCT
025326  142737  000200  030707          BICB    #200,CHARCT     ;&& BIT CLEAR PARITY BIT.
025334  122737  000021  030707          CMPB    #21,CHARCT      ;&& SEE IF THIS IS A ^Q.
025342  001363                          BNE     3$              ;;BRANCH BACK FOR MORE WAIT IF NOT.
025344  122766  000015  000002  2$:     CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
025352  001003                          BNE     1$              ;;BRANCH IF NO
025354  105037  025374                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
025360  000406                          BR      $TYPEX          ;;EXIT
025362  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
025370  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
025372  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
025374  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
025376  000207          $TYPEX: RTS     PC
```

```
3794                                    .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
                                        ;**************************************************************
                                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                        ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                        ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                        ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                        ;*      .BYTE   M               ;;M=1 OR 0
                                        ;*                                      ;;1=TYPE LEADING ZEROS
                                        ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                        ;*
                                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                        ;*$TYPOS OR $TYPOC
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT
025400  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
025404  116637  000001  025623          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
025412  112637  025625                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
025416  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
025422  000406                          BR      $TYPON
025424  112737  000001  025623  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
025432  112737  000006  025625          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
025440  112737  000005  025622  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
025446  010346                          MOV     R3,-(SP)        ;;SAVE R3
025450  010446                          MOV     R4,-(SP)        ;;SAVE R4
025452  010546                          MOV     R5,-(SP)        ;;SAVE R5
025454  113704  025625                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
025460  005404                          NEG     R4
025462  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
025466  110437  025624                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
025472  113704  025623                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
025476  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
025502  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
025504  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
025506  000404                          BR      3$              ;;GO DO MSB
025510  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
025512  006105                          ROL     R5
025514  006105                          ROL     R5
025516  010503                          MOV     R5,R3
025520  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
025522  105337  025624                  DECB    $OMODE          ;;TYPE THIS DIGIT?
025526  100016                          BPL     7$              ;;BR IF NO
025530  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
025534  001002                          BNE     4$              ;;TEST FOR 0
025536  005704                          TST     R4              ;;SUPPRESS THIS 0?
025540  001403                          BEQ     5$              ;;BR IF YES
025542  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
025544  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
025550  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
```

```
025554  110337  025620                  MOVB    R3,8$           ;;SAVE FOR TYPING
025560  104401  025620                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
025564  105337  025622          7$:     DECB    $OCNT           ;;COUNT BY 1
025570  003347                          BGT     2$              ;;BR IF MORE TO DO
025572  002402                          BLT     6$              ;;BR IF DONE
025574  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
025576  000744                          BR      2$              ;;GO DO THE LAST DIGIT
025600  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
025602  012604                          MOV     (SP)+,R4        ;;RESTORE R4
025604  012603                          MOV     (SP)+,R3        ;;RESTORE R3
025606  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
025614  012616                          MOV     (SP)+,(SP)
025616  000002                          RTI                     ;;RETURN
025620     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
025621     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
025622     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
025623     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
025624  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
3795                                    .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
                                        ;**********************************************************************
                                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
                                        ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
                                        ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
                                        ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
                                        ;*REPLACED WITH SPACES.
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
                                        ;*      TYPDS                   ;;GO TO THE ROUTINE
                                        ;*
                                        ;&& IF YOU SHOULD HAVE A NUMBER GREATER THAN 32767 TO PRINT, LOAD THE
                                        ;&& MULTIPLE OF 32768 ON THE STACK, THEN THE REMAINDER AS YOU WOULD
                                        ;&& ABOVE.  FOR INSTANCE, WHEN INCREMENTING A COUNTER TO BE PRINTED,
                                        ;&& TEST FOR NEGATIVE.  IF NEGATIVE, CLEAR THE LOCATION AND INCREMENT
                                        ;&& A SPECIAL OVERFLOW LOCATION.
                                        ;&& CALL:
                                        ;&&     MOV     OVFNUM,-(SP)    ;&& PUT OVERFLOW NUMBER ON STACK
                                        ;&&     MOV     NUM,-(SP)       ;&& PUT REMAINING NUMBER ON STACK
                                        ;&&     TYPDE                   ;&& GO TO THE ROUTINE
025626  012737  000001  001364  $TYPDE: MOV     #1,$TMP2        ;&& SHOW ENTRY AT $TYPDE
025634  000402                  BR      $TYPD           ;&& GO TO ROUTINE
025636  005037  001364          $TYPDS: CLR     $TMP2           ;&& SHOW ENTRY AT $TYPDS
025642                          $TYPD:
025642  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
025644  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
025646  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
025650  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
025652  010546                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
025654  012746  020200          MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
025660  016605  000020          MOV     20(SP),R5       ;;GET THE INPUT NUMBER
025664  100004                  BPL     1$              ;;BR IF INPUT IS POS.
025666  005405                  NEG     R5              ;;MAKE THE BINARY NUMBER POS.
025670  112766  000055  000001  MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
025676  005000          1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
025700  012703  026142          MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
025704  112723  000040          MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
025710  005002          2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
025712  016001  026132          MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
025716  160105          3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
025720  002402                  BLT     4$              ;;BR IF DONE
025722  005202                  INC     R2              ;;INCREASE THE BCD DIGIT BY 1
025724  000774                  BR      3$
025726  060105          4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
025730  005702                  TST     R2              ;;CHECK IF BCD DIGIT=0
025732  001002                  BNE     5$              ;;FALL THROUGH IF 0
025734  105716                  TSTB    (SP)            ;;STILL DOING LEADING 0'S?
025736  100407                  BMI     7$              ;;BR IF YES
025740  106316          5$:     ASLB    (SP)            ;;MSD?
025742  103003                  BCC     6$              ;;BR IF NO
025744  116663  000001  177777  MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
025752  052702  000060  6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
025756  052702  000040  7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
025762  110223                  MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
025764  005720                  TST     (R0)+           ;;JUST INCREMENTING
025766  020027  000010          CMP     R0,#10          ;;CHECK THE TABLE INDEX
025772  002746                  BLT     2$              ;;GO DO THE NEXT DIGIT
```

```
025774  003002                          BGT     8$              ;;GO TO EXIT
025776  010502                          MOV     R5,R2           ;;GET THE LSD
026000  000764                          BR      6$              ;;GO CHANGE TO ASCII
026002  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
026004  100003                          BPL     9$              ;;BR IF NO
026006  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
026014  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
026016  005737  001364                  TST     $TMP2           ;&& WAS ENTRY AT $TYPDS?
026022  001405                          BEQ     10$             ;&& BRANCH IF SO
026024  005766  000020                  TST     20(SP)          ;&& TEST THE OVERFLOW LOCATION
026030  001402                          BEQ     10$             ;&& BRANCH IF NON-ZERO
026032  004737  026154                  JSR     PC,EXPAND       ;&& GO EXPAND THE DECIMAL NUMBER
026036                          10$:
026036  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
026040  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
026042  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
026044  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
026046  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
026050  104401  026142                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
026054  005737  001364                  TST     $TMP2           ;&& SEE IF ENTRY WAS AT $TYPDS
026060  001417                          BEQ     11$             ;&& BRANCH IF SO
026062  016666  000002  000006          MOV     2(SP),6(SP)     ;&& ADJUST THE STACK
026070  012666  000002                  MOV     (SP)+,2(SP)     ;&&
026074  005726                          TST     (SP)+           ;&&
026076  105037  026151                  CLRB    $DBLK+7         ;&& REPLACE ORIGINAL TERMINATOR
026102  112737  000040  026150          MOVB    #' ,$DBLK+6     ;&& REPLACE ORIGINAL SPACE CHARACTER
026110  112737  000040  026142          MOVB    #' ,$DBLK       ;&& REPLACE ORIGINAL SPACE CHARACTER
026116  000002                          RTI                     ;&& RETURN TO USER
026120  016666  000002  000004  11$:     MOV     2(SP),4(SP)     ;;ADJUST THE STACK
026126  012616                          MOV     (SP)+,(SP)
026130  000002                          RTI                     ;;RETURN TO USER
026132  023420  001750  000144  $DTBL:   .WORD   10000.,1000.,100.,10.
026142                          $DBLK:   .BLKW   5
```

```
                                        .SBTTL  SUBROUTINE TO EXPAND DECIMAL TO LARGER THAN 32767
                                        ;*****************************************************************
026154  012702  026150          EXPAND: MOV     #$DBLK+6,R2     ;&& MOVE LOCATION OF LCD+1 TO R2
026160  012703  026154                  MOV     #$DBLK+12,R3    ;&& MOVE NEW LOCATION OF LCD+2 TO R3
026164  105043                          CLRB    -(R3)           ;&& MAKE SURE TERMINATOR IS THERE
026166  114243                          MOVB    -(R2),-(R3)     ;&& MOVE THE 5 ASCII'S TO THEIR NEW LOCATIONS
026170  114243                          MOVB    -(R2),-(R3)     ;&&
026172  114243                          MOVB    -(R2),-(R3)     ;&&
026174  114243                          MOVB    -(R2),-(R3)     ;&&
026176  114243                          MOVB    -(R2),-(R3)     ;&&
026200  112743  000040                  MOVB    #' ,-(R3)       ;&& MOVE 4 SPACE CHARACTERS TO THE 4 NEW LOCATIONS
026204  112743  000040                  MOVB    #' ,-(R3)       ;&&
026210  112743  000040                  MOVB    #' ,-(R3)       ;&&
026214  112743  000040                  MOVB    #' ,-(R3)       ;&&
026220  013746  001360                  MOV     $TMP0,-(SP)     ;&& SAVE $TMP0
026224  005037  001360                  CLR     $TMP0           ;&& CLEAR LOCATION TO USE AS ACCUMULATOR
026230  013746  001362                  MOV     $TMP1,-(SP)     ;&& SAVE $TMP1
026234  005037  001362                  CLR     $TMP1           ;&& CLEAR LOCATION TO USE AS 2ND ACCUMULATOR
026240  012701  026151                  MOV     #$DBLK+7,R1     ;&& MOVE ADDRESS OF LCD TO R2
026244  012702  026420                  MOV     #$NUMS+10,R2    ;&& MOVE ADDRESS+10 OF WORD STREAM OF 8^6 TO R2
026250  016600  000026          1$:     MOV     26(SP),R0       ;&& MOVE OVERFLOW LOCATION CONTENTS TO R0
026254  111137  001360                  MOVB    (R1),$TMP0      ;&& MOVE ASCII TO THE TEMPORARY LOCATION
026260  052737  000060  001360          BIS     #'0,$TMP0       ;&& MAKE LOCATION AN ASCII IF NOT ALREADY
026266  061237  001360          2$:     ADD     (R2),$TMP0      ;&& ADD THE NUMBER TO THE ASCII
026272  022737  000071  001360          CMP     #'9,$TMP0       ;&& HAVE WE SURPASSED ASCII '9'?
026300  002025                          BGE     4$              ;&& BRANCH IF NOT
026302  162737  000012  001360          SUB     #10.,$TMP0      ;&& SUBTRACT 10 FROM THE ASCII AND
026310  010103                          MOV     R1,R3           ;&& MOVE PRESENT ASCII ADDRESS TO R3
026312  005303                  3$:     DEC     R3              ;&& DECREMENT TO NEXT SIGNIFICANT ASCII DIGIT
026314  105213                          INCB    (R3)            ;&& ADD THE CARRY TO THE ASCII
026316  152713  000060                  BISB    #'0,(R3)        ;&& SET BITS TO MAKE IT A NUMBER ASCII
026322  122713  000071                  CMPB    #'9,(R3)        ;&& SEE IF CARRY NEEDS TO BE TRANSFERED
026326  002012                          BGE     4$              ;&& BRANCH IF NOT
026330  111337  001362                  MOVB    (R3),$TMP1      ;&& MOVE BYTE TO LOCATION $TMP1
026334  162737  000012  001362          SUB     #10.,$TMP1      ;&& SUBTRACT 10 DECIMAL AND
026342  113713  001362                  MOVB    $TMP1,(R3)      ;&& MOVE IT BACK
026346  022703  026142                  CMP     #$DBLK,R3       ;&& SEE IF ALL POSITIONS HAVE BEEN DONE
026352  001357                          BNE     3$              ;&& BRANCH BACK IF NOT
026354  005300                  4$:     DEC     R0              ;&& DECREMENT THE OVERFLOW LOCATION R0
026356  001343                          BNE     2$              ;&& BRANCH IF NOT ALL ADDED
026360  113711  001300                  MOVB    $TMP0,(R1)      ;&& MOVE NEW NUMBER TO LOCATION
026364  005301                          DEC     R1              ;&& POINT R1 TO THE NEXT ASCII LOCATION
026366  005742                          TST     -(R2)           ;&& POINT R2 TO NEXT DIGIT TO ADD
026370  022701  026145                  CMP     #$DBLK+3,R1     ;&& SEE IF ALL DIGITS HAVE BEEN ADDED
026374  001325                          BNE     1$              ;&& BRANCH IF NOT DONE
026376  012637  001362                  MOV     (SP)+,$TMP1     ;&& RESTORE $TMP1
026402  012637  001360                  MOV     (SP)+,$TMP0     ;&& RESTORE $TMP0
026406  000207                          RTS     PC              ;&& RETURN
026410  000003  000002  000007  $NUMS:  .WORD   3,2,7,6,8.
```

```
3796                                    .SBTTL  TTY INPUT ROUTINE
                                        ;*****************************************************************
                                        .ENABL  LSB
                                        ;*****************************************************************
                                        ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                        ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
                                        ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                                        ;*WHEN OPERATING IN TTY FLAG MODE.
026422  022737  000176  001340  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
026430  001112                          BNE     15$             ;;BRANCH IF NO
026432  105737  030707                  TSTB    CHARCT          ;&& SEE IF CHARACTER WAS INPUTED DURING PRINT
026436  001405                          BEQ     1$              ;&& BRANCH IF NOT
026440  113746  030707                  MOVB    CHARCT,-(SP)    ;&& MOVE CHARACTER TO THE STACK
026444  105037  030707                  CLRB    CHARCT          ;&& CLEAR THE CHARACTER FROM CHARCT
026450  000405                          BR      2$              ;&& GO CHECK IT OUT
026452  105777  152666          1$:     TSTB    @$TKS           ;;CHAR THERE?
026456  100077                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
026460  117746  152662                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
026464  042716  177600          2$:     BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
026470  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
026474  001404                          BEQ     3$              ;&& YES, BRANCH AROUND RETURN TO USER SETUP
026476  116637  177776  030707          MOVB    -2(SP),CHARCT   ;&& MOVE CHARACTER BACK TO CHARCT
026504  000464                          BR      15$             ;&& RETURN TO USER
026506  123727  001334  000001  3$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
026514  001460                          BEQ     15$             ;;BRANCH IF YES
026516  104401  027202                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
026522  104401  027207          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
026526  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
026532  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
026534  104401  027220                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
026540  105037  030707          19$:    CLRB    CHARCT          ;&& CLEAR ANY CHARACTER THAT WAS INPUTED DURING PRINT
026544  005046                          CLR     -(SP)           ;;CLEAR COUNTER
026546  005046                          CLR     -(SP)           ;;THE NEW SWR
026550  105777  152570          7$:     TSTB    @$TKS           ;;CHAR THERE?
026554  100375                          BPL     7$              ;;IF NOT TRY AGAIN
026556  117746  152564                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
026562  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
026566  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
026572  001005                          BNE     10$             ;;BRANCH IF NOT
026574  104401  027175                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
026600  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
026604  000746                          BR      $GTSWR          ;;LET'S TRY IT AGAIN
026606  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
026612  001022                          BNE     16$             ;;BRANCH IF NO
026614  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
026620  001403                          BEQ     11$             ;;BRANCH IF YES
026622  016677  000002  152510          MOV     2(SP),@SWR      ;;SAVE NEW SWR
026630  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
026634  104401  001405          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
026640  123727  001335  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
026646  001003                          BNE     15$             ;;BRANCH IF NOT
026650  012777  000100  152466          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
026656  000002          15$:            RTI                     ;;RETURN
026660  004737  025244          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
026664  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
026670  002420                          BLT     18$             ;;BRANCH IF YES
026672  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
```

```
026676  003015                          BGT     18$             ;;BRANCH IF YES
026700  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
026704  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
026710  001403                          BEQ     17$             ;;BRANCH IF YES
026712  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
026714  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
026716  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
026720  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
026724  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
026730  000707                          BR      7$              ;;GET THE NEXT ONE
026732  104401  001404          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
026736  000720                          BR      20$             ;;SIMULATE CONTROL-U
                                .DSABL  LSB
                                ;*****************************************************************
```

```
                                        .SBTTL  ROUTINE TO INPUT A SINGLE CHARACTER FROM TTY
                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*CALL:
                                ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
                                ;
026740  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
026742  016665  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
026750  105777  152370          1$:     TSTB    @$TKS           ;;WAIT FOR
026754  100375                          BPL     1$              ;;A CHARACTER
026756  117766  152364  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
026764  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
026772  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
027000  001013                          BNE     3$              ;;BRANCH IF NO
027002  105777  152336          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
027006  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
027010  117746  152332                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
027014  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
027020  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
027024  001366                          BNE     2$              ;;IF NOT DISCARD IT
027026  000750                          BR      1$              ;;YES, RESUME
027030  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
027036  002407                          BLT     4$              ;;BRANCH IF YES
027040  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
027046  003003                          BGT     4$              ;;BRANCH IF YES
027050  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
027056  000002                  4$:     RTI                     ;;GO BACK TO USER
```

```
                                        .SBTTL  ROUTINE TO INPUT A STRING FROM TTY
                                ;*******************************************************************
                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;*CALL:
                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
027060  010346                 $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
027062  012703  027166          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
027066  022703  027175          2$:     CMP     #$TTYIN+7.,R3   ;;BUFFER FULL?
027072  101405                          BLOS    4$              ;;BR IF YES
027074  104411                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
027076  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
027100  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
027104  001003                          BNE     3$              ;;SKIP IF NOT
027106  104401  001404          4$:     TYPE    ,$QUES          ;;TYPE A '?'
027112  000763                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
027114  111337  027164          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
027120  104401  027164                  TYPE    ,9$
027124  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
027130  001356                          BNE     2$              ;;LOOP IF NOT RETURN
027132  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
027136  104401  001406                  TYPE    ,$LF            ;;TYPE A LINE FEED
027142  012603                          MOV     (SP)+,R3        ;;RESTORE R3
027144  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
027146  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
027154  012766  027166  000004          MOV     #$TTYIN,4(SP)
027162  000002                          RTI                     ;;RETURN
027164     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
027165     000                          .BYTE   0               ;;TERMINATOR
027166                          $TTYIN: .BLKB   7.              ;;RESERVE 7. BYTES FOR TTY INPUT
027175     136     125     015  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
027202     136     107     015  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
027207     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
027220     040     040     116  $MNEW:  .ASCIZ  /  NEW = /
                                .EVEN
```

```
3797                                    .SBTTL  READ A DECIMAL NUMBER FROM THE TTY
                                        ;**************************************************************
                                        ;*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
                                        ;*CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
                                        ;*ARE READ A "?" FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
                                        ;*THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
                                        ;*USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
                                        ;*POSITIVE 32767 TO NEGATIVE 32768.
                                        ;*CALL:
                                        ;*      RDDEC                   ;;READ A DECIMAL NUMBER
                                        ;*      RETURN HERE             ;;NUMBER IS ON TOP OF THE STACK
                                        ;
027232  011646                 $RDDEC: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR
027234  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE INPUT NUMBER
027242  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
027244  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
027246  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
027250  104412                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
027252  012600                          MOV     (SP)+,R0        ;;ADDRESS OF 1ST CHAR.
027254  010037  027400                  MOV     R0,6$           ;;SAVE INCASE OF BAD INPUT
027260  005046                          CLR     -(SP)           ;;CLEAR DATA WORD
027262  005002                          CLR     R2              ;;SIGN SET POSITIVE
027264  122710  000055                  CMPB    #'-,(R0)        ;;SEE IF A MINUS SIGN WAS TYPED
027270  001001                          BNE     2$              ;;BR IF NO MINUS SIGN
027272  112002                          MOVB    (R0)+,R2        ;;SAVE FOR LATER USE
027274  112001                  2$:     MOVB    (R0)+,R1        ;;PICKUP THIS CHARACTER
027276  001424                          BEQ     3$              ;;GET OUT IF ZERO
027300  122701  000060                  CMPB    #'0,R1          ;;MAKE SURE THIS CHARACTER
027304  003032                          BGT     5$              ;;IS A DIGIT BETWEEN 0 & 9
027306  122701  000071                  CMPB    #'9,R1
027312  002427                          BLT     5$
027314  032716  170000                  BIT     #^C7777,(SP)    ;;DON'T LET NUMBER GET TO BIG
027320  001024                          BNE     5$              ;;BR IF NUMBER WOULD OVERFLOW
027322  006316                          ASL     (SP)            ;;*2
027324  011646                          MOV     (SP),-(SP)      ;;SAVE FOR LATER
027326  006316                          ASL     (SP)            ;;*4
027330  006316                          ASL     (SP)            ;;*8.
027332  062616                          ADD     (SP)+,(SP)      ;;*10.
027334  102416                          BVS     5$              ;;OVERFLOW ISN'T ALLOWED
027336  162701  000060                  SUB     #'0,R1          ;;STRIP AWAY THE ASCII JUNK
027342  060116                          ADD     R1,(SP)         ;;ADD IN THIS DIGIT
027344  102412                          BVS     5$              ;;OVERFLOW ISN'T ALLOWED
027346  000752                          BR      2$              ;;LOOP
027350  005702                  3$:     TST     R2              ;;CHECK IF NUMBER IS NEG
027352  001401                          BEQ     4$              ;;BR IF NO
027354  005416                          NEG     (SP)            ;;YES--NEGATE THE NUMBER
027356  012666  000012          4$:     MOV     (SP)+,12(SP)    ;;SAVE THE RESULT
027362  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
027364  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
027366  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
027370  000002                          RTI                     ;;RETURN
027372  005726                  5$:     TST     (SP)+           ;;CLEAN PARTIAL NUMBER FROM STACK
027374  105010                          CLRB    (R0)            ;;SET A TERMINATOR
027376  104401                          TYPE                    ;;TYPE THE INPUT UP TO BAD CHAR.
027400  000000                  6$:     .WORD   0               ;;POINTER GOES HERE
027402  104401  001404                  TYPE    ,$QUES          ;;"?" "CR" &"LF"
027406  000720                          BR      1$              ;;TRY AGAIN
```

```
 3798                                   .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
                                        ;**************************************************************
                                        ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
                                        ;*CHANGE IT TO BINARY.
                                        ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
                                        ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
                                        ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
                                        ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
                                        ;*CALL:
                                        ;*      RDOCT                   ;;READ AN OCTAL NUMBER
                                        ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
                                        ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
 027410  011646                        $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
 027412  016666  000004  000002                 MOV     4(SP),2(SP)     ;;INPUT NUMBER
 027420  010046                                 MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 027422  010146                                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 027424  010246                                 MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 027426  104412                         1$:     RDLIN                   ;;READ AN ASCIZ LINE
 027430  012600                                 MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
 027432  010037  027536                         MOV     R0,5$           ;;AND SAVE IT
 027436  005001                                 CLR     R1              ;;CLEAR DATA WORD
 027440  005002                                 CLR     R2
 027442  112046                         2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
 027444  001420                                 BEQ     3$              ;;IF ZERO GET OUT
 027446  122716  000060                         CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
 027452  003026                                 BGT     4$              ;;IS AN OCTAL DIGIT
 027454  122716  000067                         CMPB    #'7,(SP)
 027460  002423                                 BLT     4$
 027462  006301                                 ASL     R1              ;;*2
 027464  006102                                 ROL     R2
 027466  006301                                 ASL     R1              ;;*4
 027470  006102                                 ROL     R2
 027472  006301                                 ASL     R1              ;;*8
 027474  006102                                 ROL     R2
 027476  042716  177770                         BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
 027502  062601                                 ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
 027504  000756                                 BR      2$              ;;LOOP
 027506  005726                         3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
 027510  010166  000012                         MOV     R1,12(SP)       ;;SAVE THE RESULT
 027514  010237  027546                         MOV     R2,$HIOCT
 027520  012602                                 MOV     (SP)+,R2        ;;POP STACK INTO R2
 027522  012601                                 MOV     (SP)+,R1        ;;POP STACK INTO R1
 027524  012600                                 MOV     (SP)+,R0        ;;POP STACK INTO R0
 027526  000002                                 RTI                     ;;RETURN
 027530  005726                         4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
 027532  105010                                 CLRB    (R0)            ;;SET A TERMINATOR
 027534  104401                                 TYPE                    ;;TYPE UP THRU THE BAD CHAR.
 027536  000000                         5$:     .WORD   0
 027540  104401  001404                         TYPE    ,$QUES          ;;"?" "CR" & "LF"
 027544  000730                                 BR      1$              ;;TRY AGAIN
 027546  000000                         $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
```

```
3799                                    .SBTTL  TRAP DECODER
                                        ;*********************************************************************
                                        ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
                                        ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
                                        ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
                                        ;*GO TO THAT ROUTINE.
027550  010046                          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
027552  016600  000002                          MOV     2(SP),R0        ;;GET TRAP ADDRESS
027556  005740                                  TST     -(R0)           ;;BACKUP BY 2
027560  111000                                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
027562  006300                                  ASL     R0              ;;POSITION FOR INDEXING
027564  016000  027604                          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
027570  000200                                  RTS     R0              ;;GO TO ROUTINE
                                        ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
027572  011646                          $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
027574  016666  000004  000002                  MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
027602  000002                                  RTI                     ;;RESTORE THE PSW
                                        .SBTTL  TRAP TABLE
                                        ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                        ;*BY THE "TRAP" INSTRUCTION.
                                        ;       ROUTINE
                                        ;       -------
027604  027572                          $TRPAD: .WORD   $TRAP2
027606  025032                                  $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
027610  025424                                  $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
027612  025400                                  $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
027614  025440                                  $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
027616  025636                                  $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
027620  025626                                  $TYPDE  ;;CALL=TYPDE    TRAP+6(104406)  TYPE DECIMAL NUMBER GREATER THAN 32767
027622  026522                                  $GTSWR  ;;CALL=GTSWR    TRAP+7(104407)  GET SOFT-SWR SETTING
027624  026422                                  $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
027626  026740                                  $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
027630  027060                                  $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
027632  027410                                  $RDOCT  ;;CALL=RDOCT    TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
027634  027232                                  $RDDEC  ;;CALL=RDDEC    TRAP+14(104414) READ A DECIMAL NUMBER FROM TTY
```

```
3800                                    .SBTTL  SCOPE HANDLER ROUTINE
                                        ;**************************************************************************
                                        ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                        ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                        ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW14=1         LOOP ON TEST
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*SW08=1         LOOP ON TEST IN SWR<6:0>
                                        ;*CALL
                                        ;*      SCOPE               ;;SCOPE=IOT
027636  032777  001000  151474  $SCOPE: BIT     #BIT09,@SWR         ;;LOOP ON ERROR?
027644  001406                          BEQ     1$                  ;;BR IF NO
027646  005737  001312                  TST     $ERTTL              ;;SEE IF THERE WERE ANY ERRORS YET
027652  001403                          BEQ     1$                  ;;BR IF NOT YET
027654  013716  001310                  MOV     $LPERR,(SP)         ;;FUDGE RETURN
027660  000002                          RTI                         ;;RETURN
027662  032777  040000  151450  1$:     BIT     #BIT14,@SWR         ;;LOOP ON TEST?
027670  001403                          BEQ     2$                  ;;BR IF NO
027672  013716  001306                  MOV     $LPADR,(SP)         ;;FUDGE RETURN
027676  000002                          RTI                         ;;RETURN
027700  105737  002706          2$:     TSTB    EOPLOC              ;&& TEST TO SEE IF EOP'S ARE TO PRINT
027704  001043                          BNE     6$                  ;&& BRANCH IF NOT
027706  105737  030707                  TSTB    CHARCT              ;&& SEE IF A CHARACTER WAS INPUTED IN A PRINT ROUTINE
027712  001406                          BEQ     3$                  ;&& BRANCH TO CHECK KEYBOARD IF NOT
027714  113737  030707  002706          MOVB    CHARCT,EOPLOC       ;&& MOVE THE CHARACTER TO EOPLOC
027722  105037  030707                  CLRB    CHARCT              ;&& CLEAR THE LOCATION
027726  000411                          BR      4$                  ;&& BRANCH TO CHECK THE CHARACTER
027730  105777  151410          3$:     TSTB    @$TKS               ;&& SEE IF EOP REQUESTED
027734  100055                          BPL     8$                  ;&& BRANCH IF NOT
027736  117737  151404  002706          MOVB    @$TKB,EOPLOC        ;&& GET CHARACTER
027744  142737  000200  002706          BICB    #200,EOPLOC         ;&& CLEAR PARITY BIT
027752  122737  000030  002706  4$:     CMPB    #30,EOPLOC          ;&& SEE IF A (^X)
027760  001406                          BEQ     5$                  ;&& BRANCH IF IT IS
027762  113737  002706  030707          MOVB    EOPLOC,CHARCT       ;&& MOVE CHARACTER BACK TO CHARCT FOR POSSIBLE FUTURE USE
027770  105037  002706                  CLRB    EOPLOC              ;&& CLEAR THE LOCATION
027774  000435                          BR      8$                  ;&& BRANCH TO START SCOPE ROUTINE
027776  104401  030326          5$:     TYPE    ,HAKTPM             ;&& TYPE: 'HIT ANY KEY TO OBTAIN A PROGRESS REPORT,'
                                                                    ;&&       'ENTER (^X) TO RESUME EOP'S AND EOD'S'
                                                                    ;&&       'ENTER THE KEY REPEATEDLY, AS RESETS DONE IN THE D
                                                                    ;&&       'MAY CLEAR THE CHARACTER BEFORE THE TESTS FOR THE
030002  105337  002706                  DECB    EOPLOC              ;&& MAKE ^X ANOTHER CHARACTER AND
030006  105037  030707                  CLRB    CHARCT              ;&& CLEAR ANY CHARACTER THAT MAY BE THERE
030012  000426                          BR      8$                  ;&& BRANCH TO START SCOPE ROUTINE
030014  105737  030707          6$:     TSTB    CHARCT              ;&& SEE IF A CHARACTER WAS INPUTED IN THE TYPE ROUTINE
030020  001011                          BNE     7$                  ;&& GO TEST THE CHARACTER IF SO
030022  105777  151316                  TSTB    @$TKS               ;&& SEE IF CHARACTER WAITING
030026  100020                          BPL     8$                  ;&& BRANCH IF NOT
030030  117737  151312  030707          MOVB    @$TKB,CHARCT        ;&& MOVE THE CHARACTER FOR TESTING
030036  142737  000200  030707          BICB    #200,CHARCT         ;&& CLEAR THE PARITY BIT
030044  122737  000030  030707  7$:     CMPB    #30,CHARCT          ;&& SEE IF ANOTHER (^X) WAS INPUTED
030052  001006                          BNE     8$                  ;&& BRANCH IF NOT
030054  105037  002706                  CLRB    EOPLOC              ;&& CLEAR EOPLOC TO RESUME EOP MESSAGES
030060  104401  030636                  TYPE    ,EOPRSM             ;&& TYPE: 'EOP'S AND EOD'S WILL RESUME PRINTING'
030064  105037  030707                  CLRB    CHARCT              ;&& MAKE CHARACTER SOMETHING ELSE
030070  105737  030707          8$.     TSTB    CHARCT              ;&& SEE IF A CHARACTER IS WAITING
030074  001410                          BEQ     9$                  ;&& BRANCH AROUND (^Y) TEST IF NOT
```

```
030076  122737  000031  030707          CMPB    #31,CHARCT      ;&& IS THIS A (^Y) (REQUEST FOR RUN SUMMARY)
030104  001004                          BNE     9$              ;&& BRANCH IF NOT
030106  004737  003170                  JSR     PC,PTCAPT       ;&& GO TO SUBROUTINE TO PRINT SUMMARY
030112  105037  030707                  CLRB    CHARCT          ;&& CLEAR THE CHARCT LOCATION SO SUMMARY NOT REPEATED
030116  104410                  9$:     CKSWR                   ;;TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER
                                ;#####START OF CODE FOR THE XOR TESTER#####
030120  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
030122  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
030126  012737  030146  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
030134  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
030140  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
030144  000444                          BR      $SVLAD          ;;GO TO THE NEXT TEST
030146  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
030150  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
030154  000432                          BR      7$              ;;LOOP ON THE PRESENT TEST
030156                          6$:;#####END OF CODE FOR THE XOR TESTER#####
030156  032777  000400  151154          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
030164  001432                          BEQ     4$              ;;BR IF NO
030166  005046                          CLR     -(SP)           ;;CLEAR A TEMP. LOCATION
030170  117716  151144                  MOVB    @SWR,(SP)       ;;PICKUP THE DESIRED TEST NUMBER
030174  042716  000200                  BIC     #$SWRMK,(SP)    ;;MASK OUT UNDESIRED BITS
030200  001416                          BEQ     8$              ;;BRANCH IF BAD TEST NUMBER IN SWR
030202  022716  000037                  CMP     #37,(SP)        ;;CHECK THE NUMBER IN THE SWR
030206  002413                          BLT     8$              ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
030210  011637  001302                  MOV     (SP),$TSTNM     ;;UPDATE THE TEST NUMBER IN $TSTNM
030214  011637  001414                  MOV     (SP),$TESTN     ;&& UPDATE THE TEST NUMBER IN $TESTN
030220  005316                          DEC     (SP)            ;;BACKUP BY ONE
030222  006316                          ASL     (SP)            ;;SCALE THE TEST NUMBER AS AN INDEX
030224  062716  030752                  ADD     #$SW08TBL,(SP)  ;;FORM THE ADDRESS OF TEST POINTER
030230  013637  001306                  MOV     @(SP)+,$LPADR   ;;SET LOOP ADDRESS TO DESIRED TEST
030234  000426                          BR      $OVER           ;;GO LOOP ON THE TEST
030236  005726                  8$:     TST     (SP)+           ;;CLEAN THE BAD TEST NUMBER OFF OF THE STACK
030240                          2$:     ;TSTB   $ERFLG          ;;HAS AN ERROR OCCURRED? ;&& ELIMINATED FOR CZDRLC
030240  001406                          BEQ     $SVLAD          ;;BR IF NO
030242  013737  001310  001306  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
030250  000420                          BR      $OVER
030252  105037  001303          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
030256  105237  001302          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
030262  113737  001302  001414          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
030270  011637  001306                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
030274  011637  001310                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
030300  005037  001376                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
030304  112737  000001  001315          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
030312  013777  001302  151022  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
030320  013716  001306                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
030324  000002                          RTI                     ;;RETURN
030326     136     130     200  HAKTPM: .ASCII  /^X/<CRLF>/HIT ANY KEY TO OBTAIN A PROGRESS REPORT,/<CRLF> ;&&
030402     105     116     124          .ASCII  /ENTER (^X) TO RESUME EOP'S AND EOD'S/<CRLF>                ;&&
030447     105     116     124          .ASCII  /ENTER THE KEY REPEATEDLY, AS RESETS DONE IN THE DIAGNOSTIC/<CRLF> ;&&
030542     115     101     131          .ASCIZ  /MAY CLEAR THE CHARACTER BEFORE THE TESTS FOR THE CHARACTER/<CRLF> ;&&
030636     136     130     200  EOPRSM: .ASCIZ  /^X/<CRLF>/EOP'S AND EOD'S WILL RESUME PRINTING/<CRLF>     ;&&
030707     000                  CHARCT: .BYTE   0               ;&& LOCATION TO HOLD INPUTED CHARACTER
030710     040     040     124  TESLR:  .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /     ;&& ERROR MESSAGE FOR $EOP
                                        .EVEN
```

```
030752                          $SW08TBL:
030752  011274                          .WORD   TST1+30         ;STARTING ADDRESS+30 OF TEST 1
030754  011472                          .WORD   TST2+30         ;STARTING ADDRESS+30 OF TEST 2
030756  011622                          .WORD   TST3+30         ;STARTING ADDRESS+30 OF TEST 3
030760  012154                          .WORD   TST4+30         ;STARTING ADDRESS+30 OF TEST 4
030762  012760                          .WORD   TST5+30         ;STARTING ADDRESS+30 OF TEST 5
030764  013154                          .WORD   TST6+30         ;STARTING ADDRESS+30 OF TEST 6
030766  013522                          .WORD   TST7+30         ;STARTING ADDRESS+30 OF TEST 7
030770  013700                          .WORD   TST10+30        ;STARTING ADDRESS+30 OF TEST 10
030772  014134                          .WORD   TST11+30        ;STARTING ADDRESS+30 OF TEST 11
030774  014320                          .WORD   TST12+30        ;STARTING ADDRESS+30 OF TEST 12
030776  014504                          .WORD   TST13+30        ;STARTING ADDRESS+30 OF TEST 13
031000  014670                          .WORD   TST14+30        ;STARTING ADDRESS+30 OF TEST 14
031002  015060                          .WORD   TST15+30        ;STARTING ADDRESS+30 OF TEST 15
031004  015312                          .WORD   TST16+30        ;STARTING ADDRESS+30 OF TEST 16
031006  015550                          .WORD   TST17+30        ;STARTING ADDRESS+30 OF TEST 17
031010  016146                          .WORD   TST20+30        ;STARTING ADDRESS+30 OF TEST 20
031012  016552                          .WORD   TST21+30        ;STARTING ADDRESS+30 OF TEST 21
031014  017066                          .WORD   TST22+30        ;STARTING ADDRESS+30 OF TEST 22
031016  017364                          .WORD   TST23+30        ;STARTING ADDRESS+30 OF TEST 23
031020  020122                          .WORD   TST24+30        ;STARTING ADDRESS+30 OF TEST 24
031022  020424                          .WORD   TST25+30        ;STARTING ADDRESS+30 OF TEST 25
031024  020714                          .WORD   TST26+30        ;STARTING ADDRESS+30 OF TEST 26
031026  021172                          .WORD   TST27+30        ;STARTING ADDRESS+30 OF TEST 27
031030  021410                          .WORD   TST30+30        ;STARTING ADDRESS+30 OF TEST 30
031032  021676                          .WORD   TST31+30        ;STARTING ADDRESS+30 OF TEST 31
031034  022026                          .WORD   TST32+30        ;STARTING ADDRESS+30 OF TEST 32
031036  022406                          .WORD   TST33+30        ;STARTING ADDRESS+30 OF TEST 33
031040  022712                          .WORD   TST34+30        ;STARTING ADDRESS+30 OF TEST 34
031042  023274                          .WORD   TST35+30        ;STARTING ADDRESS+30 OF TEST 35
031044  023572                          .WORD   TST36+30        ;STARTING ADDRESS+30 OF TEST 36
031046  024106                          .WORD   TST37+30        ;STARTING ADDRESS+30 OF TEST 37
```

```
3801                                    .SBTTL  ERROR HANDLER ROUTINE
                                        ;**********************************************************************
                                        ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                        ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                        ;*AND GO TO $ERRTYP ON ERROR
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW15=1         HALT ON ERROR
                                        ;*SW13=1         INHIBIT ERROR TYPEOUTS
                                        ;*SW10=1         BELL ON ERROR
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*CALL
                                        ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
031050                                  $ERROR:
031050  104410                                  CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
031052  105237  001303            7$:     INCB    $ERFLG                  ;;SET THE ERROR FLAG
031056  001775                            BEQ     7$                      ;;DON'T LET THE FLAG GO TO ZERO
031060  013777  001302  150254            MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER AND ERROR FLAG
031066  032777  002000  150244            BIT     #BIT10,@SWR             ;;BELL ON ERROR?
031074  001402                            BEQ     1$                      ;;NO - SKIP
031076  104401  001400                    TYPE    ,$BELL                  ;;RING BELL
031102  005237  001312            1$:     INC     $ERTTL                  ;;COUNT THE NUMBER OF ERRORS
031106  005237  002714                    INC     ERRCNT                  ;&& INCREMENT THE ERROR COUNT
031112  011637  001316                    MOV     (SP),$ERRPC             ;;GET ADDRESS OF ERROR INSTRUCTION
031116  162737  000002  001316            SUB     #2,$ERRPC
031124  117737  150166  001314            MOVB    @$ERRPC,$ITEMB          ;;STRIP AND SAVE THE ERROR ITEM CODE
031132  032777  020000  150200            BIT     #BIT13,@SWR             ;;SKIP TYPEOUT IF SET
031140  001002                            BNE     20$                     ;;SKIP TYPEOUTS
031142  004737  031250                    JSR     PC,$ERRTYP              ;;GO TO USER ERROR ROUTINE
031146                                  20$:
031146  122737  000001  001430            CMPB    #APTENV,$ENV            ;;RUNNING IN APT MODE
031154  001007                            BNE     2$                      ;;NO,SKIP APT ERROR REPORT
031156  113737  001314  031170            MOVB    $ITEMB,21$              ;;SET ITEM NUMBER AS ERROR NUMBER
031164  004737  031544                    JSR     PC,$ATY4                ;;REPORT FATAL ERROR TO APT
031170     000                  21$:     .BYTE   0
031171     000                           .BYTE   0
031172  000777                  22$:     BR      22$                     ;;APT ERROR LOOP
031174  005777  150140            2$:     TST     @SWR                    ;;HALT ON ERROR
031200  100002                            BPL     3$                      ;;SKIP IF CONTINUE
031202  000000                            HALT                            ;;HALT ON ERROR!
031204  104410                            CKSWR                           ;;TEST FOR CHANGE IN SOFT-SWR
031206  005737  001376            3$:     TST     $ESCAPE                 ;;CHECK FOR AN ESCAPE ADDRESS
031212  001402                            BEQ     4$                      ;;BR IF NONE
031214  013716  001376                    MOV     $ESCAPE,(SP)            ;;FUDGE RETURN ADDRESS FOR ESCAPE
031220  032777  001000  150112    4$:     BIT     #BIT9,@SWR              ;&& SEE IF WE ARE TO LOOP ON ERROR
031226  001402                            BEQ     5$                      ;&& BRANCH OUT IF NOT
031230  013716  001310                    MOV     $LPERR,(SP)             ;&& FUDGE RETURN
031234                                  5$:
031234  022737  024750  000042            CMP     #$ENDAD,@#42            ;;ACT-11 AUTO-ACCEPT?
031242  001001                            BNE     6$                      ;;BRANCH IF NO
031244  000000                            HALT                            ;;YES
031246                                  6$:
031246  000002                            RTI                             ;;RETURN
```

```
3802                                    .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.
031250                                  $ERRTYP:
031250  010046                                  MOV     R0,-(SP)        ;;SAVE R0
031252  005000                                  CLR     R0              ;;CLEAR R0 TO RECEIVE ITEM INDEX
031254  113700  001314                          MOVB    @#$ITEMB,R0     ;;PICKUP THE ITEM INDEX
031260  001004                                  BNE     1$              ;;IF ITEM NUMBER IS ZERO, TYPE THE PC OF THE ERROR
031262  013746  001316                          MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                        ;;ERROR ADDRESS
031266  104402                                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
031270  000513                                  BR      14$             ;;GET OUT
031272  010037  001360          1$:             MOV     R0,$TMPO        ;&& MOVE R0 TO $TMPO FOR 200 TEST
031276  042700  000200                          BIC     #200,R0         ;&& CLEAR BIT 7 IF PRESENT
031302  005300                                  DEC     R0              ;&& MAKE POINTER AN INDEX
031304  006300                                  ASL     R0              ;&& SHIFT TO MULTIPLY BY 10 (OCTAL)
031306  006300                                  ASL     R0              ;&&
031310  006300                                  ASL     R0              ;&&
031312  105737  001360                          TSTB    $TMPO           ;&& SEE IF ITEM NUMBER IS OVER 200
031316  100041                                  BPL     4$              ;&& BRANCH IF ITEM NUMBER IS LESS THAN 200
031320  023727  002714  000020                  CMP     ERRCNT,#20      ;&& SEE IF 20 (OCTAL) ERRORS HAVE PRINTED
031326  002404                                  BLT     2$              ;&& BRANCH TO PRINT THE ERROR IF LESS
031330  003073                                  BGT     14$             ;&& BRANCH TO RETURN IF GREATER - NO MORE DATA IS TO PRINT
031332  104401  036663                          TYPE    ,NOMORE         ;&& TYPE MESSAGE ANNOUNCING NO MORE PRINTING OF ERRORS
031336  000470                                  BR      14$             ;&& BRANCH TO RETURN
031340  022737  000001  002714  2$:             CMP     #1,ERRCNT       ;&& SEE IF THIS IS THE FIRST ERROR
031346  001415                                  BEQ     3$              ;&& BRANCH IF IT WAS AND GO TYPE ERROR MESSAGE
031350  123737  001360  031524                  CMPB    $TMPO,MEPITM    ;&& SEE IF ITEM MATCHES LAST MULTIPLE ERROR
031356  001011                                  BNE     3$              ;&& BRANCH IF NOT - NEW HEADER NEEDED
031360  032777  000200  147752                  BIT     #BIT7,@SWR      ;&& SEE IF SWITCH REGISTER BIT 7 IS SET
031366  001054                                  BNE     14$             ;&& BRANCH TO RETURN IF SWITCH SET
031370  042700  177400                          BIC     #177400,R0      ;&& CLEAR UPPER BYTE OF R0 EXPOSING ITEM BYTE
031374  062700  002300                          ADD     #ER200+4,R0     ;&& POINT TO DATA TABLE ENTRY
031400  000434                                  BR      9$              ;&& BRANCH TO PRINT DATA
031402  113737  001360  031524  3$:             MOVB    $TMPO,MEPITM    ;&& MOVE ITEM NUMBER TO MEPITM FOR POSSIBLE FUTURE USE
031410  042700  177000                          BIC     #177000,R0      ;&& CLEAR UPPER BYTE OF R0
031414  062700  000540                          ADD     #ER200-$ERRTB,R0;&& ADD 200 BASE POINTER TO R0 AND
031420  000402                                  BR      5$              ;&& BRANCH AROUND ERRCNT CLEAR
031422  005037  002714          4$:             CLR     ERRCNT          ;&& CLEAR ERRCNT SO MULTIPLE ERRORS GET NEW HEADER
031426  104401  001405          5$:             TYPE    ,$CRLF          ;&& TYPE  <CRLF>
031432  062700  001534                          ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
031436  012037  031446                          MOV     (R0)+,6$        ;;PICKUP "ERROR MESSAGE" POINTER
031442  001404                                  BEQ     7$              ;;SKIP TYPEOUT IF NO POINTER
031444  104401                                  TYPE                    ;;TYPE THE "ERROR MESSAGE"
031446  000000          6$:             .WORD   0                       ;;"ERROR MESSAGE" POINTER GOES HERE
031450  104401  001405                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
031454  012037  031464          7$:             MOV     (R0)+,8$        ;;PICKUP "DATA HEADER" POINTER
031460  001404                                  BEQ     9$              ;;SKIP TYPEOUT IF 0
031462  104401                                  TYPE                    ;;TYPE THE "DATA HEADER"
031464  000000          8$:             .WORD   0                       ;;"DATA HEADER" POINTER GOES HERE
031466  104401  001405                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
031472  011000          9$:             MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
031474  001407                                  BEQ     13$             ;;GO AROUND ROUTINE TO TYPE THE DATA IF NONE
031476  013046          10$:            MOV     @(R0)+,-(SP)    ;;PUT OCTAL DATA ON STACK FOR TYPING
031500  104402                                  TYPOC                   ;;TYPE AN OCTAL NUMBER
```

```
031502  005710                          TST     (R0)            ;;IS THERE ANOTHER NUMBER?
031504  001403                          BEQ     13$             ;;BR IF NO
031506  104401  035250                  TYPE    ,SPACES         ;;TYPE TWO(2) SPACES
031512  000771                          BR      10$             ;;GO BACK TO PRINT THE OCTAL NUMBER
031514  104401  001405          13$:    TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
031520  012600                  14$:    MOV     (SP)+,R0        ;;RESTORE R0
031522  000207                          RTS     PC              ;;RETURN
031524  000000                  MEPITM: .WORD   0               ;&& LOCATION TO STORE 200+ ERROR ITEM NUMBER
```

```
3803                                    .SBTTL  APT COMMUNICATIONS ROUTINE
                                        ;*****************************************************
031526  112737  000001  031772  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
031534  112737  000001  031770  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
031542  000403                          BR      $ATYC
031544  112737  000001  031772  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
031552                          $ATYC:
031552  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
031554  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
031556  105737  031770                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
031562  001450                          BEQ     5$              ;;IF NOT:  BR
031564  122737  000001  001430          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
031572  001031                          BNE     3$              ;;IF NOT:  BR
031574  132737  000100  001431          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
031602  001425                          BEQ     3$              ;;IF NOT:  BR
031604  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
031610  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
031616  005737  001410          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
031622  001375                          BNE     1$              ;;IF NOT:  WAIT
031624  010037  001424                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
031630  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
031632  001376                          BNE     2$
031634  163700  001424                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
031640  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
031642  010037  001426                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
031646  012737  000004  001410          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
031654  000413                          BR      5$
031656  017637  000004  031702  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
031664  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
031672  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
031676  004737  025032                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
031702  000000                  4$:     .WORD   0
031704                          5$:
031704  105737  031772          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
031710  001416                          BEQ     12$             ;;IF NOT:  BR
031712  005737  001430                  TST     $ENV            ;;RUNNING UNDER APT?
031716  001413                          BEQ     12$             ;;IF NOT:  BR
031720  005737  001410          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
031724  001375                          BNE     11$             ;;IF NOT:  WAIT
031726  017637  000004  001412          MOV     @4(SP),$FATAL   ;;GET ERROR #
031734  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
031742  005237  001410                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
031746  105037  031772          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
031752  105037  031771                  CLRB    $LFLG           ;;CLEAR LOG FLAG
031756  105037  031770                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
031762  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
031764  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
031766  000207                          RTS     PC              ;;RETURN
031770     000                  $MFLG:  .BYTE   0               ;;MESSG. FLAG
031771     000                  $LFLG:  .BYTE   0               ;;LOG FLAG
031772     000                  $FFLG:  .BYTE   0               ;;FATAL FLAG
                                        .EVEN
            000200                      APTSIZE=200
            000001                      APTENV=001
            000100                      APTSPOOL=100
            000040                      APTCSUP=040
3804                                    .SBTTL  POWER DOWN AND UP ROUTINE
```

```
                                ;*****************************************************************
                                ;POWER DOWN AND UP ROUTINE
031774  012737  032012  000024  $PWRDN: MOV     #$PWRUP,PWRVEC  ;&& SET UP VECTOR TO RETURN TO THE HALT BELOW
032002  012737  000340  000026          MOV     #LEVEL7,PWRVEC+2;&& RETURN PRIORITY TO 7
032010  000000                        HALT                    ;;HALT PROCESSOR
032012  012737  031774  000024  $PWRUP: MOV     #$PWRDN,PWRVEC  ;&& RESET PWRVEC TO PWRDN ROUTINE AND
032020  012737  000340  000026          MOV     #LEVEL7,PWRVEC+2;&& PRIORITY TO 7
032026  012706  001300                  MOV     #STACK,SP       ;&& REINITIALIZE THE STACK,
032032  012746  000340                  MOV     #LEVEL7,-(SP)   ;&& SET UP RETURN PRIORITY TO 7 AND
032036  012746  000210                  MOV     #STAGIN,-(SP)   ;&& MOVE STAGIN ADDRESS TO STACK AND
032042  005037  001360                  CLR     $TMPO           ;&& CLEAR WAIT LOOP COUNTER
032046  005237  001360          1$:     INC     $TMPO           ;&& GIVE TTY TIME TO RECOVER FROM POWER FAILURE
032052  001375                          BNE     1$              ;&& BRANCH BACK UNTIL ZERO AGAIN
032054  104401  032062                  TYPE    ,$POWER         ;&& TYPE THE POWER FAILURE MESSAGE ASCIZED BELOW
032060  000002                          RTI                     ;&& RETURN TO PROGRAM
032062     200     120     117  $POWER: .ASCIZ  <CRLF>/POWER FAILURE - RESTARTING PROGRAM/<CRLF>
                                        .EVEN
```

```
                                        .SBTTL  MULTIPLE BOARD DIALOGUE ROUTINE
                                ;*****************************************************************
                                ;>>>>>>>MULTIPLE BOARD DIALOGUE ROUTINE<<<<<<<
                                ;*****************************************************************
032130  012706  001300          MBD:    MOV     #STACK,SP       ;INITIALIZE THE STACK
032134  004737  005660                  JSR     PC,SETTUP       ;GO INITIALIZE THE COMMON TAGS
032140  104401  036531                  TYPE    ,MBDIAL         ;TYPE:  'MULTIPLE BOARD DIALOGUE'
032144  105037  030707          PROMPT: CLRB    CHARCT          ;CLEAR LOCATION FOR POSSIBLE INPUT DURING PRINT
032150  104401  036564                  TYPE    ,ECELR          ;TYPE:  'ENTER COMMAND ([E]DIT, [L]IST, [B]URST CALIBRATION,
032154  012703  000001                  MOV     #1,R3           ;EXPECT 1 CHARACTER
032160  004737  002722                  JSR     PC,READ         ;GO READ 1 CHARACTER
032164  022737  000114  002664          CMP     #'L,ANSWER      ;LIST PRESENT TABLE?
032172  001073                          BNE     1$              ;BRANCH IF NO
032174  104401  036340                  TYPE    ,HEADER         ;TYPE:  '# OF    START                                  2-N
                                                                ;       'BOARDS  REGADR  VECADR   W-B     P-LEV   CYCLE    T
032200  013737  001474  002720          MOV     $DDWO,DDW       ;SET UP THE DEVICE DESCRIPTOR WORD FOR PRINTING
032206  013746  002414                  MOV     QTYBRD,-(SP)    ;MOVE NUMBER OF DEVICES TO STACK FOR TYPING
032212  104405                          TYPDS                   ;TYPE THE NUMBER OF DEVICES
032214  104401  035253                  TYPE    ,SPACE3         ;TYPE 3 SPACE CHARACTERS
032220  013746  002416                  MOV     REGADR,-(SP)    ;MOVE THE DEVICE REGISTER ADDRESS TO THE STACK
032224  104402                          TYPOC                   ;TYPE THE DEVICE REGISTER ADDRESS
032226  104401  035257                  TYPE    ,SPACE6         ;TYPE 6 SPACE CHARACTERS
032232  013746  002456                  MOV     VECADR,-(SP)    ;MOVE THE DEVICE VECTOR ADDRESS TO THE STACK
032236  104403                          TYPOS                   ;TYPE THE DEVICE VECTOR ADDRESS
032240     003     000                  .BYTE   3,0             ;TYPE 3 CHARACTERS, LEADING ZEROS SUPPRESSED
032242  104401  035266                  TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
032246  032737  000001  002720          BIT     #BIT0,DDW       ;SEE WHICH W/B STATE FOR BOARDS
032254  001403                          BEQ     10$             ;GO PRINT W STATE IF W
032256  104401  035303                  TYPE    ,B              ;TYPE A 'B'
032262  000402                          BR      11$             ;GO TO NEXT CHECK
032264  104401  035305          10$:    TYPE    ,W              ;TYPE A 'W'
032270  104401  035266          11$:    TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
032274  004737  005630                  JSR     PC,PNTPRI       ;PRINT DEVICE PRIORITY
032300  104401  035266                  TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
032304  032737  000002  002720          BIT     #BIT1,DDW       ;SEE WHICH 2/N STATE FOR BOARDS
032312  001003                          BNE     12$             ;GO PRINT N STATE IF N
032314  104401  035316                  TYPE    ,TWO            ;TYPE A '2'
032320  000402                          BR      13$             ;GO TO NEXT CHECK
032322  104401  035320          12$:    TYPE    ,N              ;TYPE AN 'N'
032326  104401  035266          13$:    TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
032332  032737  000004  002720          BIT     #BIT2,DDW       ;SEE WHICH CABLE STATE FOR BOARDS
032340  001403                          BEQ     14$             ;GO PRINT NO CABLE IF NONE
032342  104401  035312                  TYPE    ,YES            ;TYPE 'YES'
032346  000402                          BR      15$             ;BRANCH TO CONTINUE
032350  104401  035307          14$:    TYPE    ,NO             ;TYPE 'NO'
032354  104401  035333          15$:    TYPE    ,CRLF2          ;TYPE 2 <CRLF>'S
032360  000671                          BR      PROMPT          ;BRANCH TO PROMPT ANOTHER COMMAND
032362  022737  000122  002664  1$:     CMP     #'R,ANSWER      ;RUN PROGRAM?
032370  001020                          BNE     4$              ;BRANCH IF NOT
032372  005737  002416                  TST     REGADR          ;SEE IF REGADR HAS BEEN LOADED
032376  001003                          BNE     3$              ;BRANCH TO CHECK VECADR IF SO
032400  104401  034770          2$:     TYPE    ,MUSTED         ;TYPE:  'DEVICE ADDRESS AND/OR VECTOR TABLE NOT SET UP - MUS
032404  000657                          BR      PROMPT          ;BRANCH BACK FOR PROMPT MESSAGE
032406  005737  002456          3$:     TST     VECADR          ;SEE IF VECADR HAS BEEN LOADED
032412  001772                          BEQ     2$              ;BRANCH BACK TO PRINT ERROR MESSAGE IF NOT
032414  004737  003360                  JSR     PC,FIXTBL       ;FILL TABLE
032420  012737  177777  002670          MOV     #-1,MANSIZE     ;MOVE -1 TO MANSIZE TO INDICATE WE HAVE MANUALLY SIZED
```

```
3863 032426  000137  010272                    JMP     START           ;JUMP TO START
3864 032432  022737  000105  002664  4$:       CMP     #'E,ANSWER      ;EDIT TABLE?
3865 032440  001414                            BEQ     EDIT            ;BRANCH TO EDIT IF SO
3866 032442  022737  000102  002664            CMP     #'B,ANSWER      ;ENTER BURST DATA LATE CALIBRATION?
3867 032450  001235                            BNE     PROMPT          ;BRANCH TO PROMPT IF COMMAND NOT RECOGNIZED
3868 032452  005737  002416                    TST     REGADR          ;SEE IF REGADR HAS BEEN LOADED
3869 032456  001750                            BEQ     2$              ;BRANCH TO ERROR MESSAGE IF NOT
3870 032460  005737  002456                    TST     VECADR          ;SEE IF VECADR HAS BEEN LOADED
3871 032464  001745                            BEQ     2$              ;BRANCH TO ERROR MESSAGE IF NOT
3872 032466  000137  033612                    JMP     BDLCR           ;JUMP TO BURST DATA LATE CALIBRATION ROUTINE
```

```
                                                .SBTTL  TABLE EDIT ROUTINE
3874 032472  104401  035433          EDIT:     TYPE    ,NOBUT          ;TYPE:  'NUMBER OF BOARDS UNDER TEST: '
3875 032476  013746  002414                    MOV     QTYBRD,-(SP)    ;PUT QUANTITY OF BOARDS ON STACK FOR PRINTING
3876 032502  104405                            TYPDS                   ;GO PRINT THE QUANTITY OF BOARDS
3877 032504  104401  035276                    TYPE    ,SPACEC         ;TYPE:  '  : '
3878 032510  012703  000002                    MOV     #2,R3           ;EXPECT MAX OF 2 CHARACTERS
3879 032514  112737  000071  002704            MOVB    #'9,LRGSTC      ;MOVE ASCII '9' TO THE LARGEST CHARACTER LOCATION
3880 032522  004737  002722                    JSR     PC,READ         ;GO READ 2 CHARACTERS
3881 032526  022703  000002                    CMP     #2,R3           ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
3882 032532  001017                            BNE     2$              ;BRANCH IF NOT
3883 032534  022737  000033  002664            CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
3884 032542  001453                            BEQ     5$              ;BRANCH TO PROMPT JUMP IF SO
3885 032544  022737  000003  002664            CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
3886 032552  001447                            BEQ     5$              ;BRANCH TO PROMPT JUMP IF EXIT REQUESTED
3887 032554  022737  000015  002664            CMP     #CARETN,ANSWER  ;SEE IF A <CR> WAS INPUTED
3888 032562  001412                            BEQ     4$              ;IF <CR> USE EXISTING NUMBER
3889 032564  104401  034307          1$:       TYPE    ,BDNERR         ;TYPE:  'ILLEGAL NUMBER (# OTHER THAN 1-16) OR CHARACTER INP
3890 032570  000740                            BR      EDIT            ;BRANCH BACK FOR NEW INPUT
3891 032572  005704                  2$:       TST     R4              ;CHECK FOR ZERO MODULES INPUT
3892 032574  001773                            BEQ     1$              ;BRANCH TO PRINT ERROR MESSAGE IF SO
3893 032576  022704  000020                    CMP     #20,R4          ;SEE IF BOARD NUMBER IS ILLEGAL
3894 032602  100770                            BMI     1$              ;BRANCH TO PRINT ERROR MESSAGE IF SO
3895 032604  010437  002414          3$:       MOV     R4,QTYBRD       ;MOVE INPUTED NUMBER TO QTYBRD
3896 032610  104401  036204          4$:       TYPE    ,SDADRS         ;TYPE:  ' STARTING DEVICE ADDRESS: '
3897 032614  013746  002416                    MOV     REGADR,-(SP)    ;MOVE THE PRESENT ADDRESS TO THE STACK FOR PRINTING
3898 032620  104402                            TYPOC                   ;PRINT THE ADDRESS
3899 032622  104401  035276                    TYPE    ,SPACEC         ;TYPE:  '  : '
3900 032626  012703  000006                    MOV     #6,R3           ;EXPECT MAXIMUM 6 CHARACTERS
3901 032632  112737  000067  002704            MOVB    #'7,LRGSTC      ;MOVE ASCII '7' TO THE LARGEST CHARACTER LOCATION
3902 032640  004737  002722                    JSR     PC,READ         ;GO READ 6 CHARACTERS
3903 032644  022703  000006                    CMP     #6,R3           ;SEE IF NON-NUMERIC CHARACTER INPUTED
3904 032650  001022                            BNE     8$              ;BRANCH IF NOT
3905 032652  022737  000033  002664            CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
3906 032660  001704                            BEQ     EDIT            ;BRANCH AROUND ESC PRINT AND PREVIOUS PROMPT BRANCH IF SO
3907 032662  022737  000003  002664            CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
3908 032670  001005                            BNE     7$              ;BRANCH AROUND PROMPT JUMP IF NOT
3909 032672  000137  032144          5$:       JMP     PROMPT          ;JUMP TO PROMPT A NEW COMMAND
3910 032676  104401  034432          6$:       TYPE    ,ADRERR         ;TYPE:  'ADDRESS INPUTED IS NOT IN THE RANGE 171000 TO 17700
3911 032702  000742                            BR      4$              ;BRANCH BACK FOR REINPUT
3912 032704  022737  000015  002664  7$:       CMP     #CARETN,ANSWER  ;SEE IF <CR> WAS ONLY CHARACTER INPUTED
3913 032712  001417                            BEQ     10$             ;IF <CR> USE EXISTING REG ADDRESS
3914 032714  000735                            BR      4$              ;BRANCH BACK - INPUT NOT LEGAL
3915 032716  022704  171000          8$:       CMP     #171000,R4      ;IS ANSWER BELOW 171000
3916 032722  101365                            BHI     6$              ;BRANCH TO PRINT ERROR MESSAGE IF IT IS
3917 032724  022704  177000                    CMP     #177000,R4      ;IS ANSWER ABOVE 177000
3918 032730  103762                            BLO     6$              ;BRANCH TO PRINT ERROR MESSAGE IF NOT
3919 032732  032704  000007                    BIT     #7,R4           ;TEST TO MAKE SURE A '0' IS PRESENT IN LOWEST OCTAL DIGIT
3920 032736  001403                            BEQ     9$              ;BRANCH AROUND ERROR MESSAGE TYPE IF SO
3921 032740  104401  034600                    TYPE    ,ADLCHR         ;TYPE:  'ADDRESS INPUTED HAS OTHER THAN 0 FOR LEAST'
3922                                                                    ;       'SIGNIFICANT OCTAL DIGIT'
3923 032744  000721                            BR      4$              ;BRANCH BACK FOR REINPUT
3924 032746  010437  002416          9$:       MOV     R4,REGADR       ;INSTALL NEW # IN TABLE
3925 032752  104401  036150          10$:      TYPE    ,SVADRS         ;TYPE:  'STARTING VECTOR ADDRESS:  '
3926 032756  013746  002456                    MOV     VECADR,-(SP)    ;MOVE PRESENT VECTOR TO STACK FOR PRINTING
3927 032762  104403                            TYPOS                   ;PRINT THE PRESENT VECTOR ADDRESS
3928 032764     003     000                    .BYTE   3,0             ;TYPE 3 CHARACTERS, SUPPRESS LEADING ZEROS
3929 032766  104401  035276                    TYPE    ,SPACEC         ;TYPE:  '  : '
```

```
3930 032772  012703  000003                MOV     #3,R3           ;EXPECT ONLY 3 CHARACTERS
3931 032776  004737  002722                JSR     PC,READ         ;GO READ 3 CHARACTERS
3932 033002  022703  000003                CMP     #3,R3           ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
3933 033006  001015                        BNE     11$             ;BRANCH IF NOT
3934 033010  022737  000033  002664        CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
3935 033016  001674                        BEQ     4$              ;BRANCH TO PREVIOUS PROMPT IF SO
3936 033020  022737  000003  002664        CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
3937 033026  001721                        BEQ     5$              ;BRANCH TO PROMPT JUMP IF SO
3938 033030  022737  000015  002664        CMP     #CARETN,ANSWER  ;SEE IF <CR> WAS INPUTED
3939 033036  001417                        BEQ     15$             ;BRANCH IF NO CHANGE WANTED
3940 033040  000744                        BR      10$             ;BRANCH BACK - INPUT WAS ILLEGAL
3941 033042  022704  000123        11$:    CMP     #123,R4         ;SEE IF ANSWER IS BELOW 124
3942 033046  100403                        BMI     13$             ;BRANCH AROUND ERROR MESSAGE IF NOT
3943 033050  104401  034517                TYPE    ,VECERR         ;TYPE:  'VECTOR INPUTED IS NOT IN THE RANGE OF 124 TO 777'
3944 033054  000736                        BR      10$             ;BRANCH BACK FOR REINPUT
3945 033056  032704  000003        13$:    BIT     #3,R4           ;MAKE SURE LEAST SIGNIFICANT OCTAL DIGIT IS '0' OR '4'
3946 033062  001403                        BEQ     14$             ;BRANCH OVER ERROR PRINTING IF NOT
3947 033064  104401  034703                TYPE    ,VCLCHR         ;TYPE:  'VECTOR INPUTED SHOULD HAVE ZERO AS LEAST DIGIT'
3948 033070  000730                        BR      10$             ;BRANCH BACK FOR REINPUT
3949 033072  010437  002456        14$:    MOV     R4,VECADR       ;INSTALL NEW VECTOR ADDRESS IN TABLE
3950 033076  104401  036077        15$:    TYPE    ,DR1WOB         ;TYPE:  'DR11-W OR B (W=0, B=1) CURRENT STATE = '
3951 033102  013737  001474  002720        MOV     $DDWO,DDW       ;MOVE DEVICE DESCRIPTOR WORD TO DDW
3952 033110  012737  000001  002710        MOV     #BIT0,BITTST    ;MOVE BIT STATE TO PRINT TO BITTST
3953 033116  004737  005606                JSR     PC,PSTATE       ;PRINT CURRENT W/B STATE
3954 033122  104401  035276                TYPE    ,SPACEC         ;TYPE:  '  :  '
3955 033126  012703  000001                MOV     #1,R3           ;ONLY INPUT 1 CHARACTER
3956 033132  004737  002722                JSR     PC,READ         ;GO READ 1 CHARACTER
3957 033136  005703                        TST     R3              ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
3958 033140  001415                        BEQ     16$             ;BRANCH AROUND NON-NUMERIC TESTS IF SO
3959 033142  022737  000033  002664        CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
3960 033150  001700                        BEQ     10$             ;BRANCH TO PREVIOUS PROMPT IF SO
3961 033152  022737  000015  002664        CMP     #CARETN,ANSWER  ;SEE IF USER WANTS NO CHANGE
3962 033160  001417                        BEQ     18$             ;BRANCH IF SO
3963 033162  022737  000003  002664        CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
3964 033170  001640                        BEQ     5$              ;BRANCH TO PROMPT JUMP IF SO
3965 033172  000741                        BR      15$             ;BRANCH BACK - INPUT NOT LEGAL
3966 033174  005704                16$:    TST     R4              ;CHECK FOR LEGAL INPUT
3967 033176  001403                        BEQ     17$             ;BRANCH IF OK
3968 033200  022704  000001                CMP     #1,R4           ;CHECK FOR ILLEGAL INPUT
3969 033204  001334                        BNE     15$             ;BRANCH BACK IF ILLEGAL STATE INPUTED
3970 033206  042737  000001  001474  17$:   BIC     #BIT0,$DDWO     ;CLEAR THE BIT TO BE ALTERED
3971 033214  050437  001474                BIS     R4,$DDWO        ;PUT USER INPUT INTO $DDWO
3972 033220  104401  036035        18$:    TYPE    ,DEVPRI         ;TYPE:  'DEVICE PRIORITY PRESENT LEVEL = '
3973 033224  013737  001474  002720        MOV     $DDWO,DDW       ;MOVE DEVICE DESCRIPTOR WORD TO DDW
3974 033232  004737  005630                JSR     PC,PNTPRI       ;PRINT DEVICE PRIORITY
3975 033236  104401  035276                TYPE    ,SPACEC         ;TYPE:  '  :  '
3976 033242  012703  000001                MOV     #1,R3           ;ONLY INPUT 1 CHARACTER
3977 033246  004737  002722                JSR     PC,READ         ;GO READ 1 CHARACTER
3978 033252  005703                        TST     R3              ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
3979 033254  001415                        BEQ     19$             ;BRANCH AROUND NON-NUMERIC TESTS IF NOT
3980 033256  022737  000033  002664        CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
3981 033264  001704                        BEQ     15$             ;BRANCH TO PREVIOUS PROMPT IF SO
3982 033266  022737  000003  002664        CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
3983 033274  001544                        BEQ     26$             ;BRANCH IF EXIT WANTED
3984 033276  022737  000015  002664        CMP     #CARETN,ANSWER  ;SEE IF <CR> INPUTED FOR NO CHANGE WANTED
3985 033304  001413                        BEQ     20$             ;BRANCH IF NO CHANGE WANTED
3986 033306  000744                        BR      18$             ;BRANCH BACK - INPUT NOT LEGAL
```

```
3987 033310  006304                    19$:   ASL     R4              ;PUT PRIORITY IN PROPER POSITION
3988 033312  006304                           ASL     R4              ;BY SHIFTING TO THE LEFT 5 PLACES
3989 033314  006304                           ASL     R4
3990 033316  006304                           ASL     R4
3991 033320  006304                           ASL     R4
3992 033322  042737  000340  001474          BIC     #LEVEL7,$DDW0   ;CLEAR OLD PRIORITY
3993 033330  050437  001474                   BIS     R4,$DDW0        ;SET PRIORITY INTO DEVICE DESCRIPTOR WORD
3994 033334  104401  035747          20$:     TYPE    ,TORNCB         ;TYPE:  '2 OR N CYCLE BURST (2 CY=0, N CY=1) PRESENT STATE =
3995 033340  013737  001474  002720           MOV     $DDW0,DDW       ;MOVE DEVICE DESCRIPTOR WORD TO DDW
3996 033346  012737  000002  002710           MOV     #BIT1,BITTST    ;MOVE BIT STATE TO PRINT TO BITTST
3997 033354  004737  005606                   JSR     PC,PSTATE       ;PRINT 2/N CYCLE STATE
3998 033360  104401  035276                   TYPE    ,SPACEC         ;TYPE:  '  :  '
3999 033364  012703  000001                   MOV     #1,R3           ;ONLY ONE CHARACTER TO INPUT
4000 033370  004737  002722                   JSR     PC,READ         ;READ 1 CHARACTER
4001 033374  005703                           TST     R3              ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
4002 033376  001415                           BEQ     21$             ;BRANCH AROUND NON-NUMERIC TESTS IF NOT
4003 033400  022737  000033  002664           CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
4004 033406  001704                           BEQ     18$             ;BRANCH TO PREVIOUS PROMPT IF SO
4005 033410  022737  000003  002664           CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
4006 033416  001473                           BEQ     26$             ;BRANCH IF USER WANTS TO EXIT
4007 033420  022737  000015  002664           CMP     #CARETN,ANSWER  ;SEE IF USER WANTS NO CHANGE
4008 033426  001414                           BEQ     23$             ;BRANCH IF USER WANTS NO CHANGE
4009 033430  000741                           BR      20$             ;BRANCH BACK - USER INPUT NOT LEGAL
4010 033432  005704                    21$:   TST     R4              ;CHECK FOR LEGAL INPUT
4011 033434  001403                           BEQ     22$             ;BRANCH IF OK
4012 033436  022704  000001                   CMP     #1,R4           ;CHECK FOR ILLEGAL INPUT
4013 033442  001334                           BNE     20$             ;BRANCH BACK IF ILLEGAL STATE INPUTED
4014 033444  006304                    22$:   ASL     R4              ;SHIFT BIT OVER 1 PLACE
4015 033446  042737  000002  001474           BIC     #BIT1,$DDW0     ;CLEAR OLD STATE
4016 033454  050437  001474                   BIS     R4,$DDW0        ;SET THE USERS INPUTED STATE TO $DDW0
4017 033460  104401  036261          23$:     TYPE    ,DOCTS          ;TYPE:  'DO CABLE TESTS (NO=0, YES=1) PRESENT STATE = '
4018 033464  013737  001474  002720           MOV     $DDW0,DDW       ;MOVE DEVICE DESCRIPTOR WORD TO DDW
4019 033472  012737  000004  002710           MOV     #BIT2,BITTST    ;MOVE BIT STATE TO PRINT TO BITTST
4020 033500  004737  005606                   JSR     PC,PSTATE       ;PRINT CABLE STATE
4021 033504  104401  035276                   TYPE    ,SPACEC         ;TYPE:  '  :  '
4022 033510  012703  000001                   MOV     #1,R3           ;INPUT ONLY 1 CHARACTER
4023 033514  004737  002722                   JSR     PC,READ         ;GO INPUT 1 CHARACTER
4024 033520  005703                           TST     R3              ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
4025 033522  001415                           BEQ     24$             ;BRANCH AROUND NON-NUMERIC TESTS IF NOT
4026 033524  022737  000033  002664           CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
4027 033532  001700                           BEQ     20$             ;BRANCH TO PREVIOUS PROMPT IF SO
4028 033534  022737  000003  002664           CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
4029 033542  001421                           BEQ     26$             ;BRANCH IF USER WANTS TO EXIT
4030 033544  022737  000015  002664           CMP     #CARETN,ANSWER  ;SEE IF USER WANTS NO CHANGE
4031 033552  001415                           BEQ     26$             ;BRANCH IF USER WANTS NO CHANGE
4032 033554  000741                           BR      23$             ;BRANCH BACK - USER INPUT NOT LEGAL
4033 033556  005704                    24$:   TST     R4              ;CHECK FOR LEGAL INPUT
4034 033560  001403                           BEQ     25$             ;BRANCH IF OK
4035 033562  022704  000001                   CMP     #1,R4           ;CHECK FOR ILLEGAL INPUT
4036 033566  001334                           BNE     23$             ;BRANCH BACK IF ILLEGAL STATE INPUTED
4037 033570  006304                    25$:   ASL     R4              ;SHIFT INPUTED BIT OVER 2 PLACES
4038 033572  006304                           ASL     R4
4039 033574  042737  000004  001474           BIC     #BIT2,$DDW0     ;CLEAR BIT TO BE CHANGED
4040 033602  050437  001474                   BIS     R4,$DDW0        ;SET THE USERS INPUTED STATE TO $DDW0
4041 033606  000137  032144          26$:     JMP     PROMPT          ;JUMP TO GET NEW DEVICE NUMBER
```

```
                                .SBTTL  BURST DATA LATE CALIBRATION ROUTINE
                                ;*****************************************************************
                                ;>>>>>>>BURST DATA LATE CALIBRATION ROUTINE<<<<<<<
                                ;*****************************************************************

033612  012737  177777  002670  BDLCR:  MOV     #-1,MANSIZE     ;MOVE -1 TO MANSIZE
033620  004737  003360                  JSR     PC,FIXTBL       ;GO FILL TABLE
033624  104401  035110                  TYPE    ,BDLCRM         ;TYPE:  'BURST DATA LATE CALIBRATION'
                                                                ;TYPE:  'ATTACH SCOPE PROBE...'
                                                                ;       'TO CALIBRATE NEXT BOARD, TYPE ANY CHARACTER'
033630  012737  000001  002544          MOV     #BIT0,DEVMSK    ;SET UP BIT MASK TO TEST $DEVM FOR DEVICES
033636  012700  002456                  MOV     #VECADR,R0      ;MOVE VECADR TO R0
033642  012701  002416                  MOV     #REGADR,R1      ;MOVE REGADR TO R1
033646  005037  001422                  CLR     $UNIT           ;CLEAR $UNIT
033652  033737  002544  001466  2$:     BIT     DEVMSK,$DEVM    ;CHECK TO SEE IF DEVICE IS TO BE TESTED
033660  001015                          BNE     5$              ;BRANCH IF SO
033662  005737  002544                  TST     DEVMSK          ;SEE IF BIT 15 IS SET
033666  100004                          BPL     4$              ;BRANCH TO CONTINUE IF NOT SET
033670  104401  034377                  TYPE    ,BCDONE         ;TYPE:  'BURST CALIBRATION COMPLETE'
033674  000137  032144                  JMP     PROMPT          ;JUMP TO PROMPT A NEW COMMAND
033700  022021                  4$:     CMP     (R0)+,(R1)+     ;INCREMENT THE TWO POINTERS
033702  006337  002544                  ASL     DEVMSK          ;UPDATE DEVICE MAP TEST MASK
033706  005237  001422                  INC     $UNIT           ;INCREMENT UNIT NUMBER
033712  000757                          BR      2$              ;GO TEST NEXT BIT OF DEVICE MASK
033714  011137  002522                  MOV     (R1),CSR        ;PUT UUT CSR ADDRESS INTO DEVICE CSR LOCATION
033720  062737  000004  002522          ADD     #4,CSR          ;POINT CSR TO CSR ADDRESS
033726  011037  002526                  MOV     (R0),DRINV      ;PUT UUT VECTOR ADDRESS INTO DEVICE DRINV
033732  104401  035076                  TYPE    ,DEVICE         ;TYPE:  'DEVICE # '
033736  013746  001422                  MOV     $UNIT,-(SP)     ;PUT UNIT NUMBER ON STACK FOR TYPEOUT
033742  104405                          TYPDS                   ;GO TYPE THE UNIT NUMBER IN DECIMAL
033744  104401  035407                  TYPE    ,UCAL           ;TYPE:  ' UNDER CALIBRATION'
033750  004737  004036                  JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
033754  005077  146542          6$:     CLR     @CSR            ;CLR CYCLE BIT
033760  012737  000077  002660          MOV     #77,TIME        ;MOVE WAIT LOOP COUNTER TO TIME
033766  052777  000400  146526          BIS     #CY,@CSR        ;SET CYCLE BIT
033774  005337  002660          7$:     DEC     TIME            ;SUBTRACT 1 FROM TIME UNTIL ZERO
034000  001375                          BNE     7$              ;BRANCH BACK IF NOT ZERO YET
034002  105777  145336                  TSTB    @$TKS           ;IS A CHARACTER WAITING INDICATING USER WANTS TO GO ON?
034006  100362                          BPL     6$              ;BRANCH IF NOT
034010  017737  146622  002660          MOV     @TKB,TIME       ;WASTE THE CHARACTER, CLEARING THE CHARACTER FLAG
034016  000730                          BR      4$              ;GO ON TO NEXT BOARD
```

```
                                                .SBTTL  ASCII AND ASCIZ MESSAGES AND LOCATIONS
 4084 034020  200  123  124  STKIFL: .ASCIZ  <CRLF>/STACK IS FULL - DATA MAY HAVE BEEN LOST/<CRLF><CRLF>
 4085 034073  136  131  200  ERCHDR: .ASCII  /^Y/<CRLF>/SUMMATION OF ERRORS SINCE START OR LAST REPORT/
 4086 034154  200  200  102           .ASCIZ  <CRLF><CRLF>/BOARD #   PASS #   ERRTTL/<CRLF>
 4087 034211  136  131  200  NODATA: .ASCIZ  /^Y/<CRLF>/NO ERROR TOTALS TO REPORT/<CRLF><CRLF>
 4088 034250  040  055  040  ETDEV:  .ASCIZ  / - TOTAL ERRORS THIS DEVICE = /
 4089 034307  111  114  114  BDNERR: .ASCIZ  /ILLEGAL NUMBER (# OTHER THAN 1-16) OR CHARACTER INPUTED/
 4090 034377  102  125  122  BCDONE: .ASCIZ  /BURST CALIBRATION COMPLETE/
 4091 034432  101  104  104  ADRERR: .ASCIZ  /ADDRESS INPUTED IS NOT IN THE RANGE 171000 TO 177000/
 4092 034517  126  105  103  VECERR: .ASCIZ  /VECTOR INPUTED IS NOT IN THE RANGE OF 124 TO 777/
 4093 034600  101  104  104  ADLCHR: .ASCIZ  /ADDRESS INPUTED HAS OTHER THAN 0 FOR LEAST SIGNIFICANT OCTAL DIGIT/
 4094 034703  126  105  103  VCLCHR: .ASCIZ  /VECTOR INPUTED SHOULD HAVE ZERO AS LEAST DIGIT/
 4095 034762  074  105  123  ESCAPE: .ASCIZ  /<ESC>/
 4096 034770  200  104  105  MUSTED: .ASCII  <CRLF>/DEVICE ADDRESS AND-OR VECTOR TABLE NOT SET UP - /
 4097 035051  115  125  123           .ASCIZ  /MUST EDIT FIRST/
 4098 035071  040  000       LETNCR: .ASCIZ  / /
 4099 035073  136  103  000  CNTRLC: .ASCIZ  /^C/
 4100 035076  104  105  126  DEVICE: .ASCIZ  /DEVICE # /
 4101 035110  200  102  125  BDLCRM: .ASCII  <CRLF>/BURST DATA LATE CALIBRATION/
 4102 035144  200  101  124           .ASCII  <CRLF>/ATTACH SCOPE PROBE.../
 4103 035172  200  124  117           .ASCIZ  <CRLF>/TO CALIBRATE NEXT BOARD, TYPE ANY CHARACTER/<CRLF>
 4104 035250  040  040  000  SPACES: .ASCIZ  /  /
 4105 035253  040  040  040  SPACE3: .ASCIZ  /   /
 4106 035257  040  040  040  SPACE6: .ASCIZ  /      /
 4107 035266  040  040  040  SPACE7: .ASCIZ  /       /
 4108 035276  040  040  072  SPACEC: .ASCIZ  /  : /
 4109 035303  102  000       B:      .ASCIZ  /B/
 4110 035305  127  000       W:      .ASCIZ  /W/
 4111 035307  116  117  000  NO:     .ASCIZ  /NO/
 4112 035312  131  105  123  YES:    .ASCIZ  /YES/
 4113 035316  062  000       TWO:    .ASCIZ  /2/
 4114 035320  116  000       N:      .ASCIZ  /N/
 4115 035322  102  117  101  BOARD:  .ASCIZ  /BOARD # /
 4116 035333  200  200  000  CRLF2:  .ASCIZ  <CRLF><CRLF>
 4117 035336  200  101  114  BCFIN:  .ASCIZ  <CRLF>/ALL BOARDS CALIBRATED - BEGINNING TEST/<CRLF>
 4118 035407  040  125  116  UCAL:   .ASCIZ  / UNDER CALIBRATION/<CRLF>
 4119 035433  200  116  125  NOBUT:  .ASCIZ  <CRLF>/NUMBER OF BOARDS UNDER TEST: /
 4120 035472  200  200  104  BRVWPC: .ASCII  <CRLF><CRLF>/DIAGNOSTIC HAS DETERMINED THE FOLLOWING ABOUT THE/<CRLF>
 4121 035556  104  122  061           .ASCII  /DR11-W(S) IT HAS FOUND.  USER *MUST* DETERMINE ACCURACY/<CRLF><CRLF>
 4122 035647  040  040  040           .ASCIZ  /   BOARD#   REGADR    VECADR     W-B    P-LEV   2-N CY   CABLE/<CRLF>
 4123 035747  200  062  040  TORNCB: .ASCIZ  <CRLF>/2 OR N CYCLE BURST (2 CY=0, N CY=1) PRESENT STATE = /
 4124 036035  200  104  105  DEVPRI: .ASCIZ  <CRLF>/DEVICE PRIORITY PRESENT LEVEL = /
 4125 036077  200  104  122  DR1WOB: .ASCIZ  <CRLF>/DR11-W OR B (W=0, B=1) CURRENT STATE = /
 4126 036150  200  123  124  SVADRS: .ASCIZ  <CRLF>/STARTING VECTOR ADDRESS:  /
 4127 036204  200  123  124  SDADRS: .ASCIZ  <CRLF>/STARTING DEVICE ADDRESS: /
 4128 036237  040  124  105  TSTCOM: .ASCIZ  / TESTING COMPLETE/
 4129 036261  200  104  117  DOCTS:  .ASCIZ  <CRLF>/DO CABLE TESTS (NO=0, YES=1) PRESENT STATE = /
 4130 036340  200  200  043  HEADER: .ASCII  <CRLF><CRLF>/# OF     START                                  2-N    CABLE/
 4131 036434  200  102  117           .ASCIZ  <CRLF>/BOARDS   REGADR   VECADR      W-B    P-LEV   CYCLE   TESTS/<CRLF>
 4132 036531  200  200  115  MBDIAL: .ASCIZ  <CRLF><CRLF>/MULTIPLE BOARD DIALOGUE/<CRLF>
 4133 036564  200  105  116  ECELR:  .ASCIZ  <CRLF>/ENTER COMMAND ([E]DIT, [L]IST, [B]URST CALIBRATION, [R]UN):  /
 4134 036663  124  110  105  NOMORE: .ASCII  /THERE ARE STILL MORE ERRORS, BUT WILL NOT BE PRINTED./<CRLF>
 4135 036751  105  122  122           .ASCIZ  /ERRORS WILL STILL BE COUNTED AND PRINTED AT THE EOP./<CRLF>
 4136 037037  200  116  117  NODVPR: .ASCII  <CRLF>/NO DEVICES RECOGNIZED - DIAGNOSTIC CANNOT BE RUN/<CRLF>
 4137 037121  122  105  123           .ASCIZ  /RESTART AT 204 IF A DEVICE IS PRESENT/<CRLF>
 4138 037170  200  050  136  M1:     .ASCII  <CRLF>/(^X) INHIBITS EOP'S, (^Y) FOR ERROR SUMMARY/<CRLF>
 4139 037245  125  116  111           .ASCIZ  /UNIBUS HANG?  RESTART AT ADDRESS /
```

```
4140 037307  200  200  103  M1A:     .ASCIZ  <CRLF><CRLF>/CZDRLC0 DR11 GEN NPR INTFC LOGIC TEST/<CRLF>
4141 037360  104  105  126  BARADR:  .ASCIZ  /DEVICE ADDRESS - /
4142 037402  054  040  124  TSNUMB:  .ASCIZ  /, TEST NUMBER - /
4143 037423  054  040  120  PASNUM:  .ASCIZ  /, PASS NUMBER - /
4144                         .EVEN
4145 037444  000000         .SAV:    .WORD   0
4146 037446                           .BLKW   400
4147 040446  000000         BUFF:    .WORD   0
4148 040450  040450         XINBUF:  .
4149 040452                           .BLKW   400
4150 041452  041452         XCHKBU:  .
4151 041454                           .BLKW   400
4152 042454  042456         CAPNTR:  .WORD   CAPSTK          ;LOCATION TO HOLD POINTER FOR CAPTURE STACK
4153 042456                 CAPSTK:  .BLKW   600.            ;LOCATIONS TO STORE UP TO 150 DECIMAL PASSES AND THEIR ERROR
4154 044736  000000         ENDSTK:  .WORD   0               ;FLAG SIGNALING END OF THE STACK
```

```
                                                .SBTTL  ERROR MESSAGES
044740  124  105  123  EM1:    .ASCIZ  /TEST SEQUENCING ERROR/
044766  103  101  116  EM2:    .ASCIZ  /CANNOT ACCESS DR11 REGISTER/
045022  104  122  061  EM3:    .ASCIZ  /DR11-B OR W MODE INCORRECT (0=B, 1=W)/
045070  111  116  111  EM4:    .ASCIZ  /INIT FAILED TO CLEAR WCR/
045121  111  116  111  EM5:    .ASCIZ  /INIT FAILED TO CLEAR BAR/
045152  111  116  111  EM6:    .ASCIZ  /INIT FAILED TO CLEAR BDR/
045203  111  116  111  EM7:    .ASCIZ  /INIT FAILED TO CLEAR ALL CSR R-W BITS/
045251  122  105  123  EM10:   .ASCIZ  /RESET FAILED TO CLEAR WCR/
045303  101  124  124  EM11:   .ASCIZ  /ATTEMPT TO SET ALL WCR BITS FAILED/
045346  122  105  123  EM12:   .ASCIZ  /RESET FAILED TO CLEAR BAR/
045400  101  124  124  EM13:   .ASCIZ  /ATTEMPT TO SET ALL BAR BITS TO 1 FAILED/
045450  103  123  122  EM14:   .ASCIZ  /CSR BIT TEST FAILED (FATAL - DIAGNOSTIC NOT CONTINUED)/
045537  103  123  122  EM15:   .ASCIZ  /CSR BIT TEST FAILED/
045563  105  111  122  EM16:   .ASCIZ  /EIR BIT TEST FAILED/
045607  122  105  101  EM17:   .ASCIZ  /READY AND MAINTENANCE ARE NOT THE ONLY BITS SET IN CSR/
045676  101  124  124  EM20:   .ASCIZ  /ATTN AND ERROR FAILED TO SET PROPERLY/
045744  101  124  124  EM21:   .ASCIZ  /ATTN AND ERROR FAILED TO CLEAR PROPERLY/
046014  105  122  122  EM22:   .ASCIZ  /ERROR BIT SHOULD HAVE BEEN CLEAR/
046055  102  111  124  EM23:   .ASCIZ  /BIT PATTERN NOT LOADED PROPERLY IN WCR/
046124  122  105  101  EM24:   .ASCIZ  /READY OF CSR WAS NOT SET/
046155  102  111  124  EM25:   .ASCIZ  /BIT 0 OF THE BAR WAS SET/
046206  102  111  124  EM26:   .ASCIZ  /BIT PATTERN TEST FAILED IN BAR/
046245  127  103  122  EM27:   .ASCIZ  /WCR DATA PATTERN NOT CORRECT/
046302  106  125  116  FM30:   .ASCIZ  /FUNCTION BIT(S) ARE NOT CLEAR/
046340  104  123  124  EM31:   .ASCIZ  /DSTAT A, B OR C ARE NOT AS EXPECTED/
046404  102  104  122  EM32:   .ASCIZ  /BDR IS NOT CLEAR/
046425  101  114  114  EM33:   .ASCIZ  /ALL BDR BITS ARE NOT SET/
046456  102  104  122  EM34:   .ASCIZ  /BDR FAILS TO HOLD A BIT PATTERN/
046516  102  104  122  EM35:   .ASCIZ  /BDR SHOULD NOT HAVE BEEN LOADED WITH NEW PATTERN/
046577  102  104  122  EM36:   .ASCIZ  /BDR PATTERN NOT CORRECT/
046627  122  105  101  EM37:   .ASCIZ  /READY IS NOT THE ONLY BIT SET/
046665  122  105  101  EM40:   .ASCIZ  /READY SHOULD NOT BE SET/
046715  122  105  101  EM41:   .ASCIZ  /READY WAS CLEARED BUT NEVER SET AGAIN/
046763  122  105  101  EM42:   .ASCIZ  /READY CANNOT BE SET BY INIT/
047017  104  122  061  EM43:   .ASCIZ  /DR11 FAILED TO INTERRUPT/
047050  104  122  061  EM44:   .ASCIZ  /DR11 INTERRUPTED, BUT IT SHOULDN'T HAVE/
047120  105  122  122  EM45:   .ASCIZ  /ERROR BIT SHOULD NOT BE CLEAR/
047156  106  125  116  EM46:   .ASCIZ  /FUNCTION BITS DIDN'T INCREMENT IN MAINT MODE/
047233  103  123  122  EM47:   .ASCIZ  /CSR IS WRONG/
047250  124  122  101  EM50:   .ASCIZ  /TRANSFERS SHOULD HAVE BEEN INHIBITED/
047315  104  122  061  EM51:   .ASCIZ  /DR11 SHOULD NOT HAVE INTERRUPTED A SECOND TIME/
047374  105  130  120  EM52:   .ASCIZ  /EXPECTED INTERRUPT DID NOT OCCUR/
047435  127  103  122  EM53:   .ASCIZ  /WCR NOT EQUAL TO ZERO/
047463  102  101  122  EM54:   .ASCIZ  /BAR IS WRONG/
047500  102  101  104  EM55:   .ASCIZ  /BAD DATA IN BDR/
047520  104  101  124  EM56:   .ASCIZ  /DATA NOT TRANSFERED CORRECTLY/
047556  102  125  106  EM57:   .ASCIZ  /BUFFER DATA NOT CORRECT/
047606  124  117  117  EM60:   .ASCIZ  /TOO MANY WORDS WERE TRANSFERED/
047645  125  116  105  EM61:   .ASCIZ  /UNEXPECTED TRAP OR INTERRUPT TO TRAP ADDRESS BELOW/
047730  103  123  122  EM62:   .ASCIZ  /CSR AND-OR WCR AND-OR BAR ARE INCORECT/
047777  104  122  061  EM63:    ASCIZ  /DR11 INTERRUPTED AT WRONG LEVEL/
050037  062  055  116  EM65:   .ASCIZ  /2-N CYCLE BURST SWITCH IN WRONG POSITION/
050110  115  125  114  EM66:   .ASCIZ  /MULTICYCLE BIT IN THE EIR IS WRONG/
050153  103  123  122  EM202:  .ASCIZ  /CSR PATTERN NOT CORRECT/
050203  102  104  122  EM211:  .ASCIZ  /BDR AND-OR WCR AND-OR BAR ARE INCORECT/
```

```
                                                .SBTTL  DATA HEADERS
4212 050252  105  130  120  DH1:    .ASCII  /EXPCTD  RECVD/<CRLF>
4213 050270  124  105  123          .ASCIZ  /TEST #  TEST #/
4214 050307  124  105  123  DH2:    .ASCIZ  /TEST #  ERR PC  ABRTPC  REGISTER/
4215 050350  124  105  123  DH3:    .ASCIZ  /TEST #  ERR PC  EXPMOD  ACTMOD  CSRADR/
4216 050417  124  105  123  DH4:    .ASCIZ  /TEST #  ERR PC  WCRADR  WCRCONTENTS/
4217 050463  124  105  123  DH5:    .ASCIZ  /TEST #  ERR PC  BARADR  BAREXP  BARRCV/
4218 050532  124  105  123  DH6:    .ASCIZ  /TEST #  ERR PC  BDRADR  BDRCONTENTS/
4219 050576  124  105  123  DH7:    .ASCIZ  /TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS/
4220 050652  040  040  040  DH14:   .ASCII  /                BIT(S)/<CRLF>
4221 050701  124  105  123          .ASCIZ  /TEST #  ERR PC  TESTED  CSRADR  CSREXP  CSRCONTENTS/
4222 050765  040  040  040  DH16:   .ASCII  /                BIT(S)/<CRLF>
4223 051014  124  105  123          .ASCIZ  /TEST #  ERR PC  TESTED  EIRADR  EIREXP  EIRCONTENTS/
4224 051100  124  105  123  DH17:   .ASCIZ  /TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS/
4225 051154  124  105  123  DH23:   .ASCIZ  /TEST #  ERR PC  WCRADR  WCREXP  WCRCONTENTS/
4226 051230  124  105  123  DH26:   .ASCIZ  /TEST #  ERR PC  BARADR  BAREXP  BARCONTENTS/
4227 051304  124  105  123  DH34:   .ASCIZ  /TEST #  ERR PC  BDRADR  BDREXP  BDRCONTENTS/
4228 051360  124  105  123  DH43:   .ASCIZ  /TEST #  ERR PC  CSRADR  CSRCONTENTS/
4229 051424  124  105  123  DH50:   .ASCIZ  /TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  BARADR  BAREXP  BARRCV/
4230 051523  124  105  123  DH56:   .ASCIZ  /TEST #  ERR PC  NPR1AD  NPR1EX  NPR1RC  CSRADR/
4231 051602  040  040  040  DH57:   .ASCII  /                CHECK   CHECK   INPUT   INPUT/<CRLF>
4232 051660  124  105  123          .ASCIZ  /TEST #  ERR PC  BUFADR  BUFDAT  BUFADR  BUFDAT  CSRADR/
4233 051747  040  040  040  DH60:   .ASCII  /                DIDNOT/<CRLF>
4234 051776  124  105  123          .ASCIZ  /TEST #  ERR PC  EXPECT  ADRESS  CSRADR/
4235 052045  124  105  123  DH61:   .ASCIZ  /TEST #  ERR PC  WCRADR  OLDPC   TRAP ADR/
4236 052116  124  105  123  DH62:   .ASCII  /TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  CSREXP  /
4237 052176  103  123  122          .ASCIZ  /CSRRCV  BAREXP  BARRCV/
4238 052225  124  105  123  DH63:   .ASCIZ  /TEST #  ERR PC  EXPLVL  RCVLVL  CSRADR/
4239 052274  124  105  123  DH64:   .ASCIZ  /TEST #  ERR PC  EXPLVL  CSRADR/
4240 052333  124  105  123  DH65:   .ASCIZ  /TEST #  ERR PC  CSRADR  EIREXP  EIRRCV/
4241 052402  124  105  123  DH66:   .ASCIZ  /TEST #  ERR PC  PATERN  CSRADR  CSRRCV/
4242 052451  124  105  123  DH202:  .ASCIZ  /TEST #  ERR PC  CSRADR  PATLDD  CSREXP  CSRRCV/
4243 052530  124  105  123  DH203:  .ASCIZ  /TEST #  ERR PC  CSRADR  PATERN  CSREXP  CSRCONTENTS/
4244 052614  124  105  123  DH207:  .ASCIZ  /TEST #  ERR PC  PATERN  CSRADR  CSRCONTENTS/
4245 052670  124  105  123  DH210:  .ASCIZ  /TEST #  ERR PC  WCRADR  WCRCONTENTS/
4246 052734  124  105  123  DH211:  .ASCIZ  /TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  BDREXP  BDRRCV  BAREXP  BARRCV/
4247                                    .EVEN
```

```
                                                .SBTTL  DATA TABLES
053044  001362  001414  000000  DT1:    .WORD   $TMP1,$TESTN,0
053052  001414  001316  002672  DT2:    .WORD   $TESTN,$ERRPC,OLDPC1,DREG,0
053064  001414  001316  001362  DT3:    .WORD   $TESTN,$ERRPC,$TMP1,BORW,CSR,0
053100  001414  001316  002516  DT4:    .WORD   $TESTN,$ERRPC,WCR,RWCR,0
053112  001414  001316  002520  DT5:    .WORD   $TESTN,$ERRPC,BAR,EBAR,RBAR,0
053126  001414  001316  002524  DT6:    .WORD   $TESTN,$ERRPC,BDR,RBDR,0
053140  001414  001316  002522  DT7:    .WORD   $TESTN,$ERRPC,CSR,ECSR,RCSR,0
053154  001414  001316  002536  DT14:   .WORD   $TESTN,$ERRPC,BUT,CSR,ECSR,RCSR,0
053172  001414  001316  002536  DT16:   .WORD   $TESTN,$ERRPC,BUT,CSR,EEIR,REIR,0
053210  001414  001316  002522  DT17:   .WORD   $TESTN,$ERRPC,CSR,ECSR,RCSR,0
053224  001414  001316  002516  DT23:   .WORD   $TESTN,$ERRPC,WCR,EWCR,RWCR,0
053240  001414  001316  002520  DT26:   .WORD   $TESTN,$ERRPC,BAR,EBAR,RBAR,
053254  001414  001316  002524  DT34:   .WORD   $TESTN,$ERRPC,BDR,EBDR,RBDR,0
053270  001414  001316  002522  DT43:   .WORD   $TESTN,$ERRPC,CSR,RCSR,0
053302  001414  001316  002516  DT50:   .WORD   $TESTN,$ERRPC,WCR,EWCR,RWCR,BAR,EBAR,RBAR,0
053324  001414  001316  002606  DT56:   .WORD   $TESTN,$ERRPC,ANPR1,ENPR1,NPR1,CSR,0
053342  001414  001316  001370  DT57:   .WORD   $TESTN,$ERRPC,$TMP4,$TMP2,$TMP5,$TMP3,CSR,0
053362  001414  001316  001364  DT60:   .WORD   $TESTN,$ERRPC,$TMP2,$TMP3,CSR,0
053376  001414  001316  002516  DT61:   .WORD   $TESTN,$ERRPC,WCR,OLDPC2,BDVECT,0
053412  001414  001316  002516  DT62:   .WORD   $TESTN,$ERRPC,WCR,EWCR,RWCR,ECSR,RCSR,EBAR,RBAR,0
053436  001414  001316  002552  DT63:   .WORD   $TESTN,$ERRPC,DRLEV,LEVEL,CSR,0
053452  001414  001316  001362  DT64:   .WORD   $TESTN,$ERRPC,$TMP1,CSR,0
053464  001414  001316  002522  DT65:   .WORD   $TESTN,$ERRPC,CSR,EEIR,REIR,0
053500  001414  001316  002604  DT66:   .WORD   $TESTN,$ERRPC,ENPR1,CSR,RCSR,0
053514  001414  001316  002522  DT202:  .WORD   $TESTN,$ERRPC,CSR,BUT,ECSR,RCSR,0
053532  001414  001316  002522  DT203:  .WORD   $TESTN,$ERRPC,CSR,$TMP0,ECSR,RCSR,0
053550  001414  001316  001362  DT207:  .WORD   $TESTN,$ERRPC,$TMP1,CSR,RCSR,0
053564  001414  001316  002516  DT210:  .WORD   $TESTN,$ERRPC,WCR,RWCR,0
053576  001414  001316  002516  DT211:  .WORD   $TESTN,$ERRPC,WCR,EWCR,RWCR,EBDR,RBDR,EBAR,RBAR,0
```

```
                                                .SBTTL  BUS HANG ROUTINE
053622  104401  037360          UBHANG. TYPE    ,BARADR                 ;TYPE:  'DEVICE ADDRESS - '
053626  013746  002516                  MOV     WCR,-(SP)               ;PUT DEVICE ADDRESS ON STACK
053632  104402                          TYPOC                           ;GO TYPE IT IN OCTAL
053634  104401  037402                  TYPE    ,TSNUMB                 ;TYPE:  ', TEST NUMBER - '
053640  013746  001414                  MOV     $TESTN,-(SP)            ;PUT TEST NUMBER ON STACK
053644  104403                          TYPOS                           ;GO TYPE IT IN OCTAL
053646     002     000                  .BYTE   2,0                     ;TYPE 2 DIGITS, LEADING ZEROS SUPRESSED
053650  104401  037423                  TYPE    ,PASNUM                 ;TYPE:  ', PASS NUMBER - '
053654  013746  002716                  MOV     $PASS2,-(SP)            ;MOVE OVERFLOW NUMBER TO THE STACK
053660  013746  001416                  MOV     $PASS,-(SP)             ;PUT PASS NUMBER ON STACK
053664  104406                          TYPDE                           ;GO TYPE IT IN EXTENDED DECIMAL
053666  104401  001405                  TYPE    ,$CRLF                  ;TYPE A <CRLF>
053672  000000                          HALT                            ;WHOA - YOU GOTTA SERIOUSA PROBLEMA, BUDDY!
053674  000137  010266                  JMP     START1                  ;JUMP TO RESTART PROGRAM
053700  000000          NOCARE: .WORD   0       ;LOCATION FOR USE WHENEVER CYCLE BIT OF CSR IS USED. THIS
                                                ;SHOULD *ALWAYS* BE THE LAST WORD LOCATION IN THIS DIAGNOSTIC

        000001                  .END
```

SYMBOL TABLE

```
ABASE = 172410        AVECT1= 000300        CARETN= 000015        DATCHK  003716        DT1     053044
ACDW1 = 000000        AVECT2= 000000        CAT   = 157777        DATCH1  003736        DT14    053154
ACDW2 = 000000        B       035303        CATCH   005536        DATCH2  004014        DT16    053172
ACPUOP= 000000        BAR     002520        CBIT0 = 177776        DATOCK  004106        DT17    053210
ADDR    002646        BARADR  037360        CBIT1 = 177775        DATOC1  004126        DT2     053052
ADDW0 = 000000        BCDONE  034377        CBIT10= 175777        DATOC2  004170        DT202   053514
ADDW1 = 000000        BCFIN   035336        CBIT11= 173777        DBC   = 003000        DT203   053532
ADDW10= 000000        BDFAIL  002666        CBIT12= 167777        DDISP = 177570        DT207   053550
ADDW11= 000000        BDLCR   033612        CBIT13= 157777        DDW     002720        DT210   053564
ADDW12= 000000        BDLCRM  035110        CBIT14= 137777        DEVADS  004344        DT211   053576
ADDW13= 000000        BDNERR  034307        CBIT15= 077777        DEVICE  035076        DT23    053224
ADDW14= 000000        BDR     002524        CBIT2 = 177773        DEVMSK  002544        DT26    053240
ADDW15= 000000        BDVECT  002542        CBIT3 = 177767        DEVPRI  036035        DT3     053064
ADDW2 = 000000        BEGIN   010466        CBIT4 = 177757        DH1     050252        DT34    053254
ADDW3 = 000000        BEGIN1  011006        CBIT5 = 177737        DH14    050652        DT4     053100
ADDW4 = 000000        BITTST  002710        CBIT6 = 177677        DH16    050765        DT43    053270
ADDW5 = 000000        BIT0  = 000001        CBIT7 = 177577        DH17    051100        DT5     053112
ADDW6 = 000000        BIT00 = 000001        CBIT8 = 177377        DH2     050307        DT50    053302
ADDW7 = 000000        BIT01 = 000002        CBIT9 = 176777        DH202   052451        DT56    053324
ADDW8 = 000000        BIT02 = 000004        CB1513= 057777        DH203   052530        DT57    053342
ADDW9 = 000000        BIT03 = 000010        CCY   = 177377        DH207   052614        DT6     053126
ADEVCT= 000000        BIT04 = 000020        CDAB  = 171777        DH210   052670        DT60    053362
ADEVM = 000000        BIT05 = 000040        CDAC  = 172777        DH211   052734        DT61    053376
ADLCHR  034600        BIT06 = 000100        CDBC  = 174777        DH23    051154        DT62    053412
ADRERR  034432        BIT07 = 000200        CDSA  = 173777        DH26    051230        DT63    053436
AENV  = 000000        BIT08 = 000400        CDSB  = 175777        DH3     050350        DT64    053452
AENVM = 000000        BIT09 = 001000        CDSC  = 176777        DH34    051304        DT65    053464
AFATAL= 000000        BIT1  = 000002        CDST  = 170777        DH4     050417        DT66    053500
AMADR1= 000000        BIT10 = 002000        CEIR  = 077777        DH43    051360        DT7     053140
AMADR2= 000000        BIT11 = 004000        CER   = 077777        DH5     050463        EBAR    002600
AMADR3= 000000        BIT12 = 010000        CFNC  = 177761        DH50    051424        EBDR    002576
AMADR4= 000000        BIT13 = 020000        CF1   = 177775        DH56    051523        ECELR   036564
AMAMS1= 000000        BIT14 = 040000        CF2   = 177773        DH57    051602        ECSR    002572
AMAMS2= 000000        BIT15 = 100000        CF3   = 177767        DH6     050532        EDIT    032472
AMAMS3= 000000        BIT2  = 000004        CGO   = 177776        DH60    051747        EEIR    002574
AMAMS4= 000000        BIT3  = 000010        CHARCT  030707        DH61    052045        EIR   = 100000
AMSGAD= 000000        BIT4  = 000020        CHKBFF  003520        DH62    052116        EMTVEC= 000030
AMSGLG= 000000        BIT5  = 000040        CHKBUF  002620        DH63    052225        EM1     044740
AMSGTY= 000000        BIT6  = 000100        CHKCAB  004060        DH64    052274        EM10    045251
AMTYP1= 000000        BIT7  = 000200        CHK4DR  005104        DH65    052333        EM11    045303
AMTYP2= 000000        BIT8  = 000400        CIE   = 177677        DH66    052402        EM12    045346
AMTYP3= 000000        BIT9  = 001000        CKSWR = 104410        DH7     050576        EM13    045400
AMTYP4= 000000        BOARD   035322        CLENUP  004036        DIOMEM  002614        EM14    045450
ANPR1   002606        BORW    002610        CMA   = 167777        DISPLA  001342        EM15    045537
ANSWER  002664        BPINIT  004374        CNTLC = 000003        DISPRE  000174        EM16    045563
APASS = 000000        BPT   = 000003        CNTRLC  035073        DOCTS   036261        EM17    045607
APRIOR= 000000        BPTINT  004436        CNX   = 137777        DREG    002550        EM2     044766
APTCSU= 000040        BPTVCT  000014        CR    = 000015        DRGET   004452        EM20    045676
APTENV= 000001        BPTVEC= 000014        CRLF  = 000200        DRINV   002526        EM202   050153
APTSIZ= 000200        BRVWPC  035472        CRLF2   035333        DRLEV   002552        EM21    045744
APTSPO= 000100        BRWAIT  002626        CRY   = 177577        DRVS    002530        EM211   050203
ASIZE   005274        BUFF    040446        CSR     002522        DR1WOB  036077        EM22    046014
ASWREG= 000000        BUFLEN  002622        CX6   = 177757        DSA   = 004000        EM23    046055
AT    = 020000        BUSERR= 000004        CX7   = 177737        DSB   = 002000        EM24    046124
ATESTN= 000000        BUT     002536        CY    = 000400        DSC   = 001000        EM25    046155
AUNIT = 000000        CAPNTR  042454        DAB   = 006000        DST   = 007000        EM26    046206
AUSWR = 000000        CAPSTK  042456        DAC   = 005000        DSWR  = 177570        EM27    046245
```

```
EM3     045022          F1      = 000002        NXTTST  002554          STACK = 001300          TST12   014270
EM30    046302          F2      = 000004        N2      = 000400        STAGIN  000210          TST13   014454
EM31    046340          F3      = 000010        OFL     002702          START   010272          TST14   014640
EM32    046404          GO      = 000001        OLDPC1  002672          START1  010266          TST15   015030
EM33    046425          GOAGIN  025002          OLDPC2  002676          STKIFL  034020          TST16   015262
EM34    046456          GTSWR = 104407          OLDPS1  002674          STKLMT= 177774          TST17   015520
EM35    046516          HAKTPM  030326          OLDPS2  002700          SVADRS  036150          TST2    011442
EM36    046577          HEADER  036340          PASCNT  002556          SWR     001340          TST20   016116
EM37    046627          HT      = 000011        PASNUM  037423          SWREG   000176          TST21   016522
EM4     045070          IE      = 000100        PATRNS  006250          SW0     = 000001        TST22   017036
EM40    046665          INBUF   002616          PIRQ    = 177772        SW00    = 000001        TST23   017334
EM41    046715          INBUF1  002656          PIRQVE= 000240          SW01    = 000002        TST24   020072
EM42    046763          INOOUT  021344          PNTPRI  005630          SW02    = 000004        TST25   020374
EM43    047017          INTA    003546          PROMPT  032144          SW03    = 000010        TST26   020664
EM44    047050          IOTVEC= 000020          PR0     = 000000        SW04    = 000020        TST27   021142
EM45    047120          KDPAR2= 172364          PR1     = 000040        SW05    = 000040        TST3    011572
EM46    047156          KDPDR2= 172324          PR2     = 000100        SW06    = 000100        TST30   021360
EM47    047233          KIPAR2= 172344          PR3     = 000140        SW07    = 000200        TST31   021646
EM5     045121          KIPDR2= 172304          PR4     = 000200        SW08    = 000400        TST32   021776
EM50    047250          LENCHK  002624          PR5     = 000240        SW09    = 001000        TST33   022356
EM51    047315          LETNCR  035071          PR6     = 000300        SW1     = 000002        TST34   022662
EM52    047374          LEVEL   002540          PR7     = 000340        SW10    = 002000        TST35   023244
EM53    047435          LEVELS  005264          PS      = 177776        SW11    = 004000        TST36   023542
EM54    047463          LEVEL3= 000140          PSTATE  005606          SW12    = 010000        TST37   024056
EM55    047500          LEVEL4= 000200          PSW     = 177776        SW13    = 020000        TST4    012124
EM56    047520          LEVEL5= 000240          PTCAPT  003170          SW14    = 040000        TST40 = 024360
EM57    047556          LEVEL6= 000300          PWRVEC= 000024          SW15    = 100000        TST5    012730
EM6     045152          LEVEL7= 000340          QTYBRD  002414          SW2     = 000004        TST6    013124
EM60    047606          LF      = 000012        RA      = 002416        SW3     = 000010        TST7    013472
EM61    047645          LODBUF  003472          RBAR    002566          SW4     = 000020        TWO     035316
EM62    047730          LOOP    002662          RBDR    002564          SW5     = 000040        TYPCNF  004630
EM63    047777          LRGSTC  002704          RCSR    002560          SW6     = 000100        TYPDE = 104406
EM65    050037          MA      = 010000        RDCHR = 104411          SW7     = 000200        TYPDS = 104405
EM66    050110          MAICLR  006266          RDDEC = 104414          SW8     = 000400        TYPE  = 104401
EM7     045203          MAISET  007266          RDLIN = 104412          SW9     = 001000        TYPOC = 104402
ENDEV   024360          MANSIZ  002670          RDOCT = 104413          TABINX  002546          TYPON = 104404
ENDSTK  044736          MBD     032130          RDYCHK  002632          TBITVE= 000014          TYPOS = 104403
ENPR1   002604          MBDIAL  036531          READ    002722          TESLR   030710          UBHANG  053622
EOPLOC  002706          MEMGMT  002712          REGADR  002416          TIME    002660          UCAL    035407
EOPRSM  030636          MEPITM  031524          REINIT  011206          TKB     002636          VA      = 002456
ER      = 100000        MESSAG  002650          REIR    002562          TKS     002634          VCLCHR  034703
ERCAPT  003126          MMPS    = 000252        RESVEC= 000010          TKVEC = 000060          VCTADS  005502
ERCHDR  034073          MMR0    = 177572        RSTRT   025022          TMOPSW= 000006          VECADR  002456
ERRCHK  004254          MMVECT= 000250          RWCR    002570          TORNCB  035747          VECERR  034517
ERRCNT  002714          MSG     002644          RY      = 000200        TOVECT= 000004          W       035305
ERROR = 104000          MUSTED  034770          R6      =%000006        TPB     002642          WCLEN   002630
ERRVEC= 000004          M1      037170          R7      =%000007        TPS     002640          WCR     002516
ER200   002274          M1A     037307          SCOPE = 000004          TPVEC = 000064          XCHKBU  041452
ESC     = 000033        N       035320          SDADRS  036204          TRAPVE= 000034          XINBUF  040450
ESCAPE  034762          NO      035307          SDRINV  002532          TRTVEC= 000014          X6      = 000020
ETDEV   034250          NOBUT   035433          SDRVS   002534          TSNUMB  037402          X7      = 000040
EWCR    002602          NOCARE  053700          SETTUP  005660          TSTCOM  036237          YES     035312
EXPAND  026154          NODATA  034211          SPACEC  035276          TSTDEV  011020          $APTHD  001000
FIXTBL  003360          NODVPR  037037          SPACES  035250          TSTMM   006134          $ATYC   031552
FLAG    002652          NOMORE  036663          SPACE3  035253          TST1    011244          $ATY1   031526
FNC     = 000016        NPR1    002612          SPACE6  035257          TST10   013650          $ATY3   031534
FNCCNT  002654          NX      = 040000        SPACE7  035266          TST11   014104          $ATY4   031544
```

SYMBOL TABLE

```
$AUTOB  001334        $DEVCT  001420        $ICNT   001304        $OVER   030312        $TMP6   001374
$BASE   001464        $DEVM   001466        $INTAG  001335        $PASS   001416        $TN   = 000040
$BDADR  001322        $DOAGN  024760        $ITEMB  001314        $PASS2  002716        $TPB    001352
$BDDAT  001326        $DTBL   026132        $LF     001406        $PASTM  001006        $TPFLG  001357
$BELL   001400        $ENDAD  024750        $LFLG   031771        $POWER  032062        $TPS    001350
$CDW1   001470        $ENDCT  024566        $LPADR  001306        $PWRDN  031774        $TRAP   027550
$CDW2   001472        $ENDMG  024767        $LPERR  001310        $PWRUP  032012        $TRAP2  027572
$CHARC  025374        $ENULL  024764        $MADR1  001442        $QUES   001404        $TRP  = 000015
$CKSWR  026422        $ENV    001430        $MADR2  001446        $RDCHR  026740        $TRPAD  027604
$CMTAG  001300        $ENVM   001431        $MADR3  001452        $RDDEC  027232        $TSTM   001004
$CM3  = 000000        $EOP    024532        $MADR4  001456        $RDLIN  027060        $TSTNM  001302
$CM4  = 000007        $EOPCT  024560        $MAIL   001410        $RDOCT  027410        $TTYIN  027166
$CNTLG  027202        $ERFLG  001303        $MAMS1  001440        $RDSZ = 000007        $TYPD   025642
$CNTLU  027175        $ERMAX  001315        $MAMS2  001444        $RTNAD  024762        $TYPDE  025626
$CPUOP  001436        $ERROR  031050        $MAMS3  001450        $SCOPE  027636        $TYPDS  025636
$CRLF   001405        $ERRPC  001316        $MAMS4  001454        $SETUP= 000137        $TYPE   025032
$DBLK   026142        $ERRTB  001534        $MBADR  001002        $STUP = 177777        $TYPEC  025244
$DDW0   001474        $ERRTY  031250        $MFLG   031770        $SVLAD  030256        $TYPEX  025376
$DDW1   001476        $ERTTL  001312        $MNEW   027220        $SVPC = 001000        $TYPOC  025424
$DDW10  001520        $ESCAP  001376        $MSGAD  001424        $SWR  = 163400        $TYPON  025440
$DDW11  001522        $ETABL  001430        $MSGLG  001426        $SWREG  001432        $TYPOS  025400
$DDW12  001524        $ETEND  001534        $MSGTY  001410        $SWRMK= 000200        $UNIT   001422
$DDW13  001526        $FATAL  001412        $MSWR   027207        $SW08T  030752        $UNITM  001010
$DDW14  001530        $FFLG   031772        $MTYP1  001441        $TESTN  001414        $USWR   001434
$DDW15  001532        $FILLC  001356        $MTYP2  001445        $TKB    001346        $VECT1  001460
$DDW2   001500        $FILLS  001355        $MTYP3  001451        $TKS    001344        $VECT2  001462
$DDW3   001502        $GDADR  001320        $MTYP4  001455        $TMP0   001360        $XTSTR  030120
$DDW4   001504        $GDDAT  001324        $NULL   001354        $TMP1   001362        $$GET4= 000000
$DDW5   001506        $GET42  024740        $NUMS   026410        $TMP2   001364        $$SW08= 000040
$DDW6   001510        $GTSWR  026522        $NWTST= 000001        $TMP3   001366        $OFILL  025623
$DDW7   001512        $HD   = 000000        $OCNT   025622        $TMP4   001370        .SAV    037444
$DDW8   001514        $HIBTS  001000        $OMODE  025624        $TMP5   001372        .$X   = 001000
$DDW9   001516        $HIOCT  027546

. ABS.  053702      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  56272 WORDS  ( 220 PAGES)
DYNAMIC MEMORY:  20346 WORDS  ( 78 PAGES)
ELAPSED TIME:  00:12:58
CZDRLC.BIN,CZDRLC.SEQ/-SP/NL:TOC=CZDRLC.MLB/ML,CZDRLC.P11
```