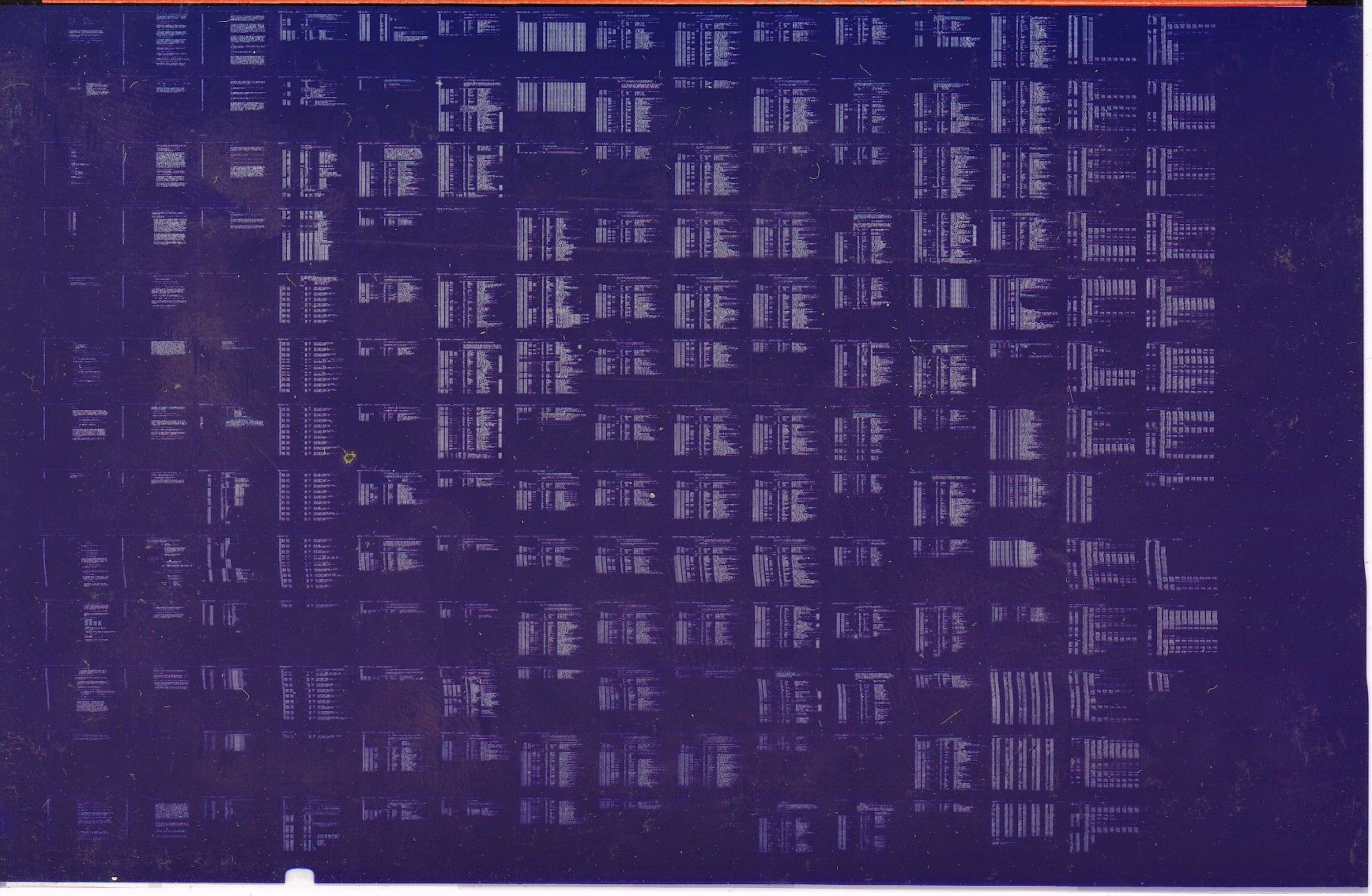
DR11-W DR11 GEN NPR INTFC AH-E780D-MC
CZDRLDO FICHE 1 OF 1



```
.REM    @

        IDENTIFICATION


PRODUCT CODE:           AC-E779D-MC

PRODUCT NAME:           C2DRLDO DR11 GEN NPR INTFC

DATE RELEASED:          OCTOBER, 1981

MAINTAINER:             DIAGNOSTIC ENGINEERING

AUTHOR:                 DAN P. MILLEVILLE
```

```
                                        HISTORY

                                REV     DATE            NOTE

                                A       1977            INITIAL RELEASE
                                B       1980            CORRECTION OF CODING ERRORS
                                C       1980            11/05 PROBLEM AND ENDLESS LOOP PROBLEM FIXED
                                D       1981          > 11/10 PROBLEM - FAILS TEST 4 MISTAKENLY
                                                      > CORRECTION OF CODING PROBLEMS WHEN RUN
                                (ALL REV 'D' FIXES ARE  ON 11/34'S.
                                HIGHLIGHTED BY ';DPM001'
                                IN THE COMMENT FIELD)
                                                      > ADDITION OF CODE TO CHECK THE POWER
                                                        MONITOR BIT OF THE CPU ERROR REGISTER,
                                                        IF ONE EXISTS.
                                                      > CHANGED AUTOSIZE ROUTINE TO PROMPT USER
                                                        FOR DEVICE ADDRESS IF NON-OPTION MANU-
                                                        FACTURING ENVIRONMENT (IF OPTION MANF.,
                                                        SIZING OCCURS AS BEFORE), AND (IF MODULE
                                                        FAILS TO INTERRUPT) ITS VECTOR AND
                                                        PRIORITY.
```

TABLE OF CONTENTS

```
                                1.0     ABSTRACT

                                        THIS DIAGNOSTIC PROGRAM IS CAPABLE OF TESTING THE DR11-W
                                        NPR GENERAL INTERFACE IN DR11-W OR DR11-B MODE.

                                        IT HAS THE FOLLOWING FEATURES:

                                        1.  APT11/XXDP COMPATIBLE
                                        2.  MULTIPLE BOARD TESTING USING TABLE CREATED BY USER
                                        3.  BURST DATA LATE CALIBRATION
                                        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. 56000 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)

                                                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).
```

```
223
224                          4.0     LOAD AND START PROCEDURE
225
226                                  1.  LOAD PROGRAM INTO MEMORY.
227                                  2.  LOAD STARTING ADDRESS 200, 204 OR 210.  (SEE SECTS. 7.1, 7.2, 7.3
228                                      RESPECTIVELY)
229                                  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          100% AUTOSIZE MODE

                                THIS IS TO BE USED BY OPTION MANUFACTURING ONLY.
                                BECAUSE OF THE LARGE ADDRESSING WINDOW, OTHER
                                OPTIONS HAVE BEEN FOUND THAT GIVE THE AUTOSIZER
                                THE IMPRESSION A DR11 IS WHERE IT IS NOT.  THIS
                                BIT SET WILL BYPASS THE ROUTINE PROMPTING THE
                                USER FOR THE DR11 ADDRESSES.

                                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 CON-
                                JUNCTION 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 OF ER '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 N CHANGE ANY VALUES PRESENT OR CHANGED. TO RETURN TO E 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 PROMPT THE USER FOR THE STARTING ADDRESSES OF ALL BOARD(S) TO TEST. IF THIS IS NOT DESIRED, RAISE BIT 9 (1000) IN THE SWR *ONLY*IF*THERE*ARE* *NO*OTHER*NON-DR11*OPTIONS*THAT*COULD*BE*MISTAKENLY*AUTO-* *SIZED*IN*THE*160010*TO*172770*ADDRESS*RANGE*. THE DIAGNOSTIC *WILL* FAIL IF IT FINDS A NON-DR11 OPTION AND STARTS TO TEST IT AS SUCH. IF THE BOARD WHOS ADDRESS YOU JUST INPUTED FAILS TO INTERRUPT, THE AUTOSIZE ROUTINE WILL PROMPT YOU FOR THE VECTOR AND PRIORITY. ALL OTHER INFORMATION ON THE BOARD IS AUTO-SIZED. THE ONLY LEGAL INTERRUPT VECTORS THE DR11 CAN BE SET UP FOR ARE 300-774, 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 160010, 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 (ONLY IF THE BOARD INTERRUPTS PROPERLY), W/B, 2/N CYCLE, AND CABLE STATES OF EACH BOARD, INDEPENDENT OF THE STATES OF OTHER BOARDS.

```
585                                   B.  THE FOLLOWING WILL NOT BE OFFERED TO THE USER IN AUTOSIZE
586                                       MODE:
587
588                                       1.  BURST DATA LATE CALIBRATION
589                                       2.  MULTIPLE BOARD DIALOGUE
590
591                                   THE DIAGNOSTIC WILL PRINT ONE OF THE FOLLOWING:
592
593                                     I) DO YOU WISH TO RELOAD THE TABLE (Y OR <CR>) ?
594
595                                                          OR
596
597                                     II) INPUT DEVICE    X STARTING ADDRESS (<CR> IF NO MORE) ?
598
599                                   IF THE DIAGNOSTIC HAS BEEN RUN AND THE TABLE HAS BEEN LOADED,
600                                   IT WILL PRINT PROMPT I).  IF YOU ANSWER 'Y', IT WILL PRINT
601                                   PROMPT II), ALLOWING INPUT OF ALL DR11 DEVICE ADDRESSES.  IF
602                                   THE DIAGNOSTIC HAS NOT BEEN RUN, IT WILL BYPASS PRINTING I)
603                                   AND PROMPT II) WILL APPEAR.  AFTER INPUTING A <CR> TO THE II)
604                                   PROMPT, THE FOLLOWING WILL PRINT:
605
606                                     DIAGNOSTIC HAS DETERMINED THE FOLLOWING ABOUT THE
607                                     DR11-W(S) IT HAS FOUND.  USER *MUST* DETERMINE ACCURACY
608
609                                        BOARD#  REGADR   VECADR   W/B    P-LEV   2-N CY  CABLE
610                                          X     XXXXXX     XXX     X       X       X       X
611
612                                   DATA WILL CONTINUE TO PRINT UNTIL DATA FOR ALL MODULES HAS BEEN
613                                   PRINTED.  IF YOU INPUTED OTHER THAN A 'Y' TO THE I) PROMPT, THE
614                                   ABOVE MESSAGE WILL NOT PRINT.  THE FOLLOWING WILL PRINT UNCON-
615                                   DITIONALLY:
616
617                                     (^X) INHIBITS EOP'S, (^Y) FOR ERROR SUMMARY
618                                     UNIBUS HANG?  RESTART AT ADDRESS XXXXXX
619
620                                     CZDRLDO DR11 GEN NPR INTFC LOGIC TEST
```

```
                THE CONTROL X (^X) FEATURE BYPASSES THE SECTIONS THAT PRINT THE
                END-OF-PASS AND END-OF-DEVICE MESSAGES.  THIS IS TO IMPROVE THE
                NUMBER OF PASSES EXECUTED OVER ANY PERIOD OF TIME, AS WELL AS
                MAKE OVERNIGHT OR WEEKEND RUNS USE NO PAPER (IF NO ERRORS).
                ERROR TYPEOUTS ARE NOT DISABLED IN THIS MODE.  WHEN AN ERROR
                OCCURS, THE END-OF-PASS (EOP) WILL PRINT FOR THAT PASS, AND, IF
                MORE THAN ONE MODULE IS BEING TESTED, AN END-OF-DEVICE (EOD) AS
                WELL AS END-OF-PASS WILL PRINT SO YOU WILL KNOW WHICH DEVICE
                AND PASS WAS EXECUTING WHEN THE ERROR OCCURED.  IN ORDER TO
                OBTAIN A PROGRESS REPORT, HIT ANY KEY REPEATEDLY, SINCE INITS
                AND RESETS DONE DURING THE EXECUTION OF THE DIAGNOSTIC MAY
                CLEAR THE CHARACTER WAITING FLAG BEFORE THE CHECK FOR THIS BIT.
                WHEN THE CHARACTER IS RECOGNIZED, AN EOP, AND IF MORE THAN ONE
                MODULE, AN EOD MESSAGE WILL PRINT GIVING THE USER A PROGRESS
                REPORT.  TO DISABLE THIS FEATURE, REPEATEDLY ENTER (^X) AGAIN
                UNTIL THE CPU RECOGNIZES YOUR INPUT.
```

```
637                                     THE CONTROL Y (^Y) FUNCTION CALLS FOR A SUMMARY OF DEVICE(S)
638                                     AND PASS(ES) THAT HAD ERRORS.  IF NO ERRORS OCCURED SINCE THE
639                                     BEGINNING OF THE DIAGNOSTIC, OR SINCE THE LAST ERROR REPORT,
640                                     THE FOLLOWING IS PRINTED:
641
642                                       NO ERROR TOTALS TO REPORT
643
644                                     IF THERE WERE ERRORS, THE FOLLOWING IS PRINTED:
645
646                                       SUMMATION OF ERRORS SINCE BEGINNING OR LAST REPORT
647                                         BOARD #  PASS #  ERRTTL
648                                           X        X        X
649                                           X        X        X
650                                           X        X        X (ETC.)
651
652                                     THE INFORMATION IS STORED ON A STACK THAT WILL HOLD UP TO 150
653                                     (DECIMAL) DEVICE-PASS ERROR DATA LINES ABOVE.  IF THE LIMIT IS
654                                     REACHED, DIAGNOSTIC WILL CONTINUE, BUT FURTHER DATA WILL NOT BE
655                                     STORED.  THE DATA ACCUMULATED IS NOT WRITTEN OVER, BUT WHEN (^Y)
656                                     IS ENTERED, THE FOLLOWING IS PRINTED JUST BEFORE THE 'SUMMATION...'
657                                     STATEMENT ABOVE:
658
659                                       STACK IS FULL - DATA MAY HAVE BEEN LOST
660
661                                     WHEN THE DATA IS PRINTED, THE STACK IS REINITIALIZED AND WILL
662                                     START STORING UP TO ANOTHER 150 ERROR DATA LINES.
663
664                                     IN THE EVENT THE UNIBUS BECOMES HUNG, AND YOU HAVE NON-VOLATILE
665                                     MEMORY OR BATTERY BACKUP, RESTART THE PROGRAM AT THE ADDRESS
666                                     SPECIFIED BY THE 'UNIBUS HUNG...' PROMPT AT THE START OF THE
667                                     DIAGNOSTIC.  THE PRINTOUT WILL BE AS FOLLOWS:
668
669                                       DEVICE ADDRESS - XXXXXX, TEST NUMBER - XXXXXX, PASS NUMBER - XXXXXX
670
671                                     CPU WILL HALT.  HITTING CONTINUE WILL CAUSE THE PROGRAM TO
672                                     RESTART AS THOUGH YOU HAD STARTED AT 200.
```

```
673                                    7.3     RESTARTING PROGRAM IN MEMORY (STARTING ADDRESS = 210)
674
675                                            WHENEVER THE PROGRAM IS HALTED, ALL HISTORY OF PREVIOUS TESTING
676                                            IS SAVED.  IT WILL REMAIN INTACT UNTIL:
677
678                                              1.  ANOTHER PROGRAM IS LOADED INTO MEMORY
679                                              2.  THE USER RE-EDITS THE TABLE
680
681                                            TO RESTART THE PROGRAM, ENTER SA 210 AND START.  THIS START
682                                            PRECLUDES ANY SETUP AND NEGATES THE START MESSAGE OBTAINED
683                                            WHEN STARTING AT 200.  DO NOT START AT THIS LOCATION IF THE
684                                            DIAGNOSTIC HAS NOT BEEN PREVIOUSLY "STARTED" AT EITHER 200
685                                            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 (*NOTE*: STARTING
                                                        WITH REV 'D' AND ABOVE, RUNNING DIAG
                                                        IN THIS MODE REQUIRES USER TO INPUT
                                                        THE STARTING ADDRESSES, AS IS WITH
                                                        STANDALONE MODE)

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

                            VECTOR ADDRESS ($VECT1):  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

                            UNTIL THE CPU                                   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 ONCE EVERY 20 SECONDS, APPROXIMATELY (17
                                        SECONDS ON AN 11/44), AS FOLLOWS WITH NO ERRORS ON THAT PASS:

                                        END PASS # XXXXXX

                                        THE EOP WILL PRINT AFTER 64 DEVICES HAVE BEEN TESTED.  IF THERE
                                        IS ONLY 1 MODULE, THE EOP WILL PRINT AFTER 64 PASSES.  WITH
                                        16 MODULES, THE EOP WILL PRINT AFTER ONLY 4 PASSES.
                                        THE EOP WILL PRINT AS FOLLOWS WITH SOME ERRORS WHEN TESTING 1
                                        DEVICE:

                                        END PASS # XXXXXX   TOTAL ERRORS THIS PASS ALL MODULE(S) 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.
```

```
782                          9.0     EXECUTION TIME
783
784                                  ON A PDP11/44:
785
786                                  IN ALL MODES: APPROXIMATELY 64 DEVICE PASSES IN 17 SECONDS, ALL MODES.
```

```
787                             10.0    SUBROUTINE ABSTRACT
788
789
790                                     10.1    READ
791
792                                             THE READ SUBROUTINE IS USED IN THE EDIT ROUTINE TO INPUT UP TO
793                                             6 DIGITS IN OCTAL, 2 DIGITS IN DECIMAL, OR A SINGLE NON-
794                                             NUMERIC CHARACTER.  R4 IS USED AS THE LOCATION TO HOLD THE
795                                             NUMBER IN OCTAL, AND IS CLEARED FOR THAT PURPOSE AT THE START
796                                             OF THE SUBROUTINE.  R3 IS TO BE PRELOADED WITH THE NUMBER OF
797                                             DIGITS EXPECTED, SINCE A <CRLF> IS PRINTED WHEN THE LIMIT IS
798                                             REACHED.  ENTERING A <CRLF> BEFORE THE LIMIT IS ACCEPTABLE, AS
799                                             IT WILL BE INTERPRETED AS A NON-NUMERIC CHARACTER AND EXIT.
800                                             IN ANY CASE, THE LAST INPUTED ASCII CHARACTER IS LEFT IN LOCA-
801                                             TION 'ANSWER'.  IF A NUMERIC CHARACTER IS INPUTED, IT WILL
802                                             CLEAR ALL BUT THE 1ST 4 BITS IN LOCATION 'ANSWER', EXPOSING THE
803                                             VALUE OF THE DIGIT INPUTED, ROTATE R4 TO THE LEFT 3 PLACES TO
804                                             MAKE ROOM FOR THE INPUTED DIGIT, AND ADD IT TO R4.  LOCATION
805                                             'LRGSTC' IS TO BE LOADED WITH THE LARGEST ASCII NUMBER DIGIT
806                                             ACCEPTABLE FOR THIS NUMBER, I.E. 7 OR 9 (FOR OCTAL OR DECIMAL
807                                             INPUT RESPECTIVELY).  ANY CHARACTER OUTSIDE ASCII '0' OR '7/9'
808                                             IS TREATED AS A NON-NUMERIC, TRIGGERING AN AUTOMATIC <CRLF>
809                                             AND EXIT.
810
811                                     10.2    ERCAPT
812
813                                             THIS SUBROUTINE SAVES THE UNIT NUMBER, PASS NUMBER AND TOTAL
814                                             ERRORS FOR THAT DEVICE/PASS WHENEVER IT ENCOUNTERED ERRORS.
815                                             THIS ROUTINE SAVES DATA FOR 150 (DECIMAL) PASSES.  IF THE STACK
816                                             SHOULD BECOME FULL, DATA STARTING WITH THE 151ST PASS CONTAIN-
817                                             ING ERRORS IS LOST.
818
819                                     10.3    FIXTBL
820
821                                             THIS SUBROUTINE FILLS THE 17 OCTAL LOCATIONS STARTING AT
822                                             'REGADR' AND 'VECADR' FROM THE STARTING VALUES ALREADY LOADED
823                                             IN THE FIRST LOCATIONS IN STEPS OF 10 FOR EACH TABLE.
824
825                                     10.4    LODBUF
826
827                                             THE INBUF BUFFER IS LOADED WITH AN INCREMENTING PATTERN
828                                             (0,1,2,3,...) BEGINNING AT THE STARTING ADDRESS OF INBUF.  THE
829                                             NUMBER OF WORDS LOADED IS DETERMINED BY THE CONTENTS OF BUFLEN.
```

10.5 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.

10.6 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.

10.7 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 RETUPN AFTER THE ERROR CALL AND JSR RETURN.

10.8 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.

10.9 CHKCAB

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

10.10 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.

```
886                                     10.11   ERRCHK
887
888                                             THIS ROUTINE CLEARS IE AND UPDATES THE PC FOR RETURN AFTER THE
889                                             ERROR IN THE TEST IF ERROR IS CLEAR.  IF SET, RETURN IS EXECU-
890                                             TED WITHOUT UPDATING THE PC RETURN SO THE ERROR CALL AFTER THE
891                                             JSR CALL IN THE TEST WILL BE CALLED.
892
893                                     10.12   BPINIT
894
895                                             THIS SUBROUTINE RELOADS THE ".+2" AND 'BPT" INTO THE UNUSED
896                                             LOCATIONS BETWEEN 4 AND 776.
897
898                                     10.13   DRGET
899
900                                             THIS SUBROUTINE EXTRACTS INFORMATION ABOUT THE DR11 WHOS ADDRESS
901                                             WAS INPUTED AND LOADS THE ACCUMULATED DATA INTO THE DEVICE
902                                             DESCRIPTOR WORD FOR THAT BOARD.
903
904                                     10.14   TYPCNF
905
906                                             THIS SUBROUTINE PRINTS THE BOARD CONFIGURATIONS THAT THE ASIZE
907                                             SUBROUTINE SIZED FOR ON THE UNIBUS.
908
909                                     10.15   ASIZE
910
911                                             THIS ROUTINE SEMI-AUTOSIZES THE BOARD CONFIGURATION (DUE TO THE
912                                             WIDE ADDRESS WINDOW, REV 'D' WAS MODIFIED TO PROMPT USER FOR
913                                             STARTING ADDRESS(ES) OF THE MODULE(S) UNDER TEST.  AUTOSIZE
914                                             ROUTINE DOES GET CABLE STATUS, INTERRUPT PRIORITY, ETC. ONCE
915                                             ADDRESS IS INPUTED.  IF BOARD FAILS TO INTERRUPT, DIAGNOSTIC
916                                             THEN PROMPTS USER FOR THE VECTOR AND PRIORITY.  EVERYTHING ELSE
917                                             IS AUTO-SIZED) AND PRINTS THE CONFIGURATION IF THE TABLE WAS
918                                             NEWLY CREATED THIS RUN.  IF THE TABLE ALREADY EXISTS, THE
919                                             ROUTINE ASKS THE USER IF A NEW TABLE IS TO BE CREATED.  IF NOT,
920                                             THE ROUTINE IS BYPASSED.
```

10.16 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.

10.17 PSTATE

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

10.18 PNTPRI

THIS ROUTINE PRINTS THE DEVICE PRIORITY IN THE DDW LOCATION.

10.19 SETTUP

THIS SUBROUTINE INITIALIZES THE TRAP AND INTERRUPT VECTORS.

10.20 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.
```

```
977                             @
978                             .NLIST  MC,MD,CND
```

```
3032                                    .TITLE  CZDRLDO-DR11 GEN NPR INTFC
                                        ;*COPYRIGHT (C) 1981
                                        ;*DIGITAL EQUIPMENT CORP.
                                        ;*MAYNARD, MASS. 01754
                                        ;*
                                        ;*PROGRAM BY DAN MILLEVILLE
                                        ;*
                                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                        ;*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.
                                        ;*
```

OPERATIONAL SWITCH SETTINGS

```
3034                            .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        14            LOOP ON TEST
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        12            100% AUTOSIZE
                                ;*        10            BELL ON ERROR
                                ;*         9            LOOP ON ERROR
                                ;*         8            LOOP ON TEST IN SWR<6:0>
3035                            ;*         7            INHIBIT MULTIPLE ERROR TYPEOUTS
3036
3037                            ;;*****************************************************************
3038                            ;*                      >>>>>>>>>>>>>>NOTE<<<<<<<<<<<<<<
3039                            ;*      THE ''TEST NUMBER TRACE ENABLING'' ABOVE IS TO ENABLE PRINTING OF THE
3040                            ;*      TEST NUMBER AT THE BEGINNING OF EACH TEST, SHOULD YOU HAVE A PROBLEM
3041                            ;*      WITH THE DIAGNOSTIC JUMPING TO AN ILLEGAL LOCATION FROM ANYWHERE IN
3042                            ;*      THE TEST.  BY ENABLING IT, YOU CAN SEE THE LAST TEST ENTERED BEFORE
3043                            ;*      THE OUT-OF-SEQUENCE JUMPING OCCURED.  THIS IS USEFUL IF YOU ENCOUNTER
3044                            ;*      AN UNEXPECTED TRAP FROM A PARTICULAR TEST, BUT DISCOVER THAT THE TEST
3045                            ;*      RAN SUCCESSFULLY WHEN LOOPED.
3046                            ;*
3047                            ;;*****************************************************************
```

```
                                .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
3050            000004                  BUSERR  =ERRVEC
3051            172410                  ABASE   =172410         ;BASE DEVICE ADDRESS
3052            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                          ER      =100000 ;ERROR
```

CSR BIT COMPLIMENT DEFINITIONS

```
                                        .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
```

```
3102                                    .SBTTL  COMPLEMENTS OF BIT DEFINITIONS
3103                                ;;**********************************************************************
3104            177776              CBIT0   =177776 ;COMPLIMENT OF BIT0
3105            177775              CBIT1   =177775 ;COMPLIMENT OF BIT1
3106            177773              CBIT2   =177773 ;COMPLIMENT OF BIT2
3107            177767              CBIT3   =177767 ;COMPLIMENT OF BIT3
3108            177757              CBIT4   =177757 ;COMPLIMENT OF BIT4
3109            177737              CBIT5   =177737 ;COMPLIMENT OF BIT5
3110            177677              CBIT6   =177677 ;COMPLIMENT OF BIT6
3111            177577              CBIT7   =177577 ;COMPLIMENT OF BIT7
3112            177377              CBIT8   =177377 ;COMPLIMENT OF BIT8
3113            176777              CBIT9   =176777 ;COMPLIMENT OF BIT9
3114            175777              CBIT10  =175777 ;COMPLIMENT OF BIT10
3115            173777              CBIT11  =173777 ;COMPLIMENT OF BIT11
3116            167777              CBIT12  =167777 ;COMPLIMENT OF BIT12
3117            157777              CBIT13  =157777 ;COMPLIMENT OF BIT13
3118            137777              CBIT14  =137777 ;COMPLIMENT OF BIT14
3119            077777              CBIT15  =77777  ;COMPLIMENT OF BIT15
3120            057777              CB1513  =57777  ;COMPLIMENT OF BIT15 & BIT 13
```

```
3121                                    .SBTTL  PRIORITY LEVELS AND OTHER DEFINITIONS
3122                                    ;;*****************************************************************
3123            000140                  LEVEL3  =140
3124            000200                  LEVEL4  =200
3125            000240                  LEVEL5  =240
3126            000300                  LEVEL6  =300
3127            000340                  LEVEL7  =340
3128            000033                  ESC     =33
3129            000003                  CNTLC   =3
3130            000015                  CARETN  =15
3131            177572                  MMR0    =177572
3132            172300                  KIPDR0  =172300
3133            172304                  KIPDR2  =172304
3134            172324                  KDPDR2  =172324
3135            172340                  KIPAR0  =172340
3136            172344                  KIPAR2  =172344
3137            172364                  KDPAR2  =172364
3138            000250                  MMVECT  =250
3139            000252                  MMPS    =252
3140            000004                  TOVECT  =4
3141            000006                  TMOPSW  =6
3142            000003                  BPT     =3
3143            025622                  TST40=ENDEV     ;BRANCH TO TEST 40 - BRANCH TO ENDEV (THERE IS NO TEST 40)
```

PRIORITY LEVELS AND OTHER DEFINITIONS

```
3144                                    ;;*****************************************************************
3145                                    ;* ALL UNUSED LOCATIONS FROM 4-776 WILL CONTAIN A ".+2,BPT" SEQUENCE
3146                                    ;* TO CATCH ILLEGAL TRAPS & INTERRUPTS TO THE 'CATCH' LOCATION
3147                                    ;* 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
3148                                    ;;*****************************************************************
3149          000014                    .=14                    ;THE BPT TRAP VECTOR POINTS TO THE
3150 000014  006466                    BPTVCT: .WORD   CATCH     ;   ILLEGAL TRAP HANDLER "CATCH"
3151 000016  000340                            .WORD   LEVEL7
3153          000042                    .=42
3154 000042  000000                    .WORD   0               ;CLEAR THIS LOCATION (FOR APT MONITOR STARTING ADDRESS)
3155          000174                    .=174
3156 000174  000000                    DISPRE: .WORD   0
3157 000176  000000                    SWREG:  .WORD   0
3158                                    ;;*****************************************************************
3159                                    ;
3160                                    ;PROGRAM STARTING LOCATIONS
3161                                    ;
3162                                    ;;*****************************************************************
3163 000200  000137  011420            JMP     START1          ;NORMAL START
3164 000204  000137  034230            JMP     MBD             ;ENTER MULTIPLE BOARD DIALOQUE
3165 000210  005037  001416    STAGIN: CLR     $PASS           ;CLEAR $PASS
3166 000214  005037  002716            CLR     $PASS2          ;CLEAR $PASS2
3167 000220  005037  001422            CLR     $UNIT           ;CLEAR $UNIT
3168 000224  005037  001420            CLR     $DEVCT          ;CLEAR $DEVCT
3169 000230  005037  001414            CLR     $TESTN          ;CLEAR $TESTN
3170 000234  005037  002706            CLR     EOPLOC          ;CLEAR EOPLOC
3171 000240  012737  044652  044650    MOV     #CAPSTK,CAPNTR  ;RESET THE CAPTURE POINTER
3172 000246  012706  011424            MOV     #START,SP       ;RESET THE STACK POINTER
3173 000252  000137  012356            JMP     BEGIN1          ;JUMP TO BEGIN1 FOR RESTART WITHOUT HEADER PRINTING
3174          001000                    .=1000
```

ACT11 HOOKS

```
3175                                    .SBTTL  ACT11 HOOKS
                                        ;;******************************************************************
                                        ;HOOKS REQUIRED BY ACT11
            001000                              $SVPC=.                 ;SAVE PC
            000046                              .=46
      000046 026324                             $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
            000052                              .=52
      000052 000000                             .WORD   0               ;;2)SET LOC.52 TO ZERO
            001000                              .=$SVPC                 ;; RESTORE PC
3176                                    .SBTT'  APT PARAMETER BLOCK
                                        ;;******************************************************************
                                        ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                        ;;******************************************************************
            001000                              .$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)
```

```
 3178                                   .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                     $ERRPC: .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  000300                  $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
```

ERROR POINTER TABLE

```
                                .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                     $ERRTB:
  3179
  3180 001534  047162                   ;ITEM 1
                                        .WORD   EM2     ;CANNOT ACCESS DR11 REGISTER
  3181 001536  052503                   .WORD   DH2     ;TEST #  ERR PC  ABRTPC  REGISTER
  3182 001540  055246                   .WORD   DT2     ;$TESTN,$ERRPC,OLDPC1,DREG,0
  3183 001542  000000                   .WORD   0       ;PRINT ALL DATA OCTAL
  3184
  3185                                  ;ITEM 2
  3186 001544  047216                   .WORD   EM3     ;DR11-B OR W MODE INCORRECT (0=B, 1=W)
  3187 001546  052544                   .WORD   DH3     ;TEST #  ERR PC  EXPMOD  ACTMOD  CSRADR
  3188 001550  055260                   .WORD   DT3     ;$TESTN,$ERRPC,$TMP1,BORW,CSR,0
  3189 001552  000000                   .WORD   0       ;PRINT ALL DATA OCTAL
  3190
  3191                                  ;ITEM 3
  3192 001554  047264                   .WORD   EM4     ;INIT FAILED TO CLEAR WCR
  3193 001556  052613                   .WORD   DH4     ;TEST #  ERR PC  WCRADR  WCRCONTENTS
  3194 001560  055274                   .WORD   DT4     ;$TESTN,$ERRPC,WCR,RWCR,0
  3195 001562  000000                   .WORD   0       ;PRINT ALL DATA OCTAL
  3196
  3197                                  ;ITEM 4
  3198 001564  047315                   .WORD   EM5     ;INIT FAILED TO CLEAR BAR
  3199 001566  052657                   .WORD   DH5     ;TEST #  ERR PC  BARADR  BAREXP  BARRCV
  3200 001570  055306                   .WORD   DT5     ;$TESTN,$ERRPC,BAR,EBAR,RBAR,0
  3201 001572  000000                   .WORD   0       ;PRINT ALL DATA OCTAL
  3202
  3203                                  ;ITEM 5
  3204 001574  047346                   .WORD   EM6     ;INIT FAILED TO CLEAR BDR
  3205 001576  052726                   .WORD   DH6     ;TEST #  ERR PC  BDRADR  BDRCONTENTS
  3206 001600  055322                   .WORD   DT6     ;$TESTN,$ERRPC,BDR,RBDR,0
  3207 001602  000000                   .WORD   0       ;PRINT ALL DATA OCTAL
  3208
  3209                                  ;ITEM 6
  3210 001604  047377                   .WORD   EM7     ;INIT FAILED TO CLEAR ALL CSR R-W BITS
  3211 001606  052772                   .WORD   DH7     ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
  3212 001610  055334                   .WORD   DT7     ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
  3213 001612  000000                   .WORD   0       ;PRINT ALL DATA OCTAL
  3214
  3215                                  ;ITEM 7
  3216 001614  047445                   .WORD   EM10    ;RESET FAILED TO CLEAR WCR
  3217 001616  052613                   .WORD   DH4     ;TEST #  ERR PC  WCRADR  WCRCONTENTS
  3218 001620  055274                   .WORD   DT4     ;$TESTN,$ERRPC,WCR,RWCR,0
  3219 001622  000000                   .WORD   0       ;PRINT ALL DATA OCTAL
```

```
3220                                    ;ITEM 10
3221 001624  047477                             .WORD   EM11    ;ATTEMPT TO SET ALL WCR BITS FAILED
3222 001626  052613                             .WORD   DH4     ;TEST #  ERR PC  WCRADR  WCRCONTENTS
3223 001630  055274                             .WORD   DT4     ;$TESTN,$ERRPC,WCR,RWCR,0
3224 001632  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3225
3226                                    ;ITEM 11
3227 001634  047542                             .WORD   EM12    ;RESET FAILED TO CLEAR BAR
3228 001636  052657                             .WORD   DH5     ;TEST #  ERR PC  BARADR  BAREXP  BARRCV
3229 001640  055306                             .WORD   DT5     ;$TESTN,$ERRPC,BAR,EBAR,RBAR,0
3230 001642  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3231
3232                                    ;ITEM 12
3233 001644  047574                             .WORD   EM13    ;ATTEMPT TO SET ALL BAR BITS TO 1 FAILED
3234 001646  052657                             .WORD   DH5     ;TEST #  ERR PC  BARADR  BAREXP  BARRCV
3235 001650  055306                             .WORD   DT5     ;$TESTN,$ERRPC,BAR,EBAR,RBAR,0
3236 001652  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3237
3238                                    ;ITEM 13
3239 001654  047644                             .WORD   EM14    ;CSR BIT TEST FAILED (FATAL - DIAGNOSTIC NOT CONTINUED)
3240 001656  053046                             .WORD   DH14    ;                 BIT(S)
3241                                                            ;TEST #  ERR PC  TESTED  CSRADR  CSREXP  CSRCONTENTS
3242 001660  055350                             .WORD   DT14    ;$TESTN,$ERRPC,BUT,CSR,ECSR,RCSR,0
3243 001662  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3244
3245                                    ;ITEM 14
3246 001664  047733                             .WORD   EM15    ;CSR BIT TEST FAILED
3247 001666  053046                             .WORD   DH14    ;                 BIT(S)
3248                                                            ;TEST #  ERR PC  TESTED  CSRADR  CSREXP  CSRCONTENTS
3249 001670  055350                             .WORD   DT14    ;$TESTN,$ERRPC,BUT,CSR,ECSR,RCSR,0
3250 001672  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3251
3252                                    ;ITEM 15
3253 001674  047757                             .WORD   EM16    ;EIR BIT TEST FAILED
3254 001676  053161                             .WORD   DH16    ;                   BIT(S)
3255                                                            ;TEST #  ERR PC  TESTED  EIRADR  EIREXP  EIRCONTENTS
3256 001700  055366                             .WORD   DT16    ;$TESTN,$ERRPC,BUT,CSR,EEIR,REIR,0
3257 001702  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3258
3259                                    ;ITEM 16
3260 001704  050003                             .WORD   EM17    ;READY AND MAINTENANCE ARE NOT THE ONLY BITS SET IN CSR
3261 001706  053274                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
3262 001710  055404                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
3263 001712  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3264
3265                                    ;ITEM 17
3266 001714  050072                             .WORD   EM20    ;ATTN AND ERROR FAILED TO SET PROPERLY
3267 001716  053274                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
3268 001720  055404                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
3269 001722  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3270
3271                                    ;ITEM 20
3272 001724  050140                             .WORD   EM21    ;ATTN AND ERROR FAILED TO CLEAR PROPERLY
3273 001726  053274                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
3274 001730  055404                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
3275 001732  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
```

ERROR POINTER TABLE

```
3276                                    ;ITEM 21
3277 001734  050210                             .WORD   EM22    ;ERROR BIT SHOULD HAVE BEEN CLEAR
3278 001736  053274                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
3279 001740  055404                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
3280 001742  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3281
3282                                    ;ITEM 22
3283 001744  050320                             .WORD   EM24    ;READY OF CSR WAS NOT SET
3284 001746  053274                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
3285 001750  055404                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
3286 001752  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3287
3288                                    ;ITEM 23
3289 001754  050351                             .WORD   EM25    ;BIT 0 OF THE BAR WAS SET
3290 001756  052657                             .WORD   DH5     ;TEST #  ERR PC  BARADR  BAREXP  BARRCV
3291 001760  055306                             .WORD   DT5     ;$TESTN,$ERRPC,BAR,EBAR,RBAR,0
3292 001762  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3293
3294                                    ;ITEM 24
3295 001764  050476                             .WORD   EM30    ;FUNCTION BIT(S) ARE NOT CLEAR
3296 001766  053274                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
3297 001770  055404                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
3298 001772  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3299
3300                                    ;ITEM 25
3301 001774  050534                             .WORD   EM31    ;DSTAT A, B OR C ARE NOT AS EXPECTED
3302 001776  053274                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
3303 002000  055404                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
3304 002002  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3305
3306                                    ;ITEM 26
3307 002004  050600                             .WORD   EM32    ;BDR IS NOT CLEAR
3308 002006  052726                             .WORD   DH6     ;TEST #  ERR PC  BDRADR  BDRCONTENTS
3309 002010  055322                             .WORD   DT6     ;$TESTN,$ERRPC,BDR,RBDR,0
3310 002012  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3311
3312                                    ;ITEM 27
3313 002014  050621                             .WORD   EM33    ;ALL BDR BITS ARE NOT SET
3314 002016  052726                             .WORD   DH6     ;TEST #  ERR PC  BDRADR  BDRCONTENTS
3315 002020  055322                             .WORD   DT6     ;$TESTN,$ERRPC,BDR,RBDR,0
3316 002022  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3317
3318                                    ;ITEM 30
3319 002024  050712                             .WORD   EM35    ;BDR SHOULD NOT HAVE BEEN LOADED WITH NEW PATTERN
3320 002026  053500                             .WORD   DH34    ;TEST #  ERR PC  BDRADR  BDREXP  BDRCONTENTS
3321 002030  055450                             .WORD   DT34    ;$TESTN,$ERRPC,BDR,EBDR,RBDR,0
3322 002032  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3323
3324                                    ;ITEM 31
3325 002034  050773                             .WORD   EM36    ;BDR PATTERN NOT CORRECT
3326 002036  053500                             .WORD   DH34    ;TEST #  ERR PC  BDRADR  BDREXP  BDRCONTENTS
3327 002040  055450                             .WORD   DT34    ;$TESTN,$ERRPC,BDR,EBDR,RBDR,0
3328 002042  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
```

ERROR POINTER TABLE

```
3329                                    ;ITEM 32
3330 002044  051023                             .WORD   EM37    ;READY IS NOT THE ONLY BIT SET
3331 002046  053274                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
3332 002050  055404                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
3333 002052  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3334
3335                                    ;ITEM 33
3336 002054  051061                             .WORD   EM40    ;READY SHOULD NOT BE SET
3337 002056  053274                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
3338 002060  055404                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
3339 002062  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3340
3341                                    ;ITEM 34
3342 002064  051111                             .WORD   EM41    ;READY WAS CLEARED BUT NEVER SET AGAIN
3343 002066  053274                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
3344 002070  055404                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
3345 002072  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3346
3347                                    ;ITEM 35
3348 002074  051213                             .WORD   EM43    ;DR11 FAILED TO INTERRUPT
3349 002076  053554                             .WORD   DH43    ;TEST #  ERR PC  CSRADR  CSRCONTENTS
3350 002100  055464                             .WORD   DT43    ;$TESTN,$ERRPC,CSR,RCSR,0
3351 002102  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3352
3353                                    ;ITEM 36
3354 002104  051244                             .WORD   EM44    ;DR11 INTERRUPTED, BUT IT SHOULDN'T HAVE
3355 002106  053554                             .WORD   DH43    ;TEST #  ERR PC  CSRADR  CSRCONTENTS
3356 002110  055464                             .WORD   DT43    ;$TESTN,$ERRPC,CSR,RCSR,0
3357 002112  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3358
3359                                    ;ITEM 37
3360 002114  051314                             .WORD   EM45    ;ERROR BIT SHOULD NOT BE CLEAR
3361 002116  053274                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
3362 002120  055404                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
3363 002122  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3364
3365                                    ;ITEM 40
3366 002124  051427                             .WORD   EM47    ;CSR IS WRONG
3367 002126  053274                             .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
3368 002130  055404                             .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
3369 002132  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3370
3371                                    ;ITEM 41
3372 002134  051444                             .WORD   EM50    ;TRANSFERS SHOULD HAVE BEEN INHIBITED
3373 002136  053620                             .WORD   DH50    ;TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  BARADR  BAREXP  BARRCV
3374 002140  055476                             .WORD   DT50    ;$TESTN,$ERRPC,WCR,EWCR,RWCR,BAR,EBAR,RBAR,0
3375 002142  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3376
3377                                    ;ITEM 42
3378 002144  051511                             .WORD   EM51    ;DR11 SHOULD NOT HAVE INTERRUPTED A SECOND TIME
3379 002146  053554                             .WORD   DH43    ;TEST #  ERR PC  CSRADR  CSRCONTENTS
3380 002150  055464                             .WORD   DT43    ;$TESTN,$ERRPC,CSR,RCSR,0
3381 002152  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
```

ERROR POINTER TABLE

```
3382                                    ;ITEM 43
3383 002154  051570                             .WORD   EM52    ;EXPECTED INTERRUPT DID NOT OCCUR
3384 002156  053554                             .WORD   DH43    ;TEST #  ERR PC  CSRADR  CSRCONTENTS
3385 002160  055464                             .WORD   DT43    ;$TESTN,$ERRPC,CSR,RCSR,0
3386 002162  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3387
3388                                    ;ITEM 44
3389 002164  051657                             .WORD   EM54    ;BAR IS WRONG
3390 002166  053424                             .WORD   DH26    ;TEST #  ERR PC  BARADR  BAREXP  BARCONTENTS
3391 002170  055434                             .WORD   DT26    ;$TESTN,$ERRPC,BAR,EBAR,RBAR,
3392 002172  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3393
3394                                    ;ITEM 45
3395 002174  051674                             .WORD   EM55    ;BAD DATA IN BDR
3396 002176  053500                             .WORD   DH34    ;TEST #  ERR PC  BDRADR  BDREXP  BDRCONTENTS
3397 002200  055450                             .WORD   DT34    ;$TESTN,$ERRPC,BDR,EBDR,RBDR,
3398 002202  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3399
3400                                    ;ITEM 46
3401 002204  051752                             .WORD   EM57    ;BUFFER DATA NOT CORRECT
3402 002206  053776                             .WORD   DH57    ;                       CHECK   CHECK   INPUT   INPUT
3403                                                            ;TEST #  ERR PC  BUFADR  BUFDAT  BUFADR  BUFDAT  CSRADR
3404 002210  055536                             .WORD   DT57    ;$TESTN,$ERROC,$TMP4,$TMP2,$TMP5,$TMP3,CSR,0
3405 002212  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3406
3407                                    ;ITEM 47
3408 002214  052002                             .WORD   EM60    ;TOO MANY WORDS WERE TRANSFERED
3409 002216  054143                             .WORD   DH60    ;                       DIDNOT
3410                                                            ;TEST #  ERR PC  EXPECT  ADRESS  CSRADR
3411 002220  055556                             .WORD   DT60    ;$TESTN,$ERRPC,$TMP2,$TMP3,CSR,0
3412 002222  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3413
3414                                    ;ITEM 50
3415 002224  052041                             .WORD   EM61    ;UNEXPECTED TRAP OR INTERRUPT TO TRAP ADDRESS BELOW
3416 002226  054241                             .WORD   DH61    ;TEST #  ERR PC  WCRADR  OLDPC   TRAP ADR
3417 002230  055572                             .WORD   DT61    ;$TESTN,$ERRPC,WCR,OLDPC2,BDVECT,0
3418 002232  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3419
3420                                    ;ITEM 51
3421 002234  052124                             .WORD   EM62    ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
3422 002236  054312                             .WORD   DH62    ;TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  CSREXP  CSRRCV  BAREXP  BAR
3423 002240  055606                             .WORD   DT62    ;$TESTN,$ERRPC,WCR,EWCR,RWCR,ECSR,RCSR,EBAR,RBAR,0
3424 002242  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3425
3426                                    ;!ITEM 52
3427 002244  052173                             .WORD   EM63    ;DR11 INTERRUPTED AT WRONG LEVEL
3428 002246  054421                             .WORD   DH63    ;TEST #  ERR PC  EXPLVL  RCVLVL  CSRADR
3429 002250  055632                             .WORD   DT63    ;$TESTN,$ERRPC,DRLEV,LEVEL,CSR,0
3430 002252  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
3431
3432                                    ;ITEM 53
3433 002254  051213                             .WORD   EM43    ;DR11 FAILED TO INTERRUPT
3434 002256  054470                             .WORD   DH64    ;TEST #  ERR PC  EXPLVL  CSRADR
3435 002260  055646                             .WORD   DT64    ;$TESTN,$ERRPC,$TMP1,CSR,0
3436 002262  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
```

ERROR POINTER TABLE

```
3437                                    ;ITEM 54
3438 002264  052233                             .WORD   EM65    ;2-N CYCLE BURST SWITCH IN WRONG POSITION
3439 002266  054527                             .WORD   DH65    ;TEST #  ERR PC  CSR DR  EIREXP  EIRRCV
3440 002270  055660                             .WORD   DT65    ;$TESTN,$ERRPC,CSR,EEIR,REIR,0
3441 002272  000000                             .WORD   0       ;PRINT ALL DATA OCTAL
```

ERROR POINTER TABLE

```
3442 002274                    ER200:                  ;THIS IS THE STARTING POINT FOR ERROR MESSAGES 201
3443                                                   ;THROUGH 277.  THEY ARE USED FOR MULTIPLE ERROR MESSAGES.
3444                           ;ITEM 201
3445 002274  051752                     .WORD   EM57    ;BUFFER DATA NOT CORRECT
3446 002276  053776                     .WORD   DH57    ;                       CHECK   CHECK   INPUT   INPUT
3447                                                   ;TEST #  ERR PC  BUFADR  BUFDAT  BUFADR  BUFDAT  CSRADR
3448 002300  055536                     .WORD   DT57    ;$TESTN,$ERRPC,$TMP4,$TMP2,$TMP5,$TMP3,CSR,0
3449 002302  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
3450
3451                           ;ITEM 202
3452 002304  052347                     .WORD   EM202   ;CSR PATTERN NOT CORRECT
3453 002306  054645                     .WORD   DH202   ;TEST #  ERR PC  CSRADR  PATLDD  CSREXP  CSRRCV
3454 002310  055710                     .WORD   DT202   ;$TESTN,$ERRPC,CSR,BUT,ECSR,RCSR,0
3455 002312  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
3456
3457                           ;ITEM 203
3458 002314  050402                     .WORD   EM26    ;BIT PATTERN TEST FAILED IN BAR
3459 002316  053424                     .WORD   DH26    ;TEST #  ERR PC  BARADR  BAREXP  BARCONTENTS
3460 002320  055434                     .WORD   DT26    ;$TESTN,$ERRPC,BAR,EBAR,RBAR,
3461 002322  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
3462
3463                           ;ITEM 204
3464 002324  050441                     .WORD   EM27    ;WCR DATA PATTERN NOT CORRECT
3465 002326  053350                     .WORD   DH23    ;TEST #  ERR PC  WCRADR  WCREXP  WCRCONTENTS
3466 002330  055420                     .WORD   DT23    ;$TESTN,$ERRPC,WCR,EWCR,RWCR,0
3467 002332  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
3468
3469                           ;ITEM 205
3470 002334  050773                     .WORD   EM36    ;BDR PATTERN NOT CORRECT
3471 002336  053500                     .WORD   DH34    ;TEST #  ERR PC  BDRADR  BDREXP  BDRCONTENTS
3472 002340  055450                     .WORD   DT34    ;$TESTN,$ERRPC,BDR,EBDR,RBDR,0
3473 002342  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
3474
3475                           ;ITEM 206
3476 002344  051631                     .WORD   EM53    ;WCR NOT EQUAL TO ZERO
3477 002346  055064                     .WORD   DH210   ;TEST #  ERR PC  WCRADR  WCRCONTENTS
3478 002350  055760                     .WORD   DT210   ;$TESTN,$ERRPC,WCR,RWCR,0
3479 002352  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
3480
3481                           ;ITEM 207
3482 002354  051657                     .WORD   EM54    ;BAR IS WRONG
3483 002356  053424                     .WORD   DH26    ;TEST #  ERR PC  BARADR  BAREXP  BARCONTENTS
3484 002360  055434                     .WORD   DT26    ;$TESTN,$ERRPC,BAR,EBAR,RBAR,
3485 002362  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
3486
3487                           ;ITEM 210
3488 002364  051714                     .WORD   EM56    ;DATA NOT TRANSFERED CORRECTLY
3489 002366  053717                     .WORD   DH56    ;TEST #  ERR PC  NPR1AD  NPR1EX  NPR1RC  CSRADR
3490 002370  055520                     .WORD   DT56    ;$TESTN,$ERRPC,ANPR1,ENPR1,NPR1,CSR,0
3491 002372  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
3492
3493                           ;ITEM 211
3494 002374  052377                     .WORD   EM211   ;BDR AND-OR WCR AND-OR BAR ARE INCORECT
3495 002376  055130                     .WORD   DH211   ;TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  BDREXP  BDRRCV  BAREXP  BAR
3496 002400  055772                     .WORD   DT211   ;$TESTN,$ERRPC,WCR,EWCR,RWCR,ECSR,RCSR,EBAR,RBAR,0
3497 002402  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
```

ERROR POINTER TABLE

```
3498
3499 002404  051352                    ;ITEM 212
                                .WORD   EM46    ;FUNCTION BITS DIDN'T INCREMENT IN MAINT MODE
3500 002406  053274                     .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
3501 002410  055404                     .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
3502 002412  000000                     .WORD   0       ;PRINT ALL DATA OCTAL
```

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

002414  000000                  QTYBRD: .WORD   0       ;TOTAL # DR11 BOARDS BEING TESTED (DEFAULT = 0)

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  000C00                  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       ;BP 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  042644                  INBUF:  .WORD   XINBUF
002620  043646                  CHKBUF: .WORD   XCHKBU
002622  000000                  BUFLEN: .WORD   0
```

STORAGE LOCATIONS

```
3560 002624  000000                LENCHK: .WORD   0
3561 002626  000000                BRWAIT: .WORD   0
3562 002630  000000                WCLEN:  .WORD   0
3563 002632  000000                RDYCHK: .WORD   0
3564 002634  177560                TKS:    .WORD   177560
3565 002636  177562                TKB:    .WORD   177562
3566 002640  177564                TPS:    .WORD   177564
3567 002642  177566                TPB:    .WORD   177566
3568 002644  000000                MSG:    .WORD   0
3569 002646  000000                ADDR:   .WORD   0
3570 002650  000000                MESSAG: .WORD   0
3571 002652  000000                FLAG:   .WORD   0
3572 002654  000000                FNCCNT: .WORD   0
3573 002656  000000                INBUF1: .WORD   0
3574 002660  000000                TIME:   .WORD   0       ;GENERAL PURPOSE TIMER
3575 002662  000000                LOOP:   .WORD   0       ;GENERAL PURPOSE LOOP COUNTER
3576 002664  000000                ANSWER: .WORD   0
3577 002666  000000                BDFAIL: .WORD   0
3578 002670  000000                MANSIZ: .WORD   0
3579 002672  000000                OLDPC1: .WORD   0       ;LOCATION TO STORE RETURN PC IN SUBROUTINES WITH ERROR CALLS
3580 002674  000000                OLDPS1: .WORD   0       ;LOCATION TO STORE PS
3581 002676  000000                OLDPC2: .WORD   0       ;LOCATION TO STORE RETURN PC IN SUBROUTINES WITH ERROR CALLS
3582 002700  000000                OLDPS2: .WORD   0       ;LOCATION TO STORE PS
3583 002702  000000                OFL:    .WORD   0       ;FIRST CHAR FLAG
3584 002704  000000                LRGSIC: .WORD   0       ;LOCATION FOR LARGEST NUMBER CHARACTER FOR THE READ SUBROUTINE
3585 002706  000000                EOPLOC: .WORD   0       ;LOCATION TO HOLD FLAG DECIDING IF EOP MSGS ARE TO BE PRINTED
3586 002710  000000                BITTST: .WORD   0       ;LOCATION TO PUT THE BIT STATE TO PRINT - USED BY SUBROUTINE PSTATE
3587 002712  000000                MEMGMT: .WORD   0       ;LOCATION TO HOLD FLAG SAYING MEMORY MANAGEMENT IS AVAILABLE
3588 002714  000000                ERRCNT: .WORD   0       ;LOCATION TO HOLD TOTAL ERRORS FOR A PARTICULAR TEST
3589 002716  000000                $PASS2: .WORD   0       ;LOCATION TO HOLD OVERFLOW PASS NUMBER
```

DEVICE DESCRIPTOR WORD BIT DESCRIPTION

```
                                        .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
```

SUBROUTINE TO INPUT A CHARACTER OR UP TO A 6 DIGIT NUMBER

```
                                        .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 OR 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  032555                  TSTB    CHARCT          ;SEE IF A CHARACTER WAS INPUTED DURING PRINTING
002730  001406                          BEQ     1$              ;BRANCH TO INPUT A CHARACTER IF NOT
002732  113737  032555  002664          MOVB    CHARCT,ANSWER   ;MOVE THE CHARACTER TO THE ANSWER LOCATION
002740  105037  032555                  CLRB    CHARCT          ;CLEAR THAT GUY
002744  000403                          BR      2$              ;GO CHECK IT OUT
002746  104410                  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  037175                  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  037064                  TYPE    ,ESCAPE         ;TYPE:  '<ESC>'
003006  000444                          BR      6$              ;KICK OUT OF THIS ROUTINE
003010  113737  002664  037173  4$:     MOVB    ANSWER,LETNCR   ;MOVE CHARACTER FOR PRINTING
003016  104401  037173                  TYPE    ,LETNCR         ;GO TYPE THE INPUTED CHARACTER
003022  022737  000057  002664          CMP     #'/,ANSWER      ;SEE IF A NON-NUMERIC/ALPH CHARACTER WAS INPUTED
003030  100016                          BPL     5$              ;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
```

SUBROUTINE TO CAPTURE UNIT #, PASS # & TOTAL ERRORS

```
                                        .SBTTL  SUBROUTINE TO CAPTURE UNIT #, PASS # & TOTAL ERRORS
                                ;;**********************************************************************
                                ;*
                                ;*      THIS SUBROUTINE IS CALLED BY $EOP AND ENDCV TO SAVE THE UNIT NUMBER,
                                ;*      PASS NUMBER, AND TOTAL ERRORS FOR THAT DEVICE/PASS FOR SAVING WHENEVER
                                ;*      A DEVICE PASS CONTAINS ERRORS.
                                ;*
                                ;;**********************************************************************
003126  022737  047132  044650  ERCAPT: CMP     #ENDSTK,CAPNTR  ;SEE IF STACK IS FULL OF BULL
003134  001414                          BEQ     1$              ;KICK OUT IF FULL
003136  013700  044650                  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     $ERTTL,(R0)+    ;PUT TOTAL ERRORS ON STACK
003162  010037  044650                  MOV     R0,CAPNTR       ;RESTORE CAPNTR TO NEW POINTER VALUE
003166  000207                  1$:     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.
                                ;*
                                ;;**************************************************************
003170  005037  001466          FIXTBL: CLR     $DEVM           ;CLEAR THE DEVICE MASK
003174  000261                          SEC                     ;SET THE CARRY BIT AND
003176  006137  001466                  ROL     $DEVM           ;ROTATE IT INTO $DEVM FOR 1 BOARD
003202  022737  000001  002414          CMP     #1,QTYBRD       ;SEE IF ONLY 1 BOARD PRESENT
003210  001433                          BEQ     2$              ;KICK OUT IF SO - TABLE DOESN'T NEED FILLING
003212  013701  002414                  MOV     QTYBRD,R1       ;FILL ALL TABLE BOARD ENTRIES FROM FIRST
003216  005301                          DEC     R1              ;DECREMENT SINCE 1ST POSITION IS ALREADY FILLED
003220  005002                          CLR     R2              ;CLEAR INDEX TO SEND POINTER
003222  012703  000002                  MOV     #2,R3           ;PUT 2 IN INDEX TO RECEIVE POINTER
003226  000261                  1$:     SEC                     ;SET THE CARRY BIT AND
003230  006137  001466                  ROL     $DEVM           ;ROTATE IT INTO $DEVM
                002416          RA=REGADR               ;REDEFINE REGADR AS RA FOR SPACE REASONS
003234  016263  002416  002416          MOV     RA(R2),RA(R3)   ;TRANSFER ADDRESS TO NEXT POSITION AND
003242  062763  000010  002416          ADD     #10,RA(R3)      ;ADD 10 FOR NEXT POSITION
                002456          VA=VECADR               ;REDEFINE VECADR AS VA FOR SPACE REASONS
003250  016263  002456  002456          MOV     VA(R2),VA(R3)   ;TRANSFER VECTOR TO NEXT POSITION AND
003256  062763  000010  002456          ADD     #10,VA(R3)      ;ADD 10 FOR NEXT POSITION
003264  013763  001474  001474          MOV     $DDWO,$DDWO(R3) ;MOVE DEVICE DESCRIPTOR WORD TO NEXT POSITION
003272  022223                          CMP     (R2)+,(R3)+     ;UPDATE INDEX POINTERS
003274  005301                          DEC     R1              ;DECREMENT THE LOOP COUNTER
003276  001353                          BNE     1$              ;BRANCH BACK IF NOT DONE
003300  000207                  2$:     RTS     PC              ;EXIT
```

SUBROUTINE TO LOAD INBUF WITH AN INCREMENTING PATTERN

```
3706                                            .SBTTL  SUBROUTINE TO LOAD INBUF WITH AN INCREMENTING PATTERN
3707                                    ;;**************************************************************
3708                                    ;*
3709                                    ;*      THIS SUBROUTINE CLEARS THE FIRST LOCATION OF THE BUFFER AND LOADS
3710                                    ;*      NUMBERS STARTING WITH 1 INTO THE BUFFER.
3711                                    ;*
3712                                    ;;**************************************************************
3713 003302  013702  002616             LODBUF: MOV     INBUF,R2        ;MOVE STARTING ADDRESS OF INBUF TO R2
3714 003306  013703  002622                     MOV     BUFLEN,R3       ;MOVE LOOP COUNTER TO R3
3715 003312  005203                             INC     R3              ;CORRECT COUNTER
3716 003314  005001                             CLR     R1              ;CLEAR THE LOADING COUNTER
3717 003316  010122                     1$:     MOV     R1,(R2)+        ;LOAD NEXT BUFFER WORD
3718 003320  005201                             INC     R1              ;INCREMENT THE LOADING COUNTER
3719 003322  005303                             DEC     R3              ;DECREMENT THE LOOP COUNTER AND
3720 003324  001374                             BNE     1$              ;BRANCH BACK IF NOT DONE
3721 003326  000207                             RTS     PC              ;EXIT
```

SUBROUTINE TO LOAD THE CHKBUF WITH EVEN #'S STARTING WITH 0

```
                                        .SBTTL  SUBROUTINE TO LOAD THE CHKBUF WITH EVEN #'S STARTING WITH 0
                                ;;**************************************************************
                                ;*
                                ;*      THIS SUBROUTINE CLEARS THE FIRST LOCATION OF THE BUFFER AND LOADS
                                ;*      EVEN NUMBERS STARTING WITH 0 INTO THE BUFFER.
                                ;*
                                ;;**************************************************************
003330  013702  002620          CHKBFF: MOV     CHKBUF,R2       ;STARTING ADDRESS OF CHECK-BUFFER TO R2
003334  013701  002622                  MOV     BUFLEN,R1       ;MOVE LOOP COUNTER TO R1
003340  005003                          CLR     R3              ;WIPE OUT R3
003342  010322                  1$:     MOV     R3,(R2)+        ;MOVE R3 TO CHKBUF ADDRESS AND INC BY 2
003344  010322                          MOV     R3,(R2)+        ;MOVE R3 TO NEXT CHKBUF ADDRESS AND INC BY 2
003346  022341                          CMP     (R3)+,-(R1)     ;ADD 2 TO NUMBER FOR BUFFER & SUBTRACT 2 FROM LOOP COUNTER
003350  005701                          TST     R1              ;SEE IF R1 HAS REACHED ZERO YET
003352  001373                          BNE     1$              ;BRANCH BACK IF NOT DONE
003354  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
                                ;*
                                ;;*****************************************************************
003356  042777  000100  177136  INTA:   BIC     #IE,@CSR        ;CLEAR IE
003364  013702  002622          MOV     BUFLEN,R2       ;BUFFER LENGTH TO R2
003370  063702  002622          ADD     BUFLEN,R2       ;NUMBER OF XFERS TIMES 2
003374  063702  002616          ADD     INBUF,R2        ;CORRECT BAR
003400  017737  177116  002560  MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
003406  032737  010000  002560  BIT     #MA,RCSR        ;SEE IF WE ARE IN MAINTENANCE MODE
003414  001005                  BNE     1$              ;BRANCH AROUND CABLE TEST IF WE ARE, BIT 0 WILL BE CLEAR
003416  032737  000004  002720  BIT     #BIT2,DDW       ;SEE IF THERE IS A CABLE
003424  001401                  BEQ     1$              ;BRANCH IF NO CABLE
003426  005202                  INC     R2              ;CABLE MODE TESTING LEAVES BIT 0 OF BAR SET. CHECK ODD ADRS
003430  017737  177064  002566  1$:     MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
003436  005037  002602          CLR     EWCR            ;CLEAR EXPECTED LOCATION
003442  010237  002600          MOV     R2,EBAR         ;MOVE EXPECTED DATA TO EBAR
003446  013737  002560  002572  MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
003454  042737  100000  002572  BIC     #ER,ECSR        ;MAKE SURE ERROR BIT IS CLEAR
003462  052737  000200  002572  BIS     #RY,ECSR        ;MAKE SURE READY BIT IS SET
003470  017737  177022  002570  MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
003476  001012                  BNE     2$              ;BRANCH TO RETURN WITHOUT UPDATING PC SO ERROR WILL CALL
003500  023737  002560  002572  CMP     RCSR,ECSR       ;DOES CSR CONTAIN WHAT IT SHOULD
003506  001006                  BNE     2$              ;BRANCH TO RETURN WITHOUT UPDATING PC SO ERROR WILL CALL
003510  023737  002566  002600  CMP     RBAR,EBAR       ;DOES BAR CONTAIN WHAT IT SHOULD
003516  001002                  BNE     2$              ;BRANCH TO RETURN WITHOUT UPDATING PC SO ERROR WILL CALL
003520  062716  000002          ADD     #2,(SP)         ;CORRECT PC RETURN TO AFTER THE ERROR CALL
003524  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
                                ;*
                                ;;**************************************************************
003526  011637  001362          DATCHK: MOV     (SP),$TMP1      ;SAVE PC RETURN
003532  013702  002620                  MOV     CHKBUF,R2       ;STARTING ADDRESS OF CHECK BUFFER TO R2
003536  013703  002616                  MOV     INBUF,R3        ;STARTING ADDRESS OF IN BUFFER TO R3
003542  005037  002624                  CLR     LENCHK          ;CLEAR LENGTH CHECK
003546  005237  002624          DATCH1: INC     LENCHK          ;MAKE A COMPARISON
003552  022223                          CMP     (R2)+,(R3)+     ;IS THE DATA CORRECT?
003554  001423                          BEQ     DATCH2          ;BRANCH IF OK
003556  013716  001362                  MOV     $TMP1,(SP)      ;RESTORE ORIGINAL PC RETURN
003562  016237  177776  001364          MOV     -2(R2),$TMP2    ;MOVE CHECK BUFFER CONTENTS TO $TMP2
003570  010237  001370                  MOV     R2,$TMP4        ;MOVE ADDRESS +2 TO $TMP4
003574  162737  000002  001370          SUB     #2,$TMP4        ;CORRECT SO IT POINTS TO ADDRESS CAUSING ERROR
003602  016337  177776  001366          MOV     -2(R3),$TMP3    ;MOVE INPUT BUFFER CONTENTS TO $TMP3
003610  010337  001372                  MOV     R3,$TMP5        ;MOVE ADDRESS +2 TO $TMP5
003614  162737  000002  001372          SUB     #2,$TMP5        ;CORRECT SO IT POINTS TO ADDRESS CAUSING ERROR
003622  000410                          BR      DATCHX          ;RETURN TO ERROR CALL - PC ON STACK ALREADY POINTS THERE
003624  023737  002624  002622  DATCH2: CMP     LENCHK,BUFLEN   ;SEE IF THE BUFFER HAS BEEN CHECKED
003632  001345                          BNE     DATCH1          ;GO BACK FOR ANOTHER TRY IF NOT
003634  013716  001362                  MOV     $TMP1,(SP)      ;RESTORE PC RETURN
003640  062716  000006                  ADD     #6,(SP)         ;CORRECT IT SO RETURN IS AFTER THE ERROR CALL
003644  000207                  DATCHX: RTS     PC              ;RETURN
```

SUBROUTINE TO RESTORE DR11 INT VECT & SET CPU PRIORITY TO 7.

```
                                        .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.
                                ;*
                                ;;**********************************************************************
003646  012737  000340  177776  CLENUP: MOV     #LEVEL7,PSW     ;RESTORE CPU TO PRIORITY LEVEL 7
003654  012777  010000  176640          MOV     #MA,@CSR        ;DO AN INIT CLEARING WCR, BAR & BDR BY SETTING
003662  005077  176634                  CLR     @CSR            ;AND CLEARING THE MAINT BIT AND CLEAR THE CSR
003666  000207                          RTS     PC              ;EXIT
```

```
3821                                            .SBTTL  SUBROUTINE TO CHECK FOR CABLE MODE AND ALTER EXPECTED DATA
3822                                    ;;**************************************************************
3823                                    ;*
3824                                    ;*      THIS SUBROUTINE CHECKS THE DDW (DEVICE DESCRIPTOR WORD) FOR THE CABLE
3825                                    ;*      BEING IN OR OUT AND SETS BITS 12, 10, 8 AND 6 IN THE EXPECTED DATA
3826                                    ;*      LOCATION IF THE CABLE IS OUT.
3827                                    ;*
3828                                    ;;**************************************************************
3829 003670  032737  000004  002720  CHKCAB: BIT     #BIT2,DDW       ;CHECK FOR CABLE STATUS
3830 003676  001006                          BNE     1$              ;EXIT IF CABLE DOES EXIST
3831 003700  052737  127000  002572          BIS     #127000,ECSR    ;SET BITS 12, 10, 8, 7 & 6 - CABLE DOESN'T EXIST
3832 003706  052737  127000  002536          BIS     #127000,BUT     ;SET BITS 12, 10, 8, 7 & 6 IN BITS UNDER TEST LOCATION TOO
3833 003714  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
                                ;*
                                ;*****************************************************************
003716  011637  001362          DATOCK: MOV     (SP),$TMP1      ;SAVE PC RETURN
003722  012702  052525                  MOV     #52525,R2       ;DATO NUMBER 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          DATOC1: INC     LENCHK          ;MAKE A COMPARISON
003742  020223                          CMP     R2,(R3)+        ;IS THE DATA CORRECT?
003744  001415                          BEQ     DATOC2          ;BRANCH IF OK
003746  013716  001362                  MOV     $TMP1,(SP)      ;MOVE OLD PC RETURN TO STACK
003752  010237  001364                  MOV     R2,$TMP2        ;MOVE EXPECTED DATA TO $TMP2
003756  016337  177776  001366          MOV     -2(R3),$TMP3    ;MOVE RECEIVED DATA TO $TMP3
003764  010337  001370                  MOV     R3,$TMP4        ;MOVE ADDRESS +2 TO $TMP4
003770  162737  000002  001370          SUB     #2,$TMP4        ;CORRECT ADDRESS SO IT POINTS TO ADDRESS CAUSING ERROR
003776  000431                          BR      DATOCX          ;RETURN TO ERROR CALL
004000  023737  002624  002622  DATOC2: CMP     LENCHK,BUFLEN   ;SEE IF THE BUFFER HAS BEEN CHECKED
004006  001353                          BNE     DATOC1          ;BUFFER CHECKED?
004010  020223                          CMP     R2,(R3)+        ;CHECK END OF BUFFER + 1
004012  001017                          BNE     1$              ;BRANCH IF NOT LOADED
004014  010237  001364                  MOV     R2,$TMP2        ;MOVE EXPECTED DATA TO $TMP2
004020  016337  177776  001366          MOV     -2(R3),$TMP3    ;MOVE RECEIVED DATA TO $TMP3
004026  010337  001370                  MOV     R3,$TMP4        ;MOVE ADDRESS +2 TO $TMP4
004032  162737  000002  001370          SUB     #2,$TMP4        ;CORRECT ADDRESS SO IT POINTS TO ADDRESS CAUSING ERROR
004040  013716  001362                  MOV     $TMP1,(SP)      ;CORRECT PC RETURN
004044  062716  000006                  ADD     #6,(SP)         ;POINT TO 2ND ERROR CALL AFTER JSR PC,DATOCK
004050  000404                          BR      DATOCX          ;RETURN TO ERROR CALL
004052  013716  001362          1$:     MOV     $TMP1,(SP)      ;RESTORE RETURN ADDRESS
004056  062716  000010                  ADD     #10,(SP)        ;POINT TO PROPER RETURN AFTER THE ERROR CALLS
004062  000207                  DATOCX: RTS     PC              ;EXIT
```

SUBROUTINE TO CLEAR IE AND HALT IF ERROR IS SET

```
                                        .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
                                ;*
                                ;;*****************************************************************
004064  042777  000100  176430  ERRCHK: BIC     #IE,@CSR        ;CLEAR IE
004072  017737  176424  002560           MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
004100  013737  002560  002572           MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
004106  042737  100000  002572           BIC     #ER,ECSR        ;CLEAR THE ERROR BIT
004114  052737  000200  002572           BIS     #RY,ECSR        ;SET THE READY BIT
004122  013701  002560                   MOV     RCSR,R1         ;MOVE DATA TO R1 FOR CHECKING
004126  012737  000200  001360           MOV     #RY,$TMP0       ;MOVE EXPECTED DATA TO $TMP0
004134  042701  077577                   BIC     #77577,R1       ;CLEAR ALL BUT THE ERROR AND READY BITS
004140  022701  000200                   CMP     #RY,R1          ;SEE IF ERROR BIT IS CLEAR AND READY IS SET
004144  001002                           BNE     1$              ;BRANCH AROUND PC CORRECTION SO ERROR WILL CALL
004146  062716  000002                   ADD     #2,(SP)         ;CORRECT PC RETURN - DATA OK
004152  000207                   1$:     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.
                                ;*
                                ;;**************************************************************
004154  012700  004216          BPINIT: MOV     #BPTINT,R0      ;POINT R0 TO TABLE OF BPT INIT LOCATIONS
004160  012701  000003                  MOV     #3,R1           ;DO 3 SETS OF ".+2" AND "BPT" SETUPS
004164  012002                  1$:     MOV     (R0)+,R2        ;MOVE START ADDRESS TO R2
004166  012003                          MOV     (R0)+,R3        ;MOVE END ADDRESS TO R3
004170  010204                          MOV     R2,R4           ;MOVE ADDRESS TO R4
004172  005724                          TST     (R4)+           ;INCREMENT R4 TO PRODUCE THE ".+2" NUMBER
004174  010422                  2$:     MOV     R4,(R2)+        ;MOVE THE NUMBER TO THE LOCATION
004176  012722  000003                  MOV     #BPT,(R2)+      ;MOVE "BPT" TO THE NEXT LOCATION
004202  022424                          CMP     (R4)+,(R4)+     ;ADD 4 TO R4
004204  020203                          CMP     R2,R3           ;SEE IF WE HAVE DONE ALL FOR THIS LOCATION
004206  001372                          BNE     2$              ;BRANCH BACK FOR ANOTHER TRANSFER IF NOT
004210  005301                          DEC     R1              ;DECREMENT R1
004212  001364                          BNE     1$              ;BRANCH BACK IF 3 GROUPS NOT DONE
004214  000207                          RTS     PC              ;EXIT

004216  000004  000014  000050  BPTINT: .WORD   4,14,50         ;ADDRESSES USED TO PUT ".+2" & "BPT" BACK
004224  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. (ADDED TO REV 'F'): ALSO DOES A MINI-TEST OF THE CSR TO MAKE
                                ;*      SURE BOARD IS A DR11, EXITING WITH A MODIFIED RETURN IF IT FAILS THIS
                                ;*      MINI-TEST.
                                ;*
                                ;;**************************************************************
004232  012737  000340  002540  DRGET:  MOV     #LEVEL7,LEVEL   ;MOVE PRIORITY 7 TO LEVEL
004240  012703  004732                   MOV     #LEVELS,R3      ;MOVE ADDRESS OF PRIORITY LEVELS TO R3
004244  012704  000004                   MOV     #4,R4           ;DO 4 PRIORITY CHECKS
004250  012737  000400  002660  1$:      MOV     #400,TIME       ;SET UP WAIT LOOP COUNTER
004256  012710  010000                   MOV     #MA,(R0)        ;SET THE MAINTENANCE BIT AND
004262  005010                           CLR     (R0)            ;CLEAR TO DO AN INIT
004264  013737  000014  001362           MOV     BPTVCT,$TMP1    ;SAVE BPT TRAP VECTOR
004272  012737  004520  000014           MOV     #9$,BPTVCT      ;INTERRUPTS TO 9$
004300  012337  177776                   MOV     (R3)+,PSW       ;SET CPU PRIORITY TO NEXT LEVEL
004304  000240                           NOP                     ;KILL A LITTLE TIME
004306  012710  000105                   MOV     #IE+F2+GO,(R0)  ;SET IE, FNCT2 AND GO ATTEMPTING AN INTERRUPT
004312  005337  002660           2$:     DEC     TIME            ;DECREMENT TIME
004316  001375                           BNE     2$              ;BRANCH BACK UNTIL ZERO
004320  012737  000340  177776           MOV     #LEVEL7,PSW     ;SET CPU PRIORITY BACK TO 7
004326  013737  001362  000014           MOV     $TMP1,BPTVCT    ;RESTORE BPT TRAP VECTOR
004334  162737  000040  002540           SUB     #40,LEVEL       ;PUT LOCATION 'LEVEL' AT NEXT PRIORITY - INTERRUPT FAILED
004342  005304                           DEC     R4              ;DECREMENT LOOP COUNTER
004344  001341                           BNE     1$              ;BRANCH BACK IF NOT ALL PRIORITY LEVELS CHECKED YET
                                ;;**************************************************************
                                ; IF ROUTINE GETS HERE AND YOU ARE NOT IN A MANUFACTURING ENVIRONMENT, AND
                                ; IT IS A DR11, YOU PROBABLY HAVE A BAD BOARD.  PROMPTING USER FOR THE VECTOR
                                ; WILL ALLOW ONE MORE OPORTUNITY TO CHECK THE DR11 WITHOUT HAVING TO GO INTO
                                ; MANUAL MODE.  IT IS EVEN *MORE* IMPORTANT FOR THE USER TO *MAKE*SURE* THE
                                ; REST OF THE DATA (I.E. CABLE STATE, ETC.) IS AS THE MODULE IS SET UP.
                                ;;**************************************************************
004346  132737  000001  001430  3$:      BITB    #BIT0,$ENV      ;SEE IF WE ARE UNDER APT                    ;DPM001
004354  001165                           BNE     14$             ;BRANCH OUT IF SO                           ;DPM001
004356  032777  010000  174754           BIT     #BIT12,@SWR     ;IS THIS A MANUFACTURING ENVIRONMENT?       ;DPM001
004364  001161                           BNE     14$             ;BRANCH IF SO - ASSUMED TO NOT BE A DR11    ;DPM001
004366  104401  004742                   TYPE    ,ASK4VC         ;TYPE:  'INPUT VECTOR ? '                   ;DPM001
004372  012703  000003                   MOV     #3,R3           ;EXPECT 3 DIGITS                            ;DPM001
004376  004737  002722                   JSR     PC,READ         ;GO READ VECTOR INPUT                       ;DPM001
004402  020427  000300                   CMP     R4,#300         ;SEE IF VECTOR INPUTED IS BELOW 300         ;DPM001
004406  100403                           BMI     4$              ;BRANCH IF LESS THAN 300                    ;DPM001
004410  022704  000774                   CMP     #774,R4         ;SEE IF VECTOR IS ABOVE 774                 ;DPM001
004414  100003                           BPL     5$              ;BRANCH IF NOT TO LOAD INFO                 ;DPM001
004416  104401  004762           4$:     TYPE    ,OUTORG         ;TYPE:  'VECTOR OUT OF 300-774 RANGE'       ;DPM001
004422  000751                           BR      3$              ;BRANCH BACK FOR ANOTHER TRY                ;DPM001
004424  032704  000003           5$:     BIT     #3,R4           ;MAKE SURE BITS 0 & 1 ARE CLEAR             ;DPM001
004430  001403                           BEQ     55$             ;BRANCH IF OK                               ;DPM001
004432  104401  005017                   TYPE    ,NOTVEC         ;TYPE:  'VECTOR IS NOT A MULTIPLE OF 4'     ;DPM001
004436  000743                           BR      3$              ;BRANCH BACK FOR ANOTHER TRY                ;DPM001
004440  010412                   55$:    MOV     R4,(R2)         ;MOVE INPUTED VECTOR TO LOCATION            ;DPM001
004442  104401  005056           6$:     TYPE    ,ASK4PR         ;TYPE:  'INPUT BOARD BR PRIORITY'           ;DPM001
004446  012703  000001                   MOV     #1,R3           ;EXPECT 1 DIGIT                             ;DPM001
004452  004737  002722                   JSR     PC,READ         ;GO GET PRIORITY                            ;DPM001
004456  022704  000003                   CMP     #3,R4           ;SEE IF INPUT IS BELOW 4                    ;DPM001
```

```
3979 004462  100003                      BPL     7$              ;BRANCH IF SO TO TELL USER IT IS WRONG  ;DPM001
3980 004464  020427  000007              CMP     R4,#7           ;SEE IF INPUT IS ABOVE 7                ;DPM001
3981 004470  100403                      BMI     8$              ;BRANCH TO LOAD DATA IF OK              ;DPM001
3982 004472  104401  005107      7$:     TYPE    ,PROUT          ;TYPE:  'PRIORITY OUT OF 4-7 RANGE'     ;DPM001
3983 004476  000761                      BR      6$              ;BRANCH BACK TO TRY AGAIN               ;DPM001
3984 004500  006304              8$:     ASL     R4              ;SHIFT LEVEL OVER 5 PLACES              ;DPM001
3985 004502  006304                      ASL     R4              ;                                       ;DPM001
3986 004504  006304                      ASL     R4              ;                                       ;DPM001
3987 004506  006304                      ASL     R4              ;                                       ;DPM001
3988 004510  006304                      ASL     R4              ;                                       ;DPM001
3989 004512  010437  002540              MOV     R4,LEVEL        ;PUT PRI LEVEL INTO THE LEVEL LOCATION  ;DPM001
3990 004516  000410                      BR      10$             ;BRANCH OVER DR11 INTERRUPT HANDLER     ;DPM001
3991 004520  012737  000340  177776  9$: MOV     #LEVEL7,PSW     ;SET CPU PRIORITY BACK TO 7
3992 004526  011612                      MOV     (SP),(R2)       ;MOVE VECTOR+4 TO VECTOR LOCATION       ;DPM001
3993 004530  162712  000004              SUB     #4,(R2)         ;VECTOR IS WRONG - CORRECT IT
3994 004534  062706  000010              ADD     #10,SP          ;CLEAN STACK                            ;DPM001
3995 004540  013711  002540      10$:    MOV     LEVEL,(R1)      ;STORE THE OBTAINED PRIORITY LEVEL      ;DPM001
3996 004544  013737  001362  000014      MOV     $TMP1,BPTVCT    ;RESTORE BPT TRAP VECTOR
3997 004552  000261                      SEC                     ;SET THE CARRY BIT FOR THE 'ROL'        ;DPM001
3998 004554  006137  001466              ROL     $DEVM           ;INDICATE DEVICE EXISTENCE IN DEVICE MAP
3999 004560  005237  002414              INC     QTYBRD          ;INCREMENT DEVICE COUNT
4000 004564  052710  100000              BIS     #EIR,(R0)       ;GO TO EIR TO GET B/W STATE
4001 004570  011037  002562              MOV     (R0),REIR       ;MOVE EIR TO REIR
4002 004574  005010                      CLR     (R0)            ;GO BACK TO CSR
4003 004576  012710  010000              MOV     #MA,(R0)        ;SET THE MAINT BIT
4004 004602  005010                      CLR     (R0)            ;DO AN INIT
4005 004604  032737  000001  002562      BIT     #BIT0,REIR      ;TEST FOR B/W STATE
4006 004612  001003                      BNE     11$             ;BRANCH IF A W
4007 004614  052711  000001              BIS     #BIT0,(R1)      ;SET STATE IN DEVICE DESCRIPTOR WORD
4008 004620  000406                      BR      12$             ;GO TO CABLE STATUS TEST
4009 004622  032737  000400  002562  11$: BIT     #N2,REIR        ;CHECK 2/N CYCLE STATE
4010 004630  001402                      BEQ     12$             ;BRANCH IF 2 CYCLE
4011 004632  052711  000002              BIS     #BIT1,(R1)      ;N CYCLE - SET BIT IN DEVICE DESC
4012 004636  011037  002560      12$:    MOV     (R0),RCSR       ;MOVE RECEIVED DATA TO RCSR TO GET CABLE STATUS
4013 004642  032737  127000  002560      BIT     #127000,RCSR    ;CHECK IF ANY BITS ARE SET - THEY ARE IF NO CABLE
4014 004650  001015                      BNE     13$             ;BRANCH IF NO CABLE
4015 004652  112710  000004              MOVB    #F2,(R0)        ;CABLE IS POSSIBLY IN - SET FNCT2
4016 004656  011037  002560              MOV     (R0),RCSR       ;MOVE RECEIVED DATA TO RCSR
4017 004662  052710  010000              BIS     #MA,(R0)        ;SET THE MAINTENANCE BIT
4018 004666  005010                      CLR     (R0)            ;CLEAR THE CSR TO DO AN INIT
4019 004670  032737  020000  002560      BIT     #AT,RCSR        ;TEST THE ATTN BIT
4020 004676  001402                      BEQ     13$             ;BRANCH IF NOT SET - NO CABLE
4021 004700  052711  000004              BIS     #BIT2,(R1)      ;SET CABLE BIT IN DEVICE DESC
4022 004704  013704  006122      13$:    MOV     BDNUMB,R4       ;MOVE BOARD NUMBER TO R4                ;DPM001
4023 004710  005304                      DEC     R4              ;DECREMENT FOR CORRECT OFFSET           ;DPM001
4024 004712  006304                      ASL     R4              ;SHIFT LEFT TO PRODUCE WORD OFFSET      ;DPM001
4025 004714  010064  002416              MOV     R0,REGADR(R4)   ;MOVE CSR ADDRESS TO LOCATION AND       ;DPM001
4026 004720  162764  000004  002416      SUB     #4,REGADR(R4)   ;SUB 4 TO FORM ADDRESS FOR THIS MODULE  ;DPM001
4027 004726  022122                      CMP     (R1)+,(R2)+     ;R1 & R2 GET NEW DEVICE & VECTOR LOC'S  ;DPM001
4028 004730  000207              14$:    RTS     PC              ;EXIT
4029
4030 004732  000300  000240      LEVELS: .WORD   LEVEL6,LEVEL5   ;PRIORITY LEVELS TO LOAD INTO THE PSW
4031 004736  000200  000140              .WORD   LEVEL4,LEVEL3
4032 004742     111     116     120  ASK4VC: .ASCIZ  !INPUT VECTOR ? !                                      ;DPM001
4033 004762     126     105     103  OUTORG: .ASCIZ  !VECTOR OUT OF 300-774 RANGE!<CRLF>                    ;DPM001
4034 005017     126     105     103  NOTVEC: .ASCIZ  !VECTOR IS NOT A MULTIPLE OF 4!<CRLF>                  ;DPM001
4035 005056     111     116     120  ASK4PR: .ASCIZ  !INPUT BOARD BR PRIORITY !                             ;DPM001
```

```
                                                                E 6
4036 005107     120     122     111  PROUT:  .ASCIZ  !PRIORITY OUT OF 4-7 RANGE!<CRLF>                    ;DPM001
4037                                         .EVEN
```

```
                                        .SBTTL  SUBROUTINE TO PRINT THE AUTOSIZED BOARD CONFIGURATIONS
                                ;;*****************************************************************
                                ;*
                                ;*      THIS SUBROUTINE PRINTS THE BOARD CONFIGURATIONS FOUND BY ASIZE
                                ;*
                                ;;*****************************************************************
005142  005037  002662          TYPCNF: CLR     LOOP            ;CLEAR THE BOARD NUMBER COUNTER
005146  013705  001466                  MOV     $DEVM,R5        ;GET DEVICE MAP
005152  012701  002416                  MOV     #REGADR,R1      ;MOVE THE ADDRESS OF THE REGISTER ADDRESS TABLE TO R1
005156  012702  002456                  MOV     #VECADR,R2      ;MOVE THE ADDRESS OF THE VECTOR ADDRESS TABLE TO R2
005162  005003                          CLR     R3              ;CLEAR THE DEVICE DESCRIPTOR ADDRESS POINTER
005164  104401  037535                  TYPE    ,NOBUT          ;TYPE:  'NO. BOARDS UNDER TEST:  '
005170  013746  002414                  MOV     QTYBRD,-(SP)    ;MOVE THE QUANTITY TO THE STACK FOR TYPEOUT
005174  104405                          TYPDS                   ;TYPE THE NUMBER
005176  104401  037574                  TYPE    ,BRVWPC         ;TYPE THE HEADER
005202  012700  000020                  MOV     #16.,R0         ;SET UP LOOP COUNTER FOR 16 BOARDS
005206  032705  000001          1$:     BIT     #BIT0,R5        ;DEVICE UNDER TEST?
005212  001466                          BEQ     8$              ;BRANCH IF NO
005214  013746  002662                  MOV     LOOP,-(SP)      ;PUT BOARD # ON STACK FOR TYPEOUT
005220  104405                          TYPDS                   ;PRINT BOARD # (0 TO 16)
005222  104401  037361                  TYPE    ,SPACE6         ;TYPE 6 SPACE CHARACTERS
005226  011146                          MOV     (R1),-(SP)      ;SAVE REGISTER ADDRESS FOR TYPEOUT
005230  104402                          TYPOC                   ;PRINT DEVICE REGISTER ADDRESS
005232  104401  037361                  TYPE    ,SPACE6         ;TYPE 6 SPACE CHARACTERS
005236  011246                          MOV     (R2),-(SP)      ;SAVE VECTOR ADDRESS FOR TYPEOUT
005240  104403                          TYPOS                   ;PRINT VECTOR ADDRESS
005242     003     000                  .BYTE   3,0             ;PRINT 3 DIGITS, LEADING ZEROS SUPPRESSED
005244  104401  037370                  TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
005250  016337  001474  002720          MOV     $DDW0(R3),DDW   ;MOVE DEVICE DESCRIPTOR WORD TO DDW
005256  032737  000001  002720          BIT     #BIT0,DDW       ;TEST WHICH STATE, B OR W, FOR THIS BOARD
005264  001403                          BEQ     2$              ;GO PRINT W STATE IF W
005266  104401  037405                  TYPE    ,B              ;TYPE A 'B'
005272  000402                          BR      3$              ;GO TO NEXT CHECK
005274  104401  037407          2$:     TYPE    ,W              ;'TYPE A 'W'
005300  104401  037370          3$:     TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
005304  004737  006560                  JSR     PC,PNTPRI       ;PRINT DEVICE PRIORITY
005310  104401  037370                  TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
005314  032737  000002  002720          BIT     #BIT1,DDW       ;TEST 2/N CYCLE STATE
005322  001403                          BEQ     4$              ;GO PRINT 2 STATE IF 2
005324  104401  037422                  TYPE    ,N              ;TYPE AN 'N'
005330  000402                          BR      5$              ;GO TO NEXT CHECK
005332  104401  037420          4$:     TYPE    ,TWO            ;TYPE A '2'
005336  104401  037370          5$:     TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
005342  032737  000004  002720          BIT     #BIT2,DDW       ;TEST CABLE STATE
005350  001403                          BEQ     6$              ;GO PRINT 'NO' IF NO CABLE
005352  104401  037414                  TYPE    ,YES            ;TYPE 'YES'
005356  000402                          BR      7$              ;GO TO LOOP CHECK
005360  104401  037411          6$:     TYPE    ,NO             ;TYPE 'NO'
005364  104401  001405          7$:     TYPE    ,$CRLF          ;TYPE A <CRLF>
005370  005237  002662          8$:     INC     LOOP            ;INCREMENT BOARD COUNT FOR POSSIBLE NEXT PASS
005374  022122                          CMP     (R1)+,(R2)+     ;INCREMENT COUNTERS
005376  006205                          ASR     R5              ;SHIFT R5 TO THE RIGHT TO MOVE BOARD BIT INTO BIT 0
005400  062703  000002                  ADD     #2,R3           ;ADD 2 TO THE DEVICE DESCRIPTOR WORD POINTER
005404  005300                          DEC     R0              ;DECREMENT THE LOOP COUNTER AND
005406  001277                          BNE     1$              ;BRANCH BACK FOR CHECK IF 16 BOARDS NOT DONE
005410  104401  001405                  TYPE    ,$CRLF          ;TYPE ANOTHER <CRLF>
005414  000207                          RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO AUTO SIZE DR11 BOARD CONFIGURATION
                                ;;**********************************************************************
                                ;*
                                ;*      THIS SUBROUTINE AUTOSIZES THE BOARD CONFIGURATION IF PROMPTED BY THE
                                ;*      USER THAT SYSTEM IS OPERATING UNDER AN OPTION MANUFACTURING CONFIGURA-
                                ;*      TION (NO OTHER DEVICES HAVING CSR'S SIMILAR TO THAT OF A DR11), OR
                                ;*      PROMPTS USER FOR EACH DR11 ADDRESS (SEMI-AUTOSIZE).  IT THEN CALLS FOR
                                ;*      THE PRINTING OF THE CONFIGURATION IF THE TABLE WAS MADE BY THIS ROUTINE.
                                ;*
                                ;;**********************************************************************
005416  004737  004154          ASIZE:  JSR     PC,BPINIT       ;GO RESET THE ".+2" AND "BPT" LOCATIONS
005422  013737  000004  002672          MOV     TOVECT,OLDPC1   ;SAVE TIMEOUT VECTOR
005430  012737  005724  000004          MOV     #7$,TOVECT      ;TIMEOUTS TO 7$
005436  013737  000006  002674          MOV     TMOPSW,OLDPS1   ;SAVE TIMEOUT PS
005444  012737  000340  000006          MOV     #340,TMOPSW     ;TIMEOUT PRIORITY TO 7
005452  132737  000001  001430          BITB    #BIT0,$ENV      ;SEE IF WE ARE UNDER APT                    ;DPM001
005460  001004                          BNE     100$            ;BRANCH OUT IF SO                           ;DPM001
005462  032777  010000  173650          BIT     #BIT12,@SWR     ;DOES USER WANT 100% AUTOSIZE?              ;DPM001
005470  001416                          BEQ     1$              ;BRANCH IF NOT                              ;DPM001
005472  005037  002414          100$:   CLR     QTYBRD          ;CLEAR DEVICE COUNT                         ;DPM001
005476  005037  001466                  CLR     $DEVM           ;CLEAR THE DEVICE MAP                       ;DPM001
005502  012737  000001  006122          MOV     #1,BDNUMB       ;START WITH BOARD NUMBER 1                  ;DPM001
005510  012700  160014                  MOV     #160014,R0      ;MOVE START ADDRESS OF WINDOW TO R0         ;DPM001
005514  012701  001474                  MOV     #$DDW0,R1       ;LOAD DEVICE DESC ADDRESS                   ;DPM001
005520  012702  002456                  MOV     #VECADR,R2      ;POINT R2 TO UNIT #1 VECTOR ADRS LOC        ;DPM001
005524  000524                          BR      10$             ;GO START THE DR11 SEARCH                   ;DPM001
005526  005737  002414          1$:     TST     QTYBRD          ;SEE IF TABLE HAS BEEN LOADED               ;DPM001
005532  001414                          BEQ     2$              ;BRANCH TO LOAD TABLE IF NOT                ;DPM001
005534  104401  006401                  TYPE    ,DYWTLT         ;TYPE:  'DO YOU WISH TO CHANGE              ;DPM001
                                                                ;       THE CONFIGURATION (Y OR <CR>)'      ;DPM001
005540  012703  000001                  MOV     #1,R3           ;EXPECT 1 CHARACTER                         ;DPM001
005544  004737  002722                  JSR     PC,READ         ;GO READ 1 CHARACTER OF USER INPUT          ;DPM001
005550  022737  000131  002664          CMP     #'Y,ANSWER      ;WAS A <Y> INPUTED?                         ;DPM001
005556  001156                          BNE     13$             ;BRANCH OUT OF ROUTINE IF NOT               ;DPM001
005560  005037  002414                  CLR     QTYBRD          ;CLEAR DEVICE COUNT
005564  012700  001474          2$:     MOV     #$DDW0,R0       ;MOVE ADDRESS OF 1ST DDW TO R0
005570  012701  000020                  MOV     #16.,R1         ;CLEAR 16 WORDS
005574  005020                  3$:     CLR     (R0)+           ;CLEAR THE WORD
005576  005301                          DEC     R1              ;DECREMENT THE LOOP COUNTER AND
005600  001375                          BNE     3$              ;BRANCH BACK IF NOT DONE YET
005602  005037  001466                  CLR     $DEVM           ;CLEAR DEVICE MAP
005606  012737  000001  006122          MOV     #1,BDNUMB       ;START WITH 1 BOARD                         ;DPM001
005614  012701  001474                  MOV     #$DDW0,R1       ;LOAD DEVICE DESC ADDRESS                   ;DPM001
005620  012702  002456                  MOV     #VECADR,R2      ;POINT R2 TO UNIT #1 VECTOR ADRS LOC        ;DPM001
005624  012737  000067  002704          MOV     #'7,LRGSTC      ;MOVE ASCII 7 TO LRGSTC FOR OCTAL INPUT     ;DPM001
005632  104401  006124          4$:     TYPE    ,INDEV          ;TYPE:  'INPUT DEVICE '                     ;DPM001
005636  013746  006122                  MOV     BDNUMB,-(SP)    ;SAVE BDNUMB FOR TYPEOUT                    ;DPM001
005642  104403                          TYPOS                   ;TYPE THE BOARD NUMBER                      ;DPM001
005644     002                          .BYTE   2               ;TYPE 2 DIGITS                              ;DPM001
005645     000                          .BYTE   0               ;SUPPRESS LEADING ZEROS                     ;DPM001
005646  104401  006142                  TYPE    ,STAD           ;TYPE:  'STARTING ADDRESS                   ;DPM001
                                                                ;       (<CR> IF NO MORE)'                  ;DPM001
005652  012703  000006                  MOV     #6,R3           ;EXPECT 6 DIGITS                            ;DPM001
005656  004737  002722                  JSR     PC,READ         ;GO READ THE USER'S INPUT                   ;DPM001
005662  005704                          TST     R4              ;SEE IF AN INPUT WAS MADE                   ;DPM001
005664  001505                          BEQ     12$             ;BRANCH OUT IF NO MORE                      ;DPM001
005666  022704  160010                  CMP     #160010,R4      ;SEE IF ADDRESS IS TOO LOW                  ;DPM001
```

```
4152 005672  103003                      BHIS   5$          ;BRANCH TO REPORT ERROR INPUT IF SO      ;DPM001
4153 005674  022704  172770              CMP    #172770,R4  ;SEE IF ADDRESS IS TOO HIGH              ;DPM001
4154 005700  103003                      BHIS   6$          ;BRANCH IF OK                            ;DPM001
4155 005702  104401  006255      5$:     TYPE   ,OUTRAN     ;TYPE:  'ADDRESS OUT OF 160010-172770    ;DPM001
4156                                                        ;       RANGE - TRY AGAIN'               ;DPM001
4157 005706  000751                      BR     4$          ;GO DECREMENT BDNUM AND TRY AGAIN        ;DPM001
4158 005710  032704  000007      6$:     BIT    #7,R4       ;SEE THAT ADDRESS IS A START ADDRESS     ;DPM001
4159 005714  001425                      BEQ    9$          ;BRANCH IF OK                            ;DPM001
4160 005716  104401  006335              TYPE   ,NOTST      ;TYPE:  'NOT A STARTING ADDRESS -        ;DPM001
4161                                                        ;       TRY AGAIN'                       ;DPM001
4162 005722  000743                      BR     4$          ;GO DECREMENT BDNUM AND TRY AGAIN        ;DPM001
4163 005724  022626              7$:     CMP    (SP)+,(SP)+ ;CORRECT STACK AFTER TIMEOUT
4164 005726  132737  000001  001430      BITB   #BIT0,$ENV   ;SEE IF WE ARE UNDER APT                 ;DPM001
4165 005734  001004                      BNE    75$         ;BRANCH OUT IF NOT                       ;DPM001
4166 005736  032777  010000  173374      BIT    #BIT12,@SWR ;SEE IF UNDER MANUFACTURING ENVIRONMENT  ;DPM001
4167 005744  001406                      BEQ    8$          ;BRANCH IF NOT                           ;DPM001
4168 005746  062700  000010      75$:    ADD    #10,R0      ;MOVE TO NEXT MODULE ADDRESS             ;DPM001
4169 005752  022700  173004              CMP    #173004,R0  ;HAVE WE LOOKED OUT THE ENTIRE WINDOW?   ;DPM001
4170 005756  001007                      BNE    10$         ;BRANCH IF NOT - MORE DAYDREAMING        ;DPM001
4171 005760  000447                      BR     12$         ;BRANCH TO EXIT - BOSS JUST CAME IN      ;DPM001
4172 005762  104401  006207      8$:     TYPE   ,NOBD       ;TYPE:  'CAN'T SEE THE DR11 BOARD -      ;DPM001
4173                                                        ;       TRY AGAIN'                       ;DPM001
4174 005766  000721                      BR     4$          ;BRANCH BACK FOR NEW INPUT               ;DPM001
4175 005770  010400              9$:     MOV    R4,R0       ;MOVE CHECKED ADDRESS TO R0              ;DPM001
4176 005772  062700  000004              ADD    #4,R0       ;MAKE R0 ADDRESS THE CSR                 ;DPM001
4177 005776  005010              10$:    CLR    (R0)        ;MAKE SURE IT IS THERE                   ;DPM001
4178 006000  010146                      MOV    R1,-(SP)    ;SAVE PRESENT VALUE OF R1 ON THE STACK   ;DPM001
4179 006002  004737  004232              JSR    PC,DRGET    ;GO EXTRACT INFO FROM THE DR11
4180 006006  020126                      CMP    R1,(SP)+    ;WAS THE LOCATION A DR11?                ;DPM001
4181 006010  001010                      BNE    11$         ;BRANCH TO BOARD INCREMENT IF SO         ;DPM001
4182 006012  132737  000001  001430      BITB   #BIT0,$ENV  ;SEE IF WE ARE UNDER APT                 ;DPM001
4183 006020  001022                      BNE    110$        ;BRANCH OUT IF SO                        ;DPM001
4184 006022  032777  010000  173310      BIT    #BIT12,@SWR ;IS THIS A MANUFACTURING CONFIGURATION?  ;DPM001
4185 006030  001754                      BEQ    8$          ;GO CALL ERROR IF NOT                    ;DPM001
4186 006032  022737  000020  006122  11$:CMP    #16.,BDNUMB ;SEE IF THE TABLE IS FULL                ;DPM001
4187 006040  001417                      BEQ    12$         ;EXIT IF SO                              ;DPM001
4188 006042  005237  006122              INC    BDNUMB      ;INCREMENT BOARD NUMBER                  ;DPM001
4189 006046  132737  000001  001430      BITB   #BIT0,$ENV  ;SEE IF WE ARE UNDER APT                 ;DPM001
4190 006054  001004                      BNE    110$        ;BRANCH OUT IF SO                        ;DPM001
4191 006056  032777  010000  173254      BIT    #BIT12,@SWR ;SEE IF UNDER MANUFACTURING ENVIRONMENT  ;DPM001
4192 006064  001662                      BEQ    4$          ;BRANCH IF NOT                           ;DPM001
4193 006066  062700  000010      110$:   ADD    #10,R0      ;MOVE TO NEXT MODULE ADDRESS             ;DPM001
4194 006072  022700  173004              CMP    #173004,R0  ;HAVE WE LOOKED OUT THE ENTIRE WINDOW?   ;DPM001
4195 006076  001337                      BNE    10$         ;BRANCH IF NOT - MORE DAYDREAMING        ;DPM001
4196 006100  013737  002674  000006  12$:MOV    OLDPS1,TMOPSW;RESTORE TIMEOUT PS
4197 006106  013737  002672  000004      MOV    OLDPC1,TOVECT;RESTORE TIMEOUT VECTOR
4198 006114  004737  005142      13$:    JSR    PC,TYPCNF   ;GO TYPE THE BOARD CONFIGURATIONS
4199 006120  000207                      RTS    PC          ;EXIT
4200
4201 006122  000000          BDNUMB: .WORD  0               ;LOCATION TO HOLD BOARD NUMBER           ;DPM001
4202 006124     111  116  120 INDEV:  .ASCIZ !INPUT DEVICE !                                         ;DPM001
4203 006142     040  123  124 STAD:   .ASCIZ ! STARTING ADDRESS (<CR> IF NO MORE) !                  ;DPM001
4204 006207     103  101  116 NOBD:   .ASCIZ !CAN'T SEE THE DR11 BOARD - TRY AGAIN!<CRLF>          ;DPM001
4205 006255     101  104  104 OUTRAN: .ASCIZ !ADDRESS OUT OF 160010-172770 RANGE - TRY AGAIN!<CRLF> ;DPM001
4206 006335     116  117  124 NOTST:  .ASCIZ !NOT A STARTING ADDRESS - TRY AGAIN!<CRLF>             ;DPM001
4207 006401     104  117  040 DYWTLT: .ASCIZ !DO YOU WISH TO CHANGE THE CONFIGURATION (Y OR <CR>) ! ;DPM001
4208                           .EVEN
```

```
                                        .SBTTL  ROUTINE TO REPORT UNEXPECTED OR ERRONEOUS TRAPS OR INTERRUPTS
                                ;;**************************************************************
                                ;*
                                ;*      THIS IS THE ROUTINE TO REPORT UNEXPECTED OR ERPONEOUS TRAPS OR INTERRUPTS
                                ;*
                                ;;**************************************************************
006466  012737  000340  177776  CATCH:  MOV     #LEVEL7,PSW     ;REESTABLISH CPU PRIORITY AT 7
006474  012637  002542                  MOV     (SP)+,BDVECT    ;GET ADDRESS OF TRAP VECTOR + 4
006500  012637  002674                  MOV     (SP)+,OLDPS1    ;SAVE PS
006504  012637  002676                  MOV     (SP)+,OLDPC2    ;SAVE PC OF ADDRESS OF INSTRUCTION CAUSING TRAP
006510  012637  002700                  MOV     (SP)+,OLDPS2    ;SAVE 2ND PS
006514  162737  000004  002542          SUB     #4,BDVECT       ;ADJUST TO POINT TO TRAP ADDRESS
006522  104050                          ERROR   +50             ;UNEXPECTED TRAP OR INTERRUPT TO TRAP ADDRESS BELOW
006524  013746  002700                  MOV     OLDPS2,-(SP)    ;RESTORE PS RETURN ON STACK
006530  013746  002676                  MOV     OLDPC2,-(SP)    ;RESTORE PC RETURN ON STACK
006534  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.
                                ;*
                                ;;**********************************************************************
006536  005046                  PSTATE: CLR     -(SP)           ;SHOW STATE AS ZERO INITIALLY
006540  033737  002710  002720          BIT     BITTST,DDW      ;CHECK STATE OF BIT IN DDW USING BIT SET IN BITTST
006546  001401                          BEQ     1$              ;BRANCH IF NOT SET
006550  005216                          INC     (SP)            ;SHOW A '1' STATE FOR THAT BIT
006552  104403                  1$:     TYPOS                   ;TYPE THE STATE, LEADING ZEROS SUPPRESSED
006554     001      000                 .BYTE   1,0             ;TYPE 1 CHARACTER, SUPPRESS LEADING ZEROS
006556  000207                          RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO PRINT DEVICE PRIORITY
                                ;;**********************************************************************
                                ;*
                                ;*      THIS SUBROUTINE PRINTS THE DEVICE PRIORITY
                                ;*
                                ;**********************************************************************
006560  013746  002720          PNTPRI: MOV     DDW,-(SP)       ;PUT DEVICE DESCRIPTOR WORD ON STACK
006564  006216                          ASR     (SP)            ;SHIFT RIGHT STACK LOCATION 5 PLACES
006566  006216                          ASR     (SP)
006570  006216                          ASR     (SP)
006572  006216                          ASR     (SP)
006574  006216                          ASR     (SP)
006576  042716  177770                  BIC     #177770,(SP)    ;MASK TO GET PRIORITY
006602  104403                          TYPOS                   ;TYPE THE DEVICE PRIORITY
006604     001     000                  .BYTE   1,0             ;TYPE 1 CHARACTER, SUPPRESS LEADING ZEROS
006606  000207                          RTS     PC              ;EXIT
```

```
                                        .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
                                ;*
                                ;;**************************************************************
006610  011637  001360          SETTUP: MOV     (SP),$TMP0      ;SAVE RETURN ADDRESS
                                .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
006614  012706  001300                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
006620  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
006622  022706  001340                  CMP     #SWR,R6 ;;DONE?
006626  001374                          BNE     .-6             ;;LOOP BACK IF NO
006630  012706  001300                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
006634  012737  031200  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
006642  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
006650  012737  032656  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
006656  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
006664  012737  031112  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
006672  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
006700  012737  034074  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
006706  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
006714  013737  026102  026074          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
006722  005037  001376                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
006726  112737  000001  001315          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
006734  012737  006734  001306          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
006742  012737  006742  001310          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
006750  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
006754  012737  007010  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
006762  012737  177570  001340          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
006770  012737  177570  001342          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
006776  022777  177777  172334          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
007004  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
007006  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
007010  012716  007016          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
007014  000002                          RTI
007016  012737  000176  001340  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
007024  012737  000174  001342          MOV     #DISPREG,DISPLAY
007032  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
007036  005037  001416                  CLR     $PASS           ;;CLEAR PASS COUNT
007042  132737  000200  001431          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
007050  001403                          BEQ     67$             ;;YES,USE NON-APT SWITCH
007052  012737  001432  001340          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
007060                          67$:
007060  013746  001360                  MOV     $TMP0,-(SP)     ;PUT RETURN ADDRESS ON STACK AND
007064  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.
                                ;*
                                ;;**************************************************************
007066  005737  002712          TSTMM:  TST     MEMGMT          ;TEST TO SEE IF MEMORY MANAGEMENT EXISTS
007072  001440                          BEQ     4$              ;BRANCH IF NOT
007074  012737  000401  007104          MOV     #CY+GO,1$       ;SET UP BIT TEST DATA
007102  040227                          BIC     R2,(PC)+        ;TEST TO SEE IF BOTH THE CYCLE AND GO BITS ARE SET
007104  000401                  1$:     .WORD   CY+GO           ;LOCATION TO STORE THE CYCLE AND GO BITS
007106  001032                          BNE     4$              ;KICK OUT IF CYCLE AND/OR GO ARE CLEAR
007110  032737  000060  002572          BIT     #X6+X7,ECSR     ;SEE IF XBA16 OR XBA17 WERE SET IN EXPECTED DATA
007116  001426                          BEQ     4$              ;BRANCH OUT IF NOT
007120  032737  000040  002572          BIT     #X7,ECSR        ;SEE IF XBA17 IS SET
007126  001005                          BNE     2$              ;GO CHECK STATUS OF XBA16 IF SET
007130  132737  000001  002713          BITB    #BIT0,MEMGMT+1  ;SEE IF 200000+NOCARE WAS FOUND TO EXIST - IF NOT,
007136  001420                          BEQ     5$              ;GO SET EXPECTED ERROR AND NEX BITS AND CHECK FOR ERROR
007140  000415                          BR      4$              ;BRANCH OUT IF LOCATION EXISTS
007142  032737  000020  002572  2$:     BIT     #X6,ECSR        ;SEE IF XBA16 IS SET
007150  001005                          BNE     3$              ;BRANCH TO CHECK 600000+NOCARE IF SET
007152  132737  000002  002713          BITB    #BIT1,MEMGMT+1  ;SEE IF 400000+NOCARE WAS FOUND TO EXIST - IF NOT,
007160  001407                          BEQ     5$              ;GO SET EXPECTED ERROR AND NEX BITS AND CHECK FOR ERROR
007162  000404                          BR      4$              ;BRANCH OUT IF LOCATION EXISTS
007164  132737  000004  002713  3$:     BITB    #BIT2,MEMGMT+1  ;SEE IF 600000+NOCARE WAS FOUND TO EXIST - IF NOT,
007172  001402                          BEQ     5$              ;GO SET EXPECTED ERROR AND NEX BITS AND CHECK FOR ERROR
007174  062716  000002          4$:     ADD     #2,(SP)         ;CORRECT PC RETURN
007200  000207                  5$:     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.
                                        ;*
                                        ;;**************************************************************

000002                                          .RADIX  2                       ;THIS ENABLES YOU TO SEE THE PATTERNS IN BINARY
007202  177777                        PATRNS: .WORD   1111111111111111        ;ALL SET BITS
007204  000000                                .WORD   0000000000000000        ;ALL CLEAR BITS
007206  052525                                .WORD   0101010101010101        ;EVEN BITS SET, ODD BITS CLEAR
007210  125252                                .WORD   1010101010101010        ;ODD BITS SET, EVEN BITS CLEAR
007212  031463                                .WORD   0011001100110011        ;PAIRS OF BITS SET
007214  007417                                .WORD   0000111100001111        ;GROUPS OF 4 BITS SET
007216  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.
                                ;*
                                ;;**************************************************************

007220                          MAICLR: ; X=     0    1    2    3    4    5    6    7
007220  000200  000000  001202          .WORD   0200,0000,1202,1002,122204,122204,123206,123206  ;CSR 00000X EXPECTED
007240  004210  004010  005212          .WORD   4210,4010,5212,5012,126214,126214,127216,127216  ;CSR 00001X EXPECTED
007260  000220  000020  001222          .WORD   0220,0020,1222,1022,122224,122224,123226,123226  ;CSR 00002X EXPECTED
007300  004230  004030  005232          .WORD   4230,4030,5232,5032,126234,126234,127236,127236  ;CSR 00003X EXPECTED
007320  000240  000040  001242          .WORD   0240,0040,1242,1042,122244,122244,123246,123246  ;CSR 00004X EXPECTED
007340  004250  004050  005252          .WORD   4250,4050,5252,5052,126254,126254,127256,127256  ;CSR 00005X EXPECTED
007360  000260  000060  001262          .WORD   0260,0060,1262,1062,122264,122264,123266,123266  ;CSR 00006X EXPECTED
007400  004270  004070  005272          .WORD   4270,4070,5272,5072,126274,126274,127276,127276  ;CSR 00007X EXPECTED
007420  000300  000100  001302          .WORD   0300,0100,1302,1102,122304,022104,123306,023106  ;CSR 00010X EXPECTED
007440  004310  004110  005312          .WORD   4310,4110,5312,5112,126314,026114,127316,027116  ;CSR 00011X EXPECTED
007460  000320  000120  001322          .WORD   0320,0120,1322,1122,122324,022124,123326,023126  ;CSR 00012X EXPECTED
007500  004330  004130  005332          .WORD   4330,4130,5332,5132,126334,026134,127336,027136  ;CSR 00013X EXPECTED
007520  000340  000140  001342          .WORD   0340,0140,1342,1142,122344,022144,123346,023146  ;CSR 00014X EXPECTED
007540  004350  004150  005352          .WORD   4350,4150,5352,5152,126354,026154,127356,027156  ;CSR 00015X EXPECTED
007560  000360  000160  001362          .WORD   0360,0160,1362,1162,122364,022164,123366,023166  ;CSR 00016X EXPECTED
007600  004370  004170  005372          .WORD   4370,4170,5372,5172,126374,026174,127376,027176  ;CSR 00017X EXPECTED
007620  000600  000200  001602          .WORD   0600,0200,1602,1202,122604,122604,123606,123606  ;CSR 00040X EXPECTED
007640  004610  004210  005612          .WORD   4610,4210,5612,5212,126614,126614,127616,127616  ;CSR 00041X EXPECTED
007660  000620  000220  001622          .WORD   0620,0220,1622,1222,122624,122624,123626,123626  ;CSR 00042X EXPECTED
007700  004630  004230  005632          .WORD   4630,4230,5632,5232,126634,126634,127636,127636  ;CSR 00043X EXPECTED
007720  000640  000240  001642          .WORD   0640,0240,1642,1242,122644,122644,123646,123646  ;CSR 00044X EXPECTED
007740  004650  004250  005652          .WORD   4650,4250,5652,5252,126654,126654,127656,127656  ;CSR 00045X EXPECTED
007760  000660  000260  001662          .WORD   0660,0260,1662,1262,122664,122664,123666,123666  ;CSR 00046X EXPECTED
010000  004670  004270  005672          .WORD   4670,4270,5672,5272,126674,126674,127676,127676  ;CSR 00047X EXPECTED
010020  000700  000300  001702          .WORD   0700,0300,1702,1302,122704,022304,123706,023306  ;CSR 00050X EXPECTED
010040  004710  004310  005712          .WORD   4710,4310,5712,5312,126714,026314,127716,027316  ;CSR 00051X EXPECTED
010060  000720  000320  001722          .WORD   0720,0320,1722,1322,122724,022324,123726,023326  ;CSR 00052X EXPECTED
010100  004730  004330  005732          .WORD   4730,4330,5732,5332,126734,026334,127736,027336  ;CSR 00053X EXPECTED
010120  000740  000340  001742          .WORD   0740,0340,1742,1342,122744,022344,123746,023346  ;CSR 00054X EXPECTED
010140  004750  004350  005752          .WORD   4750,4350,5752,5352,126754,026354,127756,027356  ;CSR 00055X EXPECTED
010160  000760  000360  001762          .WORD   0760,0360,1762,1362,122764,022364,123766,023366  ;CSR 00056X EXPECTED
010200  004770  004370  005772          .WORD   4770,4370,5772,5372,126774,026374,127776,027376  ;CSR 00057X EXPECTED
```

EXPECTED DATA TABLE FOR CSR CHECK TEST 3

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

                                                ; X=      0     1     2     3     4     5     6     7
010220  010200  010000  011202  MAISET:       .WORD   10200,10000,11202,11002,12204,12004,13206,13006  ;CSR 01000X EXPECTED
010240  014210  014010  015212                .WORD   14210,14010,15212,15012,16214,16014,17216,17016  ;CSR 01001X EXPECTED
010260  010220  010020  011222                .WORD   10220,10020,11222,11022,12224,12024,13226,13026  ;CSR 01002X EXPECTED
010300  014230  014030  015232                .WORD   14230,14030,15232,15032,16234,16034,17236,17036  ;CSR 01003X EXPECTED
010320  010240  010040  011242                .WORD   10240,10040,11242,11042,12244,12044,13246,13046  ;CSR 01004X EXPECTED
010340  014250  014050  015252                .WORD   14250,14050,15252,15052,16254,16054,17256,17056  ;CSR 01005X EXPECTED
010360  010260  010060  011262                .WORD   10260,10060,11262,11062,12264,12064,13266,13066  ;CSR 01006X EXPECTED
010400  014270  014070  015272                .WORD   14270,14070,15272,15072,16274,16074,17276,17076  ;CSR 01007X EXPECTED
010420  010300  010100  011302                .WORD   10300,10100,11302,11102,12304,12104,13306,13106  ;CSR 01010X EXPECTED
010440  014310  014110  015312                .WORD   14310,14110,15312,15112,16314,16114,17316,17116  ;CSR 01011X EXPECTED
010460  010320  010120  011322                .WORD   10320,10120,11322,11122,12324,12124,13326,13126  ;CSR 01012X EXPECTED
010500  014330  014130  015332                .WORD   14330,14130,15332,15132,16334,16134,17336,17136  ;CSR 01013X EXPECTED
010520  010340  010140  011342                .WORD   10340,10140,11342,11142,12344,12144,13346,13146  ;CSR 01014X EXPECTED
010540  014350  014150  015352                .WORD   14350,14150,15352,15152,16354,16154,17356,17156  ;CSR 01015X EXPECTED
010560  010360  010160  011362                .WORD   10360,10160,11362,11162,12364,12164,13366,13166  ;CSR 01016X EXPECTED
010600  014370  014170  015372                .WORD   14370,14170,15372,15172,16374,16174,17376,17176  ;CSR 01017X EXPECTED
010620  010600  011202  011602                .WORD   10600,11202,11602,12204,12604,13206,13606,14210  ;CSR 01040X EXPECTED
010640  014610  015212  015612                .WORD   14610,15212,15612,16214,16614,17216,17616,10200  ;CSR 01041X EXPECTED
010660  010620  011222  011622                .WORD   10620,11222,11622,12224,12624,13226,13626,14230  ;CSR 01042X EXPECTED
010700  014630  015232  015632                .WORD   14630,15232,15632,16234,16634,17236,17636,10220  ;CSR 01043X EXPECTED
010720  010640  011242  011642                .WORD   10640,11242,11642,12244,12644,13246,13646,14250  ;CSR 01044X EXPECTED
010740  014650  015252  015652                .WORD   14650,15252,15652,16254,16654,17256,17656,10240  ;CSR 01045X EXPECTED
010760  010660  011262  011662                .WORD   10660,11262,11662,12264,12664,13266,13666,14270  ;CSR 01046X EXPECTED
011000  014670  015272  015672                .WORD   14670,15272,15672,16274,16674,17276,17676,10260  ;CSR 01047X EXPECTED
011020  010700  011302  011702                .WORD   10700,11302,11702,12304,12704,13306,13706,14310  ;CSR 01050X EXPECTED
011040  014710  015312  015712                .WORD   14710,15312,15712,16314,16714,17316,17716,10300  ;CSR 01051X EXPECTED
011060  010720  011322  011722                .WORD   10720,11322,11722,12324,12724,13326,13726,14330  ;CSR 01052X EXPECTED
011100  014730  015332  015732                .WORD   14730,15332,15732,16334,16734,17336,17736,10320  ;CSR 01053X EXPECTED
011120  010740  011342  011742                .WORD   10740,11342,11742,12344,12744,13346,13746,14350  ;CSR 01054X EXPECTED
011140  014750  015352  015752                .WORD   14750,15352,15752,16354,16754,17356,17756,10340  ;CSR 01055X EXPECTED
011160  010760  011362  011762                .WORD   10760,11362,11762,12364,12764,13366,13766,14370  ;CSR 01056X EXPECTED
011200  014770  015372  015772                .WORD   14770,15372,15772,16374,16774,17376,17776,10360  ;CSR 01057X EXPECTED
```

FUTURE PATCH AREA

```
                                        .SBTTL  FUTURE PATCH AREA
                                ;;**************************************************************
                                ;*
                                ;*      THIS AREA IS RESERVED FOR PERMANENT PATCHES WHICH MAY BE INITIATED IN
                                ;*      THE FUTURE.
                                ;*
                                ;;**************************** ***********************************
011220  000000  000000  000000  PATCHS: .WORD   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
011320  000000  000000  000000          .WORD   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

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

011420  005037  002670          START1: CLR     MANSIZE         ;CLEAR THE MANSIZE SO WE WILL AUTOSIZE
011424  005037  001416          START:  CLR     $PASS           ;CLEAR $PASS
011430  005037  026010                  CLR     PREOP           ;CLEAR THE EOP PRINTING FLAG            ;DPM001
011434  005037  002716                  CLR     $PASS2          ;CLEAR $PASS2
011440  005037  001412                  CLR     $FATAL          ;CLEAR ERROR NO.
011444  005037  001410                  CLR     $MSGTYP         ;CLEAR MESSAGE TYPE
011450  005037  001414                  CLR     $TESTN          ;CLEAR TEST NO.
011454  005037  001420                  CLR     $DEVCT          ;CLEAR DEVICE COUNT
011460  005037  001422                  CLR     $UNIT           ;CLEAR UNIT NUMBER
011464  012737  000006  000004          MOV     #TOVECT+2,TOVECT;INITIALIZE TIMEOUT VECTORS TO 6
011472  012737  000003  000006          MOV     #BPT,TMOPSW     ;CATCHER ROUTINE
011500  012706  001300                  MOV     #STACK,SP       ;INITIALIZE THE STACK
011504  004737  006610                  JSR     PC,SETTUP       ;GO TO THE SETUP ROUTINE TO INITIALIZE VECTORS
011510  005037  001422                  CLR     $UNIT           ;CLEAR $UNIT
011514  005037  001420                  CLR     $DEVCT          ;CLEAR $DEVCT
011520  005037  001414                  CLR     $TESTN          ;CLEAR $TESTN
011524  005037  002706                  CLR     EOPLOC          ;CLEAR EOPLOC
011530  132737  000001  001430          BITB    #BIT0,$ENV      ;CHECK IF ON APT
011536  001404                          BEQ     1$              ;BR IF NOT APT
011540  132737  000200  001431          BITB    #BIT7,$ENVM     ;DID APT SIZE
011546  001007                          BNE     2$              ;BR, IF APT SIZED
011550  022737  177777  002670  1$:     CMP     #-1,MANSIZE     ;WAS CONFIGURATION SET UP IN MULT. BOARD ROUTINE?
011556  001446                          BEQ     BEGIN           ;IF YES, SKIP SELF-SIZING
011560  004737  005416                  JSR     PC,ASIZE        ;AUTOMATICALLY SIZE FOR BOARD CONFIGURATION
011564  000443                          BR      BEGIN           ;BRANCH
011566  005037  002414          2$:     CLR     QTYBRD          ;CLEAR  DEVICE CNT
011572  013702  001466                  MOV     $DEVM,R2        ;MOVE DEVICE MAP TO R2
011576  005702                  3$:     TST     R2              ;TEST MSB OF DEVICE MAP
011600  100002                          BPL     4$              ;BR, IF MSB IS ZERO
011602  005237  002414                  INC     QTYBRD          ;INCREMENT DEVICE COUNT, IF MSB=1
011606  000241                  4$:     CLC                     ;CLEAR THE CARRY BIT FOR THE ROL
011610  006102                          ROL     R2              ;SHIFT NEXT BIT INTO MSB POSITION
011612  001371                          BNE     3$              ;CONTINUE CHECKING $DEVM, IF MORE BITS SET
011614  012702  002416                  MOV     #REGADR,R2      ;POINT R2 TO THE DEVICE ADDRESS TABLE
011620  013700  001464                  MOV     $BASE,R0        ;LOAD BASE DEVICE ADDRESS IN R0
011624  012701  000020                  MOV     #16.,R1         ;MOVE LOOP COUNTER TO R1
011630  010022                  5$:     MOV     R0,(R2)+        ;MOVE DEVICE ADDRESS TO TABLE
011632  062700  000010                  ADD     #10,R0          ;POINT R0 TO NEXT DEVICE ADDRESS
011636  005301                          DEC     R1              ;DECREMENT THE LOOP COUNTER AND
011640  001373                          BNE     5$              ;BRANCH IF NOT ALL DONE YET
011642  012702  002456                  MOV     #VECADR,R2      ;GET LOCATION OF VECTOR TABLE
011646  013700  001460                  MOV     $VECT1,R0       ;COPY BASE VECTOR
011652  042700  177400                  BIC     #177400,R0      ;CLEAR UPPER BYTE
011656  012701  000020                  MOV     #16.,R1         ;DO 16 VECTORS
011662  010022                  6$:     MOV     R0,(R2)+        ;PUT VECTOR ADDRESS IN TABLE
011664  062700  000010                  ADD     #10,R0          ;POINT R0 TO NEXT VECTOR ADDRESS
011670  005301                          DEC     R1              ;DECREMENT LOOP COUNTER
011672  001373                          BNE     6$              ;BRANCH IF NOT ALL DONE YET
```

```
4453                                                    .SBTTL  DETERMINE MEM MGMT AND UPPER MEMORY EXISTENCE
4454 011674  005737  001416          BEGIN:  TST     $PASS           ;SEE IF THIS IS THE FIRST PASS
4455 011700  001003                          BNE     1$              ;BRANCH IF NOT
4456 011702  005737  002716                  TST     $PASS2          ;SEE IF UPPER LOCATION HAS BEEN SET
4457 011706  001402                          BEQ     2$              ;BRANCH IF SO
4458 011710  000137  012356          1$:     JMP     BEGIN1          ;JUMP TO BEGIN1
4459 011714  005037  002712          2$:     CLR     MEMGMT          ;CLEAR THE MEMORY MANAGEMENT FLAG
4460 011720  013737  000004  002672          MOV     TOVECT,OLDPC1   ;SAVE TIMEOUT VECTOR
4461 011726  012737  012320  000004          MOV     #13$,TOVECT     ;TIMEOUT VECTOR TO 13$
4462 011734  013737  000006  002674          MOV     TMOPSW,OLDPS1   ;SAVE TIMEOUT PS
4463 011742  012737  000340  000006          MOV     #LEVEL7,TMOPSW  ;PS TIMEOUT TO PRIORITY 7
4464 011750  005037  177572                  CLR     MMR0            ;TEST FOR MEMORY MANAGEMENT & TURN IT OFF
4465 011754  105237  002712                  INCB    MEMGMT          ;INCREMENT FLAG SHOWING MEMORY MANAGEMENT EXISTS
4466 011760  012700  172300                  MOV     #KIPDR0,R0      ;MOVE ADDRESS OF 1ST MM REGISTER TO R0
4467 011764  012701  000020                  MOV     #16.,R1         ;DO ALL 16 PDR'S
4468 011770  012737  012010  000004          MOV     #4$,TOVECT      ;TIMEOUTS TO 4$
4469 011776  012720  077406          3$:     MOV     #77406,(R0)+    ;ENABLE THE PDR PAGE
4470 012002  005301                          DEC     R1              ;DECREMENT LOOP COUNTER AND
4471 012004  001374                          BNE     3$              ;BRANCH BACK IF NOT DONE
4472 012006  000403                          BR      5$              ;BRANCH OVER STACK CORRECTION
4473 012010  022626                  4$:     CMP     (SP)+,(SP)+     ;CLEAN STACK AFTER INTERRUPT
4474 012012  012700  172340                  MOV     #KIPAR0,R0      ;POINT R0 TO THE 1ST PAR
4475 012016  012703  000002          5$:     MOV     #2,R3           ;DO 2 SETS OF PAR'S
4476 012022  012737  012062  000004          MOV     #8$,4           ;TIMEOUTS TO 8$
4477 012030  005002                  6$:     CLR     R2              ;START WITH LOAD OF ZERO
4478 012032  012701  000007                  MOV     #7,R1           ;DO 7 REGISTERS
4479 012036  010220                  7$:     MOV     R2,(R0)+        ;LOAD THE PAR
4480 012040  062702  000200                  ADD     #200,R2         ;PUT R2 AT NEXT VALUE
4481 012044  005301                          DEC     R1              ;DECREMENT AND
4482 012046  001373                          BNE     7$              ;BRANCH BACK IF NOT DONE
4483 012050  012720  177600                  MOV     #177600,(R0)+   ;SETUP PAR7
4484 012054  005303                          DEC     R3              ;DECREMENT AND
4485 012056  001364                          BNE     6$              ;BRANCH BACK IF KDPAR'S NOT LOADED
4486 012060  000401                          BR      9$              ;BRANCH OVER STACK CORRECTION
4487 012062  022626                  8$:     CMP     (SP)+,(SP)+     ;CLEAN STACK AFTER TIMEOUT
4488 012064  012737  012232  000004  9$:     MOV     #10$,TOVECT     ;TIMEOUT VECTOR TO 10$                  ;DPM001
4489 012072  013737  000250  002676          MOV     MMVECT,OLDPC2   ;SAVE MEMORY MANAGEMENT VECTOR
4490 012100  012737  012232  000250          MOV     #10$,MMVECT     ;MEMORY MANAGEMENT VECTOR TO 10$
4491 012106  013737  000252  002700          MOV     MMPS,OLDPS2     ;SAVE MEMORY MANAGEMENT PS
4492 012114  012737  000340  000252          MOV     #LEVEL7,MMPS    ;MEMORY MANAGEMENT PS TO PRIORITY 7
4493 012122  005237  177572                  INC     MMR0            ;TURN ON MEMORY MANAGEMENT
4494 012126  012737  002400  172344          MOV     #2400,KIPAR2    ;SET UP KIPAR2 TO ACCESS LOCATION 240000+BITS 12-0 OF NOCARE
4495 012134  012737  002400  172364          MOV     #2400,KDPAR2    ;SET UP KDPAR2 TO ACCESS LOCATION 240000+BITS 12-0 OF NOCARE
4496 012142  005737  056104                  TST     NOCARE          ;SEE IF BITS 12-0 OF NOCARE ADRS +240000 EXISTS
4497 012146  152737  000001  002713          BISB    #BIT0,MEMGMT+1  ;SET BIT 0 OF UPPER BYTE OF MEMGMT IF IT DOES
4498 012154  012737  004400  172344          MOV     #4400,KIPAR2    ;SET UP KIPAR2 TO ACCESS LOCATION 440000+BITS 12-0 OF NOCARE
4499 012162  012737  004400  172364          MOV     #4400,KDPAR2    ;SET UP KDPAR2 TO ACCESS LOCATION 440000+BITS 12-0 OF NOCARE
4500 012170  005737  056104                  TST     NOCARE          ;SEE IF BITS 12-0 OF NOCARE ADRS +440000 EXISTS
4501 012174  152737  000002  002713          BISB    #BIT1,MEMGMT+1  ;SET BIT 1 OF UPPER BYTE OF MEMGMT IF IT DOES
4502 012202  012737  006400  172344          MOV     #6400,KIPAR2    ;SET UP KIPAR2 TO ACCESS LOCATION 640000+BITS 12-0 OF NOCARE
4503 012210  012737  006400  172364          MOV     #6400,KDPAR2    ;SET UP KDPAR2 TO ACCESS LOCATION 640000+BITS 12-0 OF NOCARE
4504 012216  005737  056104                  TST     NOCARE          ;SEE IF BITS 12-0 OF NOCARE ADRS +640000 EXISTS
4505 012222  152737  000004  002713          BISB    #BIT2,MEMGMT+1  ;SET BIT 2 OF UPPER BYTE OF MEMGMT IF IT DOES
4506 012230  000407                          BR      12$             ;BRANCH OVER STACK CORRECTION
4507 012232  027627  000000          10$:    CMP     @0(SP),(PC)+    ;COMPARE NEXT INSTRUCTION WITH A
4508 012236  005737                          TST     @(PC)+          ;TEST MODE 3 INSTRUCTION (THIS IS NOT EXECUTED)
4509 012240  001402                          BEQ     11$             ;BRANCH TO RETURN IF SO
```

```
4510 012242  012716  012250                    MOV     #12$,(SP)       ;FUDGE RETURN TO EXIT IF NOT
4511 012246  000002                  11$:      RTI                     ;RETURN
4512 012250  005037  177572          12$:      CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4513 012254  012737  012300  000004            MOV     #125$,TOVECT    ;TIMEOUT TO 125$                        ;DPM001
4514 012262  012737  000400  172344            MOV     #400,KIPAR2     ;RESTORE PAGE 2                         ;DPM001
4515 012270  012737  000400  172364            MOV     #400,KDPAR2     ;RESTORE PAGE 2                         ;DPM001
4516 012276  000401                            BR      126$            ;BRANCH OVER STACK CLEANUP              ;DPM001
4517 012300  022626                  125$:     CMP     (SP)+,(SP)+     ;CLEANUP THE STACK                      ;DPM001
4518 012302  013737  002700  000252  126$:     MOV     OLDPS2,MMPS     ;RESTORE MEMORY MANAGEMENT PS
4519 012310  013737  002676  000250            MOV     OLDPC2,MMVECT   ;RESTORE MEMORY MANAGEMENT VECTOR
4520 012316  000402                            BR      14$             ;BRANCH OVER STACK CORRECTION
4521 012320  062706  000004          13$:      ADD     #4,SP           ;CORRECT STACK AFTER TIMEOUT
4522 012324  013737  002674  000006  14$:      MOV     OLDPS1,TMOPSW   ;RESTORE TIMEOUT PS
4523 012332  013737  002672  000004            MOV     OLDPC1,TOVECT   ;RESTORE TIMEOUT VECTOR
4524 012340  104401  041302                    TYPE    ,M1             ;TYPE:  '(^X) INHIBITS EOP'S, (^Y) FOR ERROR SUMMARY'
4525                                                                    ;       'UNIBUS HANG?  RESTART AT ADDRESS '
4526 012344  012746  056016                    MOV     #UBHANG,-(SP)   ;MOVE ADDRESS OF HANG ROUTINE TO STACK
4527 012350  104402                            TYPOC                   ;GO TYPE THE ADDRESS IN OCTAL
4528 012352  104401  041421                    TYPE    ,M1A            ;       'CZDRLDO DR11 GEN NPR INTFC LOGIC TEST'
4529                                                                    ;       'EOP MSG WILL PRINT EVERY       ;DPM001
4530                                                                    ;       20 SECONDS APPROXIMATELY'       ;DPM001
4531                                          .SBTTL  PREPARE ADDRESSES AND VECTORS FOR UUT
4532 012356  012737  000001  002544  BEGIN1: MOV     #BIT0,DEVMSK    ;SET UP BIT MASK TO TEST $DEVM FOR DEVICES
4533 012364  005037  002546                    CLR     TABINX          ;CLEAR LOCATION TO STORE TABLE OFFSETS
4534 012370  033737  002544  001466  TSTDEV: BIT     DEVMSK,$DEVM    ;CHECK TO SEE IF DEVICE IS TO BE TESTED
4535 012376  001026                            BNE     2$              ;BR, IF YES
4536 012400  005737  002544                    TST     DEVMSK          ;SEE IF BIT 15 IS SET
4537 012404  100013                            BPL     1$              ;BRANCH TO CONTINUE IF NOT SET
4538 012406  005737  001416                    TST     $PASS           ;SEE IF THIS IS THE FIRST PASS
4539 012412  001361                            BNE     BEGIN1          ;BRANCH TO REINITIALIZE THE DEVMSK LOCATION IF NOT
4540 012414  005737  002716                    TST     $PASS2          ;SEE IF THIS IS THE FIRST PASS
4541 012420  001356                            BNE     BEGIN1          ;BRANCH TO REINITIALIZE THE DEVMSK LOCATION IF NOT
4542 012422  104401  041151                    TYPE    ,NODVPR         ;TYPE:  'NO DEVICES RECOGNIZED - DIAGNOSTIC CANNOT BE RUN'
4543                                                                    ;       'RESTART AT 204 IF A DEVICE IS PRESENT'
4544 012426  000000                            HALT                    ;FATAL ERROR - HALT HERE
4545 012430  000137  011420                    JMP     START1          ;JUMP TO START1 TO CHECK AGAIN FOR A MODULE
4546 012434  006337  002544          1$:       ASL     DEVMSK          ;SHIFT MASK TO CHECK NEXT $DEVM BIT
4547 012440  062737  000002  002546            ADD     #2,TABINX       ;INCREMENT TABLE INDEX
4548 012446  005237  001422                    INC     $UNIT           ;INCREMENT UNIT NUMBER
4549 012452  000746                            BR      TSTDEV          ;GO TEST NEXT BIT OF DEVICE MAP
4550
4551 012454  006337  002544          2$:       ASL     DEVMSK          ;UPDATE DEVICE MAP TEST MASK
4552 012460  013702  002546                    MOV     TABINX,R2       ;MOVE TABLE OFFSET TO R2
4553 012464  062737  000002  002546            ADD     #2,TABINX       ;UPDATE TABLE OFFSET FOR NEXT DEVICE
4554 012472  016200  002416                    MOV     REGADR(R2),R0   ;PUT UUT ADDRESS INTO R0
4555 012476  012701  002516                    MOV     #WCR,R1         ;POINT R1 TO STORAGE AREA FOR UUT ADDRESSES
4556 012502  012703  000004                    MOV     #4,R3           ;MOVE 4 ADDRESSES
4557 012506  010021                  3$:       MOV     R0,(R1)+        ;TRANSFER UUT ADDRESS
4558 012510  062700  000002                    ADD     #2,R0           ;POINT TO NEXT UUT REGISTER
4559 012514  005303                            DEC     R3              ;DECREMENT THE LOOP COUNTER AND
4560 012516  001373                            BNE     3$              ;BRANCH IF NOT DONE TRANSFERING
4561 012520  016200  002456                    MOV     VECADR(R2),R0   ;PUT UUT VECTOR INTO R0
4562 012524  010021                            MOV     R0,(R1)+        ;TRANSFER UUT VECTORS TO ACTIVE TABLE AREA
4563 012526  062700  000002                    ADD     #2,R0           ;POINT TO NEXT VECTOR
4564 012532  010011                            MOV     R0,(R1)         ;TRANSFER VECTOR TO TABLE AREA
4565 012534  016237  001474  002720            MOV     $DDW0(R2),DDW   ;SET UP DDW TO PROPER DEVICE DESCRIPTOR WORD
4566 012542  013737  002720  002552            MOV     DDW,DRLEV       ;MOVE THE WORD TO THE DRLEV LOCATION
```

```
 4567 012550  042737  177437  002552          BIC     #177437,DRLEV   ;STRIP ALL BITS EXCEPT BR LEVEL
 4568 012556  105037  032555          REINIT: CLRB    CHARCT          ;CLEAR THE CHARACTER LOCATION OF ANY CHARACTER
 4569 012562  004737  004154                  JSR     PC,BPINIT       ;GO RESET THE ".+2" AND "BPT" LOCATIONS
 4570 012566  004737  003646                  JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
 4571 012572  105737  002706                  TSTB    EOPLOC          ;SEE IF ^X IS ENABLED (IS THE PRINTER DISABLED)
 4572 012576  001003                          BNE     TST1            ;GO DO TEST IF NOT
 4573                                 ;;*****************************************************************
 4574                                 ;*      *DO*NOT*REMOVE*THE*WAIT*LOOP*ROUTINE*BELOW*.  BECAUSE OF THE SPEED OF THIS DIAG-
 4575                                 ;*      NOSTIC (.125 SECOND FOR 1 PASS, NO ERRORS), SOME VIDEO TERMINALS PRINT ERRONEOUS
 4576                                 ;*      CHARACTER(S) WITH THE EOP MESSAGE DUE TO THE RESET EXECUTED IN TEST 4.  THIS WAIT
 4577                                 ;*      LOOP ENABLES THOSE TERMINALS TO 'CATCH UP' BEFORE ITS EXECUTION.
 4578                                 ;;*****************************************************************
 4579 012600  005327  000000          1$:     DEC     #0              ;DECREMENT A LOCATION TO KILL TIME
 4580 012604  001375                          BNE     1$              ;BRANCH BACK UNTIL ZERO AGAIN
 4581                                 ;;*****************************************************************
 4582                                 ;
 4583                                 ;MAIN PROGRAM - DEVICE TESTS
 4584                                 ;
 4585                                 ;;*****************************************************************
```

TEST #1 - CAN ALL DR11 REG BE ADDRESSED WITHOUT ERROR?

```
4593                                    .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.
                                        ;*
                                        ;;*****************************************************************
     012606                             TST1:
     012606  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     012610  012737  012644  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
     012616  032777  004000  166514             BIT     #BIT11,@SWR     ;@@ TEST TO SEE IF TEST NUMBER TRACER ENABLED
     012624  001407                             BEQ     1001$           ;@@ BRANCH IF NOT
     012626  104401  012634                     TYPE    ,1000$          ;@@ TYPE: "T # 1"
     012632  000404                             BR      1001$           ;@@ GET OVER ASCII
     012634     124     040     043  1000$:    .ASCIZ  /T # 1/<CRLF>   ;@@ THE ASCII MESSAGE
                                             .EVEN
     012644                             1001$:
4594 012644  005037  001312             999$:  CLR     $ERTTL          ;CLEAR THE ERROR TOTAL - NEW PASS
4595 012650  012737  012752  000004             MOV     #1$,BUSERR      ;CPU ERROR VECTOR TO 1$
4596 012656  013737  002516  002550             MOV     WCR,DREG        ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
4597 012664  005777  167626                     TST     @WCR            ;ACCESS REGISTER
4598 012670  013737  002520  002550             MOV     BAR,DREG        ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
4599 012676  005777  167616                     TST     @BAR            ;ACCESS REGISTER
4600 012702  013737  002522  002550             MOV     CSR,DREG        ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
4601 012710  005777  167606                     TST     @CSR            ;ACCESS REGISTER
4602 012714  013737  002524  002550             MOV     BDR,DREG        ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
4603 012722  005777  167576                     TST     @BDR            ;ACCESS REGISTER
4604 012726  013737  002526  002550             MOV     DRINV,DREG      ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
4605 012734  005777  167566                     TST     @DRINV          ;ACCESS REGISTER
4606 012740  005737  001312                     TST     $ERTTL          ;SEE IF THERE WERE ANY ERRORS
4607 012744  001414                             BEQ     2$              ;BRANCH TO CONTINUE IF NONE
4608 012746  000137  025622                     JMP     ENDEV           ;GO TO END OF DEVICE ROUTINE - FATAL ERRORS
4609 012752  012637  002672             1$:     MOV     (SP)+,OLDPC1    ;SAVE PC OF TRAP FOR ERROR PRINTOUT
4610 012756  012637  002674                     MOV     (SP)+,OLDPS1    ;SAVE PS FOR RESTORATION AFTER ERROR CALL
4611 012762  104001                             ERROR   +1              ;CANNOT ACCESS DR11 REGISTER
4612 012764  013746  002674                     MOV     OLDPS1,-(SP)    ;PUT PS BACK ON STACK
4613 012770  013746  002672                     MOV     OLDPC1,-(SP)    ;PUT PC BACK ON STACK
4614 012774  000002                             RTI                     ;RETURN TO PROGRAM
4615 012776  012737  000006  000004     2$:     MOV     #6,BUSERR       ;RESTORE #6 TO BUS ERROR
```

TEST #2 - CHECK B OR W STATUS IS AS EXPECTED

```
                                        .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.
                                        ;*
                                        ;;*****************************************************************
013004                                  TST2:
013004  000004                                  SCOPE                           ;PROCESS LOOPING AND TEST NUMBER INCREMENT
013006  012737  013042  001310                  MOV     #999$,$LPERR            ;SET LOOP ON ERROR TO 999$
013014  032777  004000  166316                  BIT     #BIT11,@SWR             ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
013022  001407                                  BEQ     1001$                   ;&& BRANCH IF NOT
013024  104401  013032                          TYPE    ,1000$                  ;&& TYPE: "T # 2"
013030  000404                                  BR      1001$                   ;&& GET OVER ASCII
013032    124     040     043           1000$:  .ASCIZ  /T # 2/<CRLF>           ;&& THE ASCII MESSAGE
                                                .EVEN
013042                                  1001$:
4623 013042  005077  167454            999$:   CLR     @CSR                    ;GO TO CSR
4624 013046  012777  100000  167446            MOV     #EIR,@CSR               ;FORCE TO BE EIR
4625 013054  017737  167442  002610            MOV     @CSR,BORW               ;ATTEMPT EIR READ
4626 013062  042737  177776  002610            BIC     #CBIT0,BORW             ;MASK OFF ALL BITS EXCEPT BIT 0
4627 013070  013737  002720  001362            MOV     DDW,$TMP1               ;GET DEVICE DESCRIPTOR WORD
4628 013076  042737  177776  001362            BIC     #CBIT0,$TMP1            ;MASK OFF ALL BUT B OR W BIT
4629 013104  001403                            BEQ     1$                      ;BRANCH IF IT IS CLEAR
4630 013106  005037  001362                    CLR     $TMP1                   ;CLEAR THE BIT
4631 013112  000403                            BR      2$                      ;GO TEST THE BIT
4632 013114  012737  000001  001362    1$:     MOV     #1,$TMP1                ;SET THE BIT
4633 013122  023737  002610  001362    2$:     CMP     BORW,$TMP1              ;B OR W STATE AS EXPECTED?
4634 013130  001401                            BEQ     TST3                    ;BRANCH IF OK
4635 013132  104002                            ERROR   +2                      ;DR11-B OR W MODE INCORRECT (0=B, 1=W)
```

```
4643                                    .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.
                                        ;*
                                        ;*****************************************************************
     013134                             TST3:
     013134  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     013136  012737  013246  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
     013144  032777  004000  166166             BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     013152  001407                             BEQ     1001$           ;&& BRANCH IF NOT
     013154  104401  013162                     TYPE    ,1000$          ;&& TYPE: ''T # 3''
     013160  000404                             BR      1001$           ;&& GET OVER ASCII
     013162     124     040     043  1000$:    .ASCIZ  /T # 3/<CRLF>   ;&& THE ASCII MESSAGE
                                             .EVEN
     013172                             1001$:
4644 013172  004737  003646                     JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
4645 013176  012737  000200  001362             MOV     #RY,$TMP1       ;MOVE READY BIT TO $TMP1
4646 013204  032737  000004  002720             BIT     #BIT2,DDW       ;TEST TO SEE IF CABLE IS IN
4647 013212  001003                             BNE     1$              ;BRANCH AROUND NON-CABLE SETUP IF IN
4648 013214  052737  127000  001362             BIS     #127000,$TMP1   ;SET THE BITS TO BE EXPECTED IN $TMP1
4649 013222  012701  010220          1$:        MOV     #MAISET,R1      ;MOVE ADDRESS OF EXPECTED PATTERNS TO R1
4650 013226  012702  010000                     MOV     #MA,R2          ;START WITH JUST THE MAINTENANCE BIT
4651 013232  005037  002714                     CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +202
4652 013236  012700  000002                     MOV     #2,R0           ;DO 2 SETS OF 200 PATTERNS
4653 013242  012703  000200          2$:        MOV     #200,R3         ;MOVE 200 TO THE LOOP COUNTER
4654 013246  052777  010000  167246  999$:      BIS     #MA,@CSR        ;SET MAINTENANCE AND
4655 013254  005077  167242                     CLR     @CSR            ;CLEAR TO DO AN INIT
4656 013260  017737  167236  002560             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
4657 013266  023737  001362  002560             CMP     $TMP1,RCSR      ;MAKE SURE EXPECTED DATA CAME UP
4658 013274  001404                             BEQ     3$              ;BRANCH IF SO
4659 013276  013737  001362  002572             MOV     $TMP1,ECSR      ;MOVE EXPECTED DATA TO ECSR
4660 013304  104032                             ERROR   +32             ;CSR IS WRONG
4661 013306  012777  177777  167202  3$:        MOV     #-1,@WCR        ;MOVE 1 WORD COUNT TO WCR IN CASE OF IE ENABLED
4662 013314  012777  056104  167176             MOV     #NOCARE,@BAR    ;MOVE A NOT-CARE ADDRESS TO BAR FOR SAME REASON
4663 013322  010277  167174                     MOV     R2,@CSR         ;SET THE PARTICULAR FUNCTION BITS IN CSR
4664 013326  017737  167170  002560             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
4665 013334  011137  002572                     MOV     (R1),ECSR       ;MOVE EXPECTED DATA TO ECSR
4666 013340  023737  002572  002560             CMP     ECSR,RCSR       ;COMPARE EXPECTED WITH RECEIVED
4667 013346  001425                             BEQ     7$              ;BRANCH IF OK
4668 013350  012737  000401  013360             MOV     #CY+GO,4$       ;REESTABLISH TEST PATTERN
4669 013356  040227                             BIC     R2,(PC)+        ;SEE IF BOTH CYCLE AND GO WERE SET
4670 013360  000401                     4$:     .WORD   CY+GO           ;LOCATION TO HOLD BOTH CYCLE AND GO BITS
4671 013362  001013                             BNE     6$              ;BRANCH TO ERROR ONLY IF CYCLE AND GO WERE SET
4672 013364  032737  000060  002572             BIT     #X6+X7,ECSR     ;SEE IF EITHER XBA16 OR XBA17 ARE SET
4673 013372  001407                             BEQ     6$              ;BRANCH TO ERROR IF BOTH ARE CLEAR
4674 013374  052737  140000  002572  5$:        BIS     #ER+NX,ECSR     ;SET THE ERROR AND NEX BITS - EXPECT THEM TO SET
4675 013402  023737  002560  002572             CMP     RCSR,ECSR       ;NOW SEE IF DATA MATCHES
4676 013410  001415                             BEQ     10$             ;BRANCH AROUND ERROR IF IT DOES
4677 013412  010237  002536                     MOV     R2,BUT          ;MOVE THE BITS SET INTO CSR TO THE BUT LOCATION
4678 013416  104202                             ERROR   +202            ;CSR PATTERN NOT CORRECT
4679 013420  000411                             BR      10$             ;BRANCH AROUND MM TESTS
4680 013422  012737  000401  013432  7$:        MOV     #CY+GO,8$       ;REESTABLISH TEST PATTERN
4681 013430  040227                             BIC     R2,(PC)+        ;SEE IF BOTH CYCLE AND GO WERE SET
```

TEST #3 - CHECK CSR BIT PATTERNS WITH MAINT BIT SET

```
4682 013432  000401                 8$:     .WORD   CY+GO           ;LOCATION TO HOLD BOTH CYCLE AND GO BITS
4683 013434  001003                         BNE     10$             ;BRANCH AROUND MEM MGMT TEST IF EITHER OR BOTH WERE CLEAR
4684 013436  004737  007066         9$:     JSR     PC,TSTMM        ;GO CHECK FOR MEMORY MANAGEMENT EXISTENCE
4685 013442  000754                         BR      5$              ;IF RETURN IS HERE, GO BACK TO SET EXPECTED DATA
4686 013444  062701  000002         10$:    ADD     #2,R1           ;INCREMENT R1 TO NEXT EXPECTED PATTERN
4687 013450  005202                         INC     R2              ;INCREMENT THE PATTERN
4688 013452  005303                         DEC     R3              ;DECREMENT THE LOOP COUNTER
4689 013454  001274                         BNE     999$            ;BRANCH BACK IF NOT DONE
4690 013456  062702  000200               ADD     #200,R2         ;ADD 200 TO PATTERN LOCATION
4691 013462  005300                         DEC     R0              ;DECREMENT THE LOOP COUNTER AND
4692 013464  001266                         BNE     2$              ;BRANCH BACK IF 2ND OCTAL GROUP NOT DONE
```

```
4699                                    .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.
                                        ;*
                                        ;;*****************************************************************
013466                                  TST4:
013466  000004                                SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
013470  012737  013524  001310                MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
013476  032777  004000  165634                BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
013504  001407                                BEQ     1001$           ;&& BRANCH IF NOT
013506  104401  013514                        TYPE    ,1000$          ;&& TYPE: "T # 4"
013512  000404                                BR      1001$           ;&& GET OVER ASCII
013514    124     040     043       1000$:    .ASCIZ  /T # 4/<CRLF>   ;&& THE ASCII MESSAGE
                                              .EVEN
013524                                  1001$:
4700 013524  012777  010000  166770    999$:   MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
4701 013532  005077  166764                    CLR     @CSR            ;CLEAR TO DO AN INIT
4702 013536  012777  010000  166756            MOV     #MA,@CSR        ;DO THIS TEST IN MAINTENANCE MODE       ;DPM001
4703 013544  012777  177777  166744            MOV     #-1,@WCR        ;ALL ONES TO WCR
4704 013552  017737  166740  002570            MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
4705 013560  022737  177777  002570            CMP     #-1,RWCR        ;SEE IF DATA WAS LOADED PROPERLY
4706 013566  001423                            BEQ     4$              ;BRANCH IF OK
4707 013570  012737  013600  001310            MOV     #1$,$LPERR      ;MOVE NEW LOOP ON ERROR LOCATION TO $LPERR
4708 013576  000412                            BR      2$              ;BRANCH OVER LOOP SETUP
4709 013600  012777  177777  166710    1$:     MOV     #-1,@WCR        ;ALL ONES TO WCR
4710 013606  017737  166704  002570            MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
4711 013614  022737  177777  002570            CMP     #-1,RWCR        ;SEE IF DATA WAS LOADED PROPERLY
4712 013622  001401                            BEQ     3$              ;BRANCH IF OK
4713 013624  104010                    2$:     ERROR   +10             ;ATTEMPT TO SET ALL WCR BITS FAILED
4714 013626  032777  001000  165504    3$:     BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
4715 013634  001361                            BNE     1$              ;BRANCH BACK IF SO
4716 013636  052777  000001  166656    4$:     BIS     #GO,@CSR        ;SET GO TO CLEAR READY TO PREPARE TO SET BAR BIT 0
4717 013644  017737  166652  002560            MOV     @CSR,RCSR       ;READ CSR TO SET BIT 0 OF BAR           ;DPM001
4718 013652  012777  177777  166640            MOV     #-1,@BAR        ;ALL ONES TO BAR
4719 013660  017737  166634  002566            MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
4720 013666  022737  177777  002566            CMP     #-1,RBAR        ;SEE IF ALL BITS WERE SET
4721 013674  001431                            BEQ     8$              ;BRANCH IF OK
4722 013676  012737  013714  001310            MOV     #5$,$LPERR      ;MOVE NEW LOOP ON ERROR LOCATION TO $LPERR
4723 013704  012737  177777  002600            MOV     #-1,EBAR        ;MOVE EXPECTED DATA TO EBAR
4724 013712  000415                            BR      6$              ;BRANCH OVER LOOP SETUP
4725 013714  017737  166602  002560    5$:     MOV     @CSR,RCSR       ;ACCESS CSR TO SET BIT 0 OF BAR
4726 013722  012777  177777  166570            MOV     #-1,@BAR        ;ALL ONES TO BAR
4727 013730  017737  166564  002566            MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
4728 013736  022737  177777  002566            CMP     #-1,RBAR        ;SEE IF ALL BITS WERE SET
4729 013744  001401                            BEQ     7$              ;BRANCH IF OK
4730 013746  104012                    6$:     ERROR   +12             ;ATTEMPT TO SET ALL BAR BITS TO 1 FAILED
4731 013750  032777  001000  165362    7$:     BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
4732 013756  001356                            BNE     5$              ;BRANCH BACK IF SO
4733 013760  012777  177777  166536    8$:     MOV     #-1,@BDR        ;ALL ONES TO BDR
4734 013766  017737  166532  002564            MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
4735 013774  022737  177777  002564            CMP     #-1,RBDR        ;SEE IF DATA WAS LOADED PROPERLY
4736 014002  001423                            BEQ     12$             ;BRANCH IF OK
4737 014004  012737  014102  001310            MOV     #13$,$LPERR     ;MOVE NEW LOOP ON ERROR LOCATION TO $LPERR
4738 014012  000412                            BR      10$             ;BRANCH OVER LOOP SETUP
```

```
4739 014014  012777  177777  166502  9$:     MOV     #-1,@BDR        ;ALL ONES TO BDR
4740 014022  017737  166476  002564          MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
4741 014030  022737  177777  002564          CMP     #-1,RBDR        ;SEE IF DATA WAS LOADED PROPERLY
4742 014036  001401                          BEQ     11$             ;BRANCH IF OK
4743 014040  104027                  10$:    ERROR   +27             ;ALL BDR BITS ARE NOT SET
4744 014042  032777  001000  165270  11$:    BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
4745 014050  001361                          BNE     9$              ;BRANCH BACK IF SO
4746 014052  012777  010576  166442  12$:    MOV     #10576,@CSR     ;SET ALL CSR WRITEABLE BITS
4747 014060  000005                          RESET                   ;RESET THE WORLD OF ITS TROUBLES - HOPEFULLY
4748 014062  012737  013524  001310          MOV     #999$,$LPERR    ;RESET THE LOOP ON ERROR LOCATION
4749 014070  017737  166422  002570          MOV     @WCR,RWCR       ;WAS WCR CLEARED?
4750 014076  001401                          BEQ     13$             ;BRANCH IF WCR WAS CLEARED
4751 014100  104007                          ERROR   +7              ;RESET FAILED TO CLEAR WCR
4752 014102  017737  166412  002566  13$:    MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
4753 014110  001403                          BEQ     14$             ;BRANCH IF BAR WAS CLEARED
4754 014112  005037  002600                  CLR     EBAR            ;CLEAR EXPECTED LOCATION
4755 014116  104011                          ERROR   +11             ;RESET FAILED TO CLEAR BAR
4756 014120  017737  166376  002560  14$:    MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
4757 014126  012737  000200  002572          MOV     #RY,ECSR        ;MOVE EXPECTED DATA TO ECSR
4758 014134  004737  003670                  JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
4759 014140  023737  002572  002560          CMP     ECSR,RCSR       ;SEE IF EXPECTED DATA WAS RECEIVED
4760 014146  001401                          BEQ     15$             ;BRANCH IF IT WAS
4761 014150  104032                          ERROR   +32             ;READY IS NOT THE ONLY BIT SET
4762 014152  052777  010000  166342  15$:    BIS     #MA,@CSR        ;MAINT MODE (SO THAT IDR GETS ODR CONTENTS)
4763 014160  017737  166340  002564          MOV     @BDR,RBDR       ;MOVE CONTENTS OF BDR TO RBDR
4764 014166  001401                          BEQ     TST5            ;BRANCH IF IT CORRECTLY REMAINS ZERO
4765 014170  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.
                                        ;*
                                        ;;*****************************************************************
     014172  000004                     TST5:   SCOPE
     014174  032777  004000  165136             BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     014202  001407                             BEQ     1001$           ;&& BRANCH IF NOT
     014204  104401  014212                     TYPE    ,1000$          ;&& TYPE: "T # 5"
     014210  000404                             BR      1001$           ;&& GET OVER ASCII
     014212     124     040     043     1000$: .ASCIZ  /T # 5/<CRLF>   ;&& THE ASCII MESSAGE
                                                .EVEN
     014222                             1001$:
4772 014222  005077  166274             999$:   CLR     @CSR            ;FORCE ACCESS TO CSR
4773 014226  012777  177777  166262             MOV     #-1,@WCR        ;ALL ONES TO WCR
4774 014234  012777  177777  166262             MOV     #-1,@BDR        ;ALL ONES TO BDR
4775 014242  012777  177777  166250             MOV     #-1,@BAR        ;ALL ONES TO BAR
4776 014250  012777  010576  166244             MOV     #10576,@CSR     ;SET ALL WRITEABLE BITS IN THE CSR
4777 014256  042777  010000  166236             BIC     #MA,@CSR        ;CLEAR THE MAINT BIT TO DO AN INIT
4778 014264  017737  166226  002570             MOV     @WCR,RWCR       ;MOVE RECEIVED CONTENTS TO RWCR
4779 014272  001401                             BEQ     1$              ;BRANCH IF WCR WAS CLEARED
4780 014274  104003                             ERROR   +3              ;INIT FAILED TO CLEAR WCR
4781 014276  017737  166216  002566     1$:     MOV     @BAR,RBAR       ;MOVE RECEIVED CONTENTS TO RBAR
4782 014304  001403                             BEQ     2$              ;BRANCH IF BAR WAS CLEARED
4783 014306  005037  002600                     CLR     EBAR            ;CLEAR EXPECTED LOCATION
4784 014312  104004                             ERROR   +4              ;INIT FAILED TO CLEAR BAR
4785 014314  017737  166202  002560     2$:     MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
4786 014322  012737  000200  002572             MOV     #RY,ECSR        ;EXPECT READY BIT ONLY TO BE SET
4787 014330  004737  003670                     JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
4788 014334  023737  002572  002560             CMP     ECSR,RCSR       ;SEE IF EXPECTED DATA WAS RECEIVED
4789 014342  001401                             BEQ     3$              ;BRANCH IF THEY WERE ALL CLEAR
4790 014344  104006                             ERROR   +6              ;INIT FAILED TO CLEAR ALL CSR R-W BITS
4791 014346  012777  010000  166146     3$:     MOV     #MA,@CSR        ;GO BACK INTO MAINT MODE (SO THAT IDR GETS ODR CONTENTS)
4792 014354  017737  166144  002564             MOV     @BDR,RBDR       ;MOVE RECEIVED CONTENTS TO RBDR
4793 014362  001401                             BEQ     TST6            ;BRANCH IF IT WAS CLEARED
4794 014364  104005                             ERROR   +5              ;INIT FAILED TO CLEAR BDR
```

TEST #6 - BIT PATTERN TEST OF WCR, BDR AND BAR REGISTERS

```
4804                                    .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 $LPERR IS INITIALIZED TO A ROUTINE SPECIFI-
                                        ;*      CALLY WRITTEN FOR THAT PARTICULAR ERROR TO CREATE A VERY TIGHT LOOP.
                                        ;*
                                        ;;*****************************************************************
     014366  000004                     TST6:   SCOPE
     014370  032777  004000  164742             BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     014376  001407                             BEQ     1001$           ;&& BRANCH IF NOT
     014400  104401  014406                     TYPE    ,1000$          ;&& TYPE: 'T # 6'
     014404  000404                             BR      1001$           ;&& GET OVER ASCII
     014406     124     040     043     1000$: .ASCIZ  /T # 6/<CRLF>   ;&& THE ASCII MESSAGE
                                                .EVEN
     014416                             1001$:
4805 014416  004737  003646                     JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
4806 014422  005037  002714                     CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +204
4807 014426  012701  007202                     MOV     #PATRNS,R1      ;MOVE ADDRESS OF BIT PATTERNS TO R1
4808 014432  012702  000007                     MOV     #7,R2           ;DO 7 PATTERNS
4809 014436  012777  010000  166056     1$:     MOV     #MA,@CSR        ;GO TO MAINTENANCE MODE & SET GO TO DISABLE BAR BIT 0
4810 014444  011177  166054                     MOV     (R1),@BDR       ;MOVE THE DATA TO BDR
4811 014450  017737  166050  002564             MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
4812 014456  021137  002564                     CMP     (R1),RBDR       ;SEE IF EXPECTED DATA WAS RECEIVED
4813 014462  001423                             BEQ     5$              ;BRANCH IF SO
4814 014464  012737  014500  001310             MOV     #2$,$LPERR      ;SET UP LOOP ON ERROR LOCATION
4815 014472  011137  002576                     MOV     (R1),EBDR       ;MOVE EXPECTED DATA TO EBDR
4816 014476  000410                             BR      3$              ;SKIP OVER LOOP ON ERROR SETUP
4817 014500  011177  166020             2$:     MOV     (R1),@BDR       ;LOAD BIT PATTERN TO BDR
4818 014504  017737  166014  002564             MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
4819 014512  021137  002564                     CMP     (R1),RBDR       ;SEE IF DATA IS OK NOW
4820 014516  001401                             BEQ     4$              ;BRANCH OUT IF SO - OK NOW
4821 014520  104205                     3$:     ERROR   +205            ;BDR PATTERN NOT CORRECT
4822 014522  032777  001000  164610     4$:     BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
4823 014530  001363                             BNE     2$              ;BRANCH BACK IF SO
4824 014532  005077  165764             5$:     CLR     @CSR            ;CLEAR CSR TO DO AN INIT - WCR & BAR DON'T NEED MAINT MODE
4825 014536  011177  165754                     MOV     (R1),@WCR       ;MOVE THE DATA TO WCR
4826 014542  017737  165750  002570             MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
4827 014550  021137  002570                     CMP     (R1),RWCR       ;SEE IF EXPECTED DATA WAS RECEIVED
4828 014554  001423                             BEQ     9$              ;BRANCH IF SO
4829 014556  012737  014572  001310             MOV     #6$,$LPERR      ;SET UP LOOP ON ERROR LOCATION
4830 014564  011137  002602                     MOV     (R1),EWCR       ;MOVE EXPECTED DATA TO EWCR
4831 014570  000410                             BR      7$              ;SKIP OVER LOOP ON ERROR SETUP
4832 014572  011177  165720             6$:     MOV     (R1),@WCR       ;LOAD BIT PATTERN TO WCR
4833 014576  017737  165714  002570             MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
4834 014604  021137  002570                     CMP     (R1),RWCR       ;SEE IF DATA IS OK NOW
4835 014610  001401                             BEQ     8$              ;BRANCH OUT IF SO - OK NOW
4836 014612  104204                     7$:     ERROR   +204            ;WCR DATA PATTERN NOT CORRECT
4837 014614  032777  001000  164516     8$:     BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
4838 014622  001363                             BNE     6$              ;BRANCH BACK IF SO
4839 014624  011177  165670             9$:     MOV     (R1),@BAR       ;MOVE THE DATA TO BAR
4840 014630  017737  165664  002566             MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
4841 014636  011137  002600                     MOV     (R1),EBAR       ;MOVE EXPECTED DATA TO EBAR
4842 014642  042737  000001  002600             BIC     #BIT0,EBAR      ;DO NOT EXPECT BIT 0 TO BE READ
```

TEST #6 - BIT PATTERN TEST OF WCR, BDR AND BAR REGISTERS

```
 4843 014650  023737  002600  002566          CMP     EBAR,RBAR       ;SEE IF EXPECTED DATA WAS RECEIVED
 4844 014656  001421                          BEQ     13$             ;BRANCH IF SO
 4845 014660  012737  014670  001310          MOV     #10$,$LPERR     ;SET UP LOOP ON ERROR LOCATION
 4846 014666  000410                          BR      11$             ;SKIP OVER LOOP ON ERROR SETUP
 4847 014670  011177  165624          10$:    MOV     (R1),@BAR       ;LOAD BIT PATTERN TO BAR
 4848 014674  017737  165620  002566          MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
 4849 014702  021137  002566                  CMP     (R1),RBAR       ;SEE IF DATA IS OK NOW
 4850 014706  001401                          BEQ     12$             ;BRANCH OUT IF SO - OK NOW
 4851 014710  104203                  11$:    ERROR   +203            ;BAR DATA PATTERN NOT CORRECT
 4852 014712  032777  001000  164420  12$:    BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
 4853 014720  001363                          BNE     10$             ;BRANCH BACK IF SO
 4854 014722  005721                  13$:    TST     (R1)+           ;GO TO NEXT PATTERN
 4855 014724  005302                          DEC     R2              ;DECREMENT THE LOOP COUNTER AND
 4856 014726  001243                          BNE     1$              ;BRANCH BACK IF NOT DONE
 4857 014730  004737  003646                  JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
```

```
                                        .SBTTL  TEST #7 - TEST CSR AND EIR BITO
                                        ;;*****************************************************************
                                        ;*TEST 7         TEST CSR AND EIR BITO
                                        ;*
                                        ;*      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).
                                        ;*
                                        ;;*****************************************************************
     014734                             TST7:
     014734  000004                             SCOPE                           ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     014736  012737  014772  001310             MOV     #999$,$LPERR            ;SET LOOP ON ERROR TO 999$
     014744  032777  004000  164366             BIT     #BIT11,@SWR             ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     014752  001407                             BEQ     1001$                   ;&& BRANCH IF NOT
     014754  104401  014762                     TYPE    ,1000$                  ;&& TYPE: "T # 7"
     014760  000404                             BR      1001$                   ;&& GET OVER ASCII
     014762     124     040     043  1000$:     .ASCIZ  /T # 7/<CRLF>           ;&& THE ASCII MESSAGE
                                                .EVEN
     014772                             1001$:
4855 014772  032737  000001  002610  999$:     BIT     #BIT0,BORW              ;TEST TO SEE IF WE ARE TESTING A DR11-W
4866 015000  001444                             BEQ     TST10                   ;BRANCH TO NEXT TEST IF A DR11-B
4867 015002  005077  165514                     CLR     @CSR                    ;FORCE ACCESS TO CSR
4868 015006  012737  000001  002536             MOV     #BIT0,BUT               ;MOVE BIT 0 INDICATOR TO BIT UNDER TEST LOCATION
4869 015014  017737  165502  002560             MOV     @CSR,RCSR               ;MOVE CSR CONTENTS TO RCSR
4870 015022  032737  000001  002560             BIT     #BIT0,RCSR              ;CLEAR ALL BUT BIT 0
4871 015030  001407                             BEQ     1$                      ;BRANCH IF A ZERO
4872 015032  013737  002560  002572             MOV     RCSR,ECSR               ;MOVE CSR TO EXPECTED DATA, ECSR AND
4873 015040  042737  000001  002572             BIC     #BIT0,ECSR              ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
4874 015046  104014                             ERROR   +14                     ;CSR BIT TEST FAILED
4875 015050  012777  100000  165444  1$:        MOV     #EIR,@CSR               ;GO TO EIR MODE
4876 015056  017737  165440  002562             MOV     @CSR,REIR               ;MOVE CSR CONTENTS TO RCSR
4877 015064  032737  000001  002562             BIT     #BIT0,REIR              ;CLEAR ALL BUT BIT 0
4878 015072  001007                             BNE     TST10                   ;BRANCH IF NOT A ZERO
4879 015074  013737  002562  002574             MOV     REIR,EEIR               ;MOVE CONTENTS TO ECSR ALSO AND
4880 015102  052737  000001  002574             BIS     #BIT0,EEIR              ;SET THE 0 BIT - EXPECTED IT TO BE 1
4881 015110  104015                             ERROR   +15                     ;EIR BIT TEST FAILED
```

TEST #10 - ATTN CAN BE SET VIA FNCT2 & ERROR BIT SETS

```
4888                                    .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.
                                        ;*
                                        ;;*****************************************************************
      015112                            TST10:
      015112  000004                            SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
      015114  012737  015160  001310            MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
      015122  032777  004000  164210            BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
      015130  001407                            BEQ     1001$           ;&& BRANCH IF NOT
      015132  104401  015140                    TYPE    ,1000$          ;&& TYPE: "T # 10"
      015136  000404                            BR      1001$           ;&& GET OVER ASCII
      015140     124     040     043  1000$:    .ASCIZ  /T # 10/<CRLF>  ;&& THE ASCII MESSAGE
                                                .EVEN
      015150                            1001$:
4889  015150  032737  000001  002720            BIT     #BIT0,DDW       ;TEST TO SEE IF WE ARE TESTING A DR11-W
4890  015156  001073                            BNE     TST11           ;BRANCH TO NEXT TEST IF A DR11-B
4891  015160  005077  165336            999$:   CLR     @CSR            ;FORCE ACCESS TO CSR
4892  015164  012777  010000  165330            MOV     #MA,@CSR        ;MAINT
4893  015172  017737  165324  002560            MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
4894  015200  022737  010200  002560            CMP     #MA+RY,RCSR     ;SEE IF EXPECTED DATA WAS RECEIVED
4895  015206  001404                            BEQ     1$              ;BRANCH IF THEY ARE
4896  015210  012737  010200  002572            MOV     #MA+RY,ECSR     ;MOVE EXPECTED DATA TO ECSR
4897  015216  104016                            ERROR   +16             ;READY AND MAINTENANCE ARE NOT THE ONLY BITS SET IN CSR
4898  015220  112777  000004  165274    1$:     MOVB    #F2,@CSR        ;SET FNCT2
4899  015226  017737  165270  002560            MOV     @CSR,RCSR       ;MOVE THE CONTENTS TO RCSR
4900  015234  013701  002560                    MOV     RCSR,R1         ;MOVE CONTENTS TO R1 FOR BIT TEST
4901  015240  042701  057777                    BIC     #CB1513,R1      ;CLEAR ALL BUT BITS ERROR & ATTN FOR TEST
4902  015244  022701  120000                    CMP     #ER+AT,R1       ;TEST TO SEE IF ERROR AND ATTN ARE SET
4903  015250  001411                            BEQ     2$              ;BRANCH IF IT IS PROPERLY SET
4904  015252  013737  002560  002572            MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
4905  015260  004737  003670                    JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
4906  015264  052737  120000  002572            BIS     #ER+AT,ECSR     ;SET THE BITS THAT SHOULD HAVE BEEN SET
4907  015272  104017                            ERROR   +17             ;ATTN AND ERROR FAILED TO SET PROPERLY
4908  015274  042777  020004  165220    2$:     BIC     #AT+F2,@CSR     ;CLEAR ATTN & FNCT2
4909  015302  017737  165214  002560            MOV     @CSR,RCSR       ;MOVE CSR DATA TO RCSR
4910  015310  032737  120000  002560            BIT     #ER+AT,RCSR     ;BIT TEST ATTN AND ERROR BITS TO SEE IF THEY ARE CLEAR
4911  015316  001411                            BEQ     3$              ;BRANCH IF ATTN IS CLEAR
4912  015320  013737  002560  002572            MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
4913  015326  042737  120000  002572            BIC     #ER+AT,ECSR     ;CLEAR THE BITS THAT WERE SUPPOSED TO BE CLEAR
4914  015334  004737  003670                    JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
4915  015340  104020                            ERROR   +20             ;ATTN AND ERROR FAILED TO CLEAR PROPERLY
4916  015342  005077  165154            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.
                                        ;*
                                        ;;*****************************************************************
 015346                                 TST11:
 015346  000004                                 SCOPE                           ;PROCESS LOOPING AND TEST NUMBER INCREMENT
 015350  012737  015404  001310                 MOV     #999$,$LPERR            ;SET LOOP ON ERROR TO 999$
 015356  032777  004000  163754                 BIT     #BIT11,@SWR             ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
 015364  001407                                 BEQ     1001$                   ;&& BRANCH IF NOT
 015366  104401  015374                         TYPE    ,1000$                  ;&& TYPE: "T # 11"
 015372  000404                                 BR      1001$                   ;&& GET OVER ASCII
 015374     124     040     043         1000$: .ASCIZ  /T # 11/<CRLF>          ;&& THE ASCII MESSAGE
                                                .EVEN
 015404                                 1001$:
4923 015404  005077  165112             999$:  CLR     @CSR                    ;CLR FUNCT BITS AND FORCE ACCESS TO CSR
4924 015410  012777  010000  165104             MOV     #MA,@CSR                ;MAINT MODE
4925 015416  017737  165100  002560             MOV     @CSR,RCSR               ;MOVE CONTENTS OF CSR TO RCSR
4926 015424  013737  002560  002572             MOV     RCSR,ECSR               ;MOVE EXPECTED TO ECSR
4927 015432  013701  002572                     MOV     ECSR,R1                 ;MOVE CONTENTS TO R1 FOR TESTING
4928 015436  042701  177761                     BIC     #CFNC,R1                ;CLEAR ALL BUT THE FNCT BITS
4929 015442  001404                             BEQ     1$                      ;BRANCH IF THE FUNCTION BITS ARE CLEAR
4930 015444  042737  000016  002572             BIC     #FNC,ECSR               ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
4931 015452  104024                             ERROR   +24                     ;FUNCTION BIT(S) ARE NOT CLEAR
4932 015454  052777  000002  165040     1$:     BIS     #F1,@CSR                ;SET FNCT1
4933 015462  017737  165034  002560             MOV     @CSR,RCSR               ;MOVE CONTENTS OF CSR TO RCSR
4934 015470  013737  002560  002572             MOV     RCSR,ECSR               ;MOVE EXPECTED DATA TO ECSR
4935 015476  013701  002572                     MOV     ECSR,R1                 ;MOVE CONTENTS TO R1 FOR TEST
4936 015502  042701  170777                     BIC     #CDST,R1                ;CLEAR ALL BUT BITS 9, 10 & 11
4937 015506  022701  001000                     CMP     #DSC,R1                 ;SEE IF DSTAT A AND B ARE CLEAR & C IS SET
4938 015512  001407                             BEQ     TST12                   ;BRANCH TO NEXT TEST IF ALL CLEAR
4939 015514  042737  006000  002572             BIC     #DAB,ECSR               ;CLEAR THE DSTAT A & B BITS THAT SHOULD HAVE BEEN CLEAR
4940 015522  052737  001000  002572             BIS     #DSC,ECSR               ;SET THE DSTAT C BIT THAT SHOULD HAVE BEEN SET
4941 015530  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.
                                        ;*
                                        ;;*****************************************************************
 015532                                 TST12:
 015532  000004                                SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
 015534  012737  015570  001310                 MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
 015542  032777  004000  163570                 BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
 015550  001407                                 BEQ     1001$           ;&& BRANCH IF NOT
 015552  104401  015560                         TYPE    ,1000$          ;&& TYPE: "T # 12"
 015556  000404                                 BR      1001$           ;&& GET OVER ASCII
 015560     124     040     043  1000$:        .ASCIZ  /T # 12/<CRLF>  ;&& THE ASCII MESSAGE
                                                .EVEN
 015570                                 1001$:
4948 015570  005077  164726          999$:     CLR     @CSR            ;CLR FUNCT BITS AND FORCE ACCESS TO CSR
4949 015574  012777  010000  164720             MOV     #MA,@CSR        ;MAINT MODE
4950 015602  017737  164714  002560             MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
4951 015610  013737  002560  002572             MOV     RCSR,ECSR       ;MOVE EXPECTED TO ECSR
4952 015616  013701  002572                     MOV     ECSR,R1         ;MOVE CONTENTS TO R1 FOR TESTING
4953 015622  042701  177761                     BIC     #CFNC,R1        ;CLEAR ALL BUT THE FNCT BITS
4954 015626  001404                             BEQ     1$              ;BRANCH IF THE FUNCTION BITS ARE CLEAR
4955 015630  042737  000016  002572             BIC     #FNC,ECSR       ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
4956 015636  104024                             ERROR   +24             ;FUNCTION BIT(S) ARE NOT CLEAR
4957 015640  052777  000004  164654  1$:        BIS     #F2,@CSR        ;SET FNCT2
4958 015646  017737  164650  002560             MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
4959 015654  013737  002560  002572             MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
4960 015662  013701  002572                     MOV     ECSR,R1         ;MOVE CONTENTS TO R1 FOR TEST
4961 015666  042701  170777                     BIC     #CDST,R1        ;CLEAR ALL BUT THE DSTAT BITS
4962 015672  022701  002000                     CMP     #DSB,R1         ;IF DSTAT A AND C ARE CLEAR & B IS SET
4963 015676  001407                             BEQ     TST13           ;BRANCH TO NEXT TEST IF AS EXPECTED
4964 015700  042737  005000  002572             BIC     #DAC,ECSR       ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
4965 015706  052737  002000  002572             BIS     #DSB,ECSR       ;SET THE BIT THAT SHOULD HAVE BEEN SET
4966 015714  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.
                                        ;*
                                        ;;*****************************************************************
 015716                                 TST13:
 015716  000004                                 SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
 015720  012737  015754  001310                 MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
 015726  032777  004000  163404                 BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
 015734  001407                                 BEQ     1001$           ;&& BRANCH IF NOT
 015736  104401  015744                         TYPE    ,1000$          ;&& TYPE: "T # 13"
 015742  000404                                 BR      1001$           ;&& GET OVER ASCII
 015744     124     040     043 1000$:  .ASCIZ  /T # 13/<CRLF>  ;&& THE ASCII MESSAGE
                                        .EVEN
 015754                                 1001$:
4973 015754  005077  164542            999$:   CLR     @CSR            ;CLR FUNCT BITS AND FORCE ACCESS TO CSR
4974 015760  012777  010000  164534            MOV     #MA,@CSR        ;MAINT MODE
4975 015766  017737  164530  002560            MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
4976 015774  013737  002560  002572            MOV     RCSR,ECSR       ;MOVE EXPECTED TO ECSR
4977 016002  013701  002572                    MOV     ECSR,R1         ;MOVE CONTENTS TO R1 FOR TESTING
4978 016006  042701  177761                    BIC     #CFNC,R1        ;CLEAR ALL BUT THE FNCT BITS
4979 016012  001404                            BEQ     1$              ;BRANCH IF THE FUNCTION BITS ARE CLEAR
4980 016014  042737  000016  002572            BIC     #FNC,ECSR       ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
4981 016022  104024                            ERROR   +24             ;FUNCTION BIT(S) ARE NOT CLEAR
4982 016024  052777  000010  164470  1$:       BIS     #F3,@CSR        ;SET FNCT3
4983 016032  017737  164464  002560            MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
4984 016040  013737  002560  002572            MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
4985 016046  013701  002572                    MOV     ECSR,R1         ;MOVE CONTENTS TO R1 FOR TEST
4986 016052  042701  170777                    BIC     #CDST,R1        ;CLEAR ALL BUT DSTAT BITS
4987 016056  022701  004000                    CMP     #DSA,R1         ;SEE IF DSTAT B AND C ARE CLEAR & A IS SET
4988 016062  001407                            BEQ     TST14           ;BRANCH TO NEXT TEST IF DATA OK
4989 016064  042737  003000  002572            BIC     #DBC,ECSR       ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
4990 016072  052737  004000  002572            BIS     #DSA,ECSR       ;SET THE BIT THAT SHOULD HAVE BEEN SET
4991 016100  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).
                                        ;*
                                        ;;*****************************************************************
 4999   016102                          TST14:
        016102  000004                          SCOPE                           ;PROCESS LOOPING AND TEST NUMBER INCREMENT
        016104  012737  016150  001310          MOV     #999$,$LPERR            ;SET LOOP ON ERROR TO 999$
        016112  032777  004000  163220          BIT     #BIT11,@SWR             ;@@ TEST TO SEE IF TEST NUMBER TRACER ENABLED
        016120  001407                          BEQ     1001$                   ;@@ BRANCH IF NOT
        016122  104401  016130                  TYPE    ,1000$                  ;@@ TYPE: "T # 14"
        016126  000404                          BR      1001$                   ;@@ GET OVER ASCII
        016130     124     040     043  1000$: .ASCIZ   /T # 14/<CRLF>          ;@@ THE ASCII MESSAGE
        016140                          1001$: .EVEN
 5000   016140  032737  000001  002610          BIT     #BIT0,@ORW              ;TEST TO SEE IF WE ARE TESTING A DR11-W
 5001   016146  001451                          BEQ     TST15                   ;BRANCH TO NEXT TEST IF A DR11-B
 5002   016150  005077  164346          999$:   CLR     @CSR                    ;FORCE ACCESS TO CSR
 5003   016154  012777  010000  164340          MOV     #MA,@CSR                ;SET MAINT MODE (SO THAT IDR GETS ODR CONTENTS)
 5004   016162  012777  052525  164334          MOV     #52525,@BDR             ;SET ALT 0'S AND 1'S TO BDR
 5005   016170  017737  164330  002564          MOV     @BDR,RBDR               ;MOVE RECEIVED DATA TO RBDR
 5006   016176  022737  052525  002564          CMP     #52525,RBDR             ;SEE IF DATA WAS LOADED PROPERLY
 5007   016204  001404                          BEQ     1$                      ;BRANCH IF IT WAS
 5008   016206  012737  052525  002576          MOV     #52525,EBDR             ;MOVE EXPECTED DATA TO EBDR
 5009   016214  104031                          ERROR   +31                     ;BDR PATTERN NOT CORRECT
 5010   016216  052777  100000  164276  1$:     BIS     #EIR,@CSR               ;GO TO EIR
 5011   016224  012737  052525  002576          MOV     #52525,EBDR             ;MOVE EXPECTED DATA TO EBDR
 5012   016232  012777  125252  164264          MOV     #125252,@BDR            ;SET ALT 1'S AND 0'S TO BDR
 5013   016240  017737  164260  002564          MOV     @BDR,RBDR               ;MOVE RECEIVED DATA TO RBDR
 5014   016246  022737  052525  002564          CMP     #52525,RBDR             ;TEST FOR OLD PATTERN
 5015   016254  001404                          BEQ     2$                      ;BRANCH IF ORIGINAL PATTERN STILL THERE
 5016   016256  012737  052525  002576          MOV     #52525,EBDR             ;MOVE EXPECTED DATA TO EBDR
 5017   016264  104030                          ERROR   +30                     ;BDR SHOULD NOT HAVE BEEN LOADED WITH NEW PATTERN
 5018   016266  004737  003646          2$:     JSR     PC,CLENUP               ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
```

TEST #15 - DR11 INTERRUPTS WITH CPU AT LEVEL 3

```
                                        .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.
                                        ;*
                                        ;;*****************************************************************
016272                                  TST15:
016272  000004                                  SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
016274  012737  016330  001310                  MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
016302  032777  004000  163030                  BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
016310  001407                                  BEQ     1001$           ;&& BRANCH IF NOT
016312  104401  016320                          TYPE    ,1000$          ;&& TYPE: "T # 15"
016316  000404                                  BR      1001$           ;&& GET OVER ASCII
016320     124     040     043  1000$:          .ASCIZ  /T # 15/<CRLF>  ;&& THE ASCII MESSAGE
                                                .EVEN
016330                                  1001$:
5026 016330  012777  010000  164164  999$:    MOV     #MA,@CSR        ;SET MAINTENANCE BIT AND
5027 016336  005077  164160                    CLR     @CSR            ;CLEAR CSR TO DO AN INIT
5028 016342  012737  000140  177776            MOV     #LEVEL3,PSW     ;STATUS AT LEVEL 3
5029 016350  017737  164146  002560            MOV     @CSR,RCSR       ;MOVE CSR CONTENTS TO RCSR
5030 016356  105737  002560                    TSTB    RCSR            ;SEE IF READY BIT (BIT 7) IS SET
5031 016362  100406                            BMI     1$              ;BRANCH IF IT IS
5032 016364  012737  000200  002572            MOV     #RY,ECSR        ;SET THE BIT THAT SHOULD HAVE BEEN SET
5033 016372  004737  003670                    JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
5034 016376  104022                            ERROR   +22             ;READY OF CSR WAS NOT SET
5035 016400  017737  164122  002532  1$:       MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5036 016406  017737  164116  002534            MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
5037 016414  012777  016504  164104            MOV     #3$,@DRINV      ;SET UP INTERRUPT VECTOR
5038 016422  012777  010000  164072            MOV     #MA,@CSR        ;MAINT MODE
5039 016430  012737  001000  002660            MOV     #1000,TIME      ;SET THE TIME COUNTER
5040 016436  052777  000105  164056            BIS     #IE+F2+GO,@CSR  ;IE, FNCT2 AND GO
5041 016444  005337  002660          2$:       DEC     TIME            ;DECREMENT DOWN TO ZERO
5042 016450  001375                            BNE     2$              ;BRANCH IF NOT THERE YET
5043 016452  017737  164044  002560            MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5044 016460  013777  002532  164040            MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5045 016466  013777  002534  164034            MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5046 016474  104035                            ERROR   +35             ;DR11 FAILED TO INTERRUPT
5047 016476  005077  164020                    CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
5048 016502  000410                            BR      TST16           ;BRANCH TO THE NEXT TEST
5049 016504  062706  000004          3$:       ADD     #4,SP           ;CLEAN THE STACK AFTER THE INTERRUPT
5050 016510  013777  002532  164010            MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5051 016516  013777  002534  164004            MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
```

```
                                .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.
                                ;*
                                ;;*****************************************************************
016524                          TST16:
016524  000004                          SCOPE                           ;PROCESS LOOPING AND TEST NUMBER INCREMENT
016526  012737  016562  001310          MOV     #999$,$LPERR            ;SET LOOP ON ERROR TO 999$
016534  032777  004000  162576          BIT     #BIT11,@SWR             ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
016542  001407                          BEQ     1001$                   ;&& BRANCH IF NOT
016544  104401  016552                  TYPE    ,1000$                  ;&& TYPE: "T # 16"
016550  000404                          BR      1001$                   ;&& GET OVER ASCII
016552     124     040     043  1000$:  .ASCIZ  /T # 16/<CRLF>          ;&& THE ASCII MESSAGE
                                        .EVEN
016562                          1001$:
5059 016562  004737  003646     999$:   JSR     PC,CLENUP               ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
5060 016566  017737  163730  002560          MOV     @CSR,RCSR           ;MOVE CSR DATA TO RCSR
5061 016574  105737  002560                  TSTB    RCSR                ;CLEAR ALL BUT THE READY BIT (BIT 7)
5062 016600  100411                          BMI     1$                  ;BRANCH IF IT IS SET
5063 016602  012737  000200  002572          MOV     #RY,ECSR            ;MOVE EXPECTED DATA TO ECSR
5064 016610  004737  003670                  JSR     PC,CHKCAB           ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
5065 016614  052737  000200  002572          BIS     #RY,ECSR            ;SET THE BIT THAT SHOULD HAVE BEEN SET
5066 016622  104022                          ERROR   +22                 ;READY OF CSR WAS NOT SET
5067 016624  017737  163676  002532  1$:     MOV     @DRINV,SDRINV       ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5068 016632  017737  163672  002534          MOV     @DRVS,SDRVS         ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
5069 016640  012777  016726  163660          MOV     #3$,@DRINV          ;SET UP INT VECTOR
5070 016646  012777  000340  163654          MOV     #LEVEL7,@DRVS
5071 016654  012737  001000  002660          MOV     #1000,TIME          ;SET TIME DELAY COUNTER
5072 016662  012777  010000  163632          MOV     #MA,@CSR            ;MAINT MODE
5073 016670  052777  000105  163624          BIS     #IE+F2+GO,@CSR      ;IE, FNCT2 AND GO
5074 016676  005337  002660          2$:     DEC     TIME                ;DECREMENT UNTIL WE GET TO ZERO
5075 016702  001375                          BNE     2$                  ;BRANCH BACK IF NOT ZERO YET
5076 016704  005077  163612                  CLR     @CSR                ;CLEAR THE CSR TO DO AN INIT
5077 016710  013777  002532  163610          MOV     SDRINV,@DRINV       ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5078 016716  013777  002534  163604          MOV     SDRVS,@DRVS         ;RESTORE LOCATION USED AS THE INTERRUPT PS
5079 016724  000416                          BR      TST17               ;BRANCH TO THE NEXT TEST
5080 016726  062706  000004          3$:     ADD     #4,SP               ;RESTORE STACK
5081 016732  017737  163564  002560          MOV     @CSR,RCSR           ;MOVE RECEIVED DATA TO RCSR
5082 016740  005077  163556                  CLR     @CSR                ;CLEAR IE
5083 016744  013777  002532  163554          MOV     SDRINV,@DRINV       ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5084 016752  013777  002534  163550          MOV     SDRVS,@DRVS         ;RESTORE LOCATION USED AS THE INTERRUPT PS
5085 016760  104036                          ERROR   +36                 ;DR11 INTERRUPTED, BUT IT SHOULDN'T HAVE
```

```
5092                                    .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.
                                        ;*
                                        ;;**************************************************************
     016762                             TST17:
     016762  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     016764  012737  017074  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
     016772  032777  004000  162340             BIT     #BIT11,@SWR     ;@@ TEST TO SEE IF TEST NUMBER TRACER ENABLED
     017000  001407                             BEQ     1001$           ;@@ BRANCH IF NOT
     017002  104401  017010                     TYPE    ,1000$          ;@@ TYPE: "T # 17"
     017006  000404                             BR      1001$           ;@@ GET OVER ASCII
     017010     124     040     043  1000$:    .ASCIZ  /T # 17/<CRLF>  ;@@ THE ASCII MESSAGE
                                             .EVEN
     017020                             1001$:
5093 017020  004737  003646                     JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
5094 017024  013737  002720  001362             MOV     DDW,$TMP1       ;MOVE DEVICE DESCRIPTOR WORD TO $TMP1
5095 017032  006237  001362                     ASR     $TMP1           ;SHIFT THE LEVEL TO THE RIGHT 5 PLACES
5096 017036  006237  001362                     ASR     $TMP1
5097 017042  006237  001362                     ASR     $TMP1
5098 017046  006237  001362                     ASR     $TMP1
5099 017052  006237  001362                     ASR     $TMP1
5100 017056  042737  177770  001362             BIC     #177770,$TMP1   ;CLEAR ALL BUT THE PRIORITY
5101 017064  012700  000003             1$:     MOV     #3,R0           ;DO 3 PRIORITY LEVELS
5102 017070  012701  004732                     MOV     #LEVELS,R1      ;MOVE ADDRESS OF CPU PRIORITIES TO R1
5103 017074  012777  010000  163420     999$:   MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
5104 017102  005077  163414                     CLR     @CSR            ;CLEAR TO DO AN INIT
5105 017106  011137  177776                     MOV     (R1),PSW        ;PUT PRIORITY INTO PSW
5106 017112  017737  163404  002560             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5107 017120  012737  000200  002572             MOV     #RY,ECSR        ;MOVE READY BIT TO ECSR
5108 017126  004737  003670                     JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
5109 017132  023737  002560  002572             CMP     RCSR,ECSR       ;SEE IF RECEIVED DATA MATCHES EXPECTED
5110 017140  001412                             BEQ     2$              ;BRANCH IF OK
5111 017142  012737  017064  001310             MOV     #1$,$LPERR      ;SET UP FOR POSSIBLE LOOP ON ERROR FOR THIS ERROR ONLY
5112 017150  012737  000200  002572             MOV     #RY,ECSR        ;MOVE EXPECTED DATA TO ECSR
5113 017156  104022                             ERROR   +22             ;READY OF CSR WAS NOT SET
5114 017160  012737  017074  001310             MOV     #999$,$LPERR    ;RETURN ORIGINAL LOOP ON ERROR ADDRESS - DID NOT LOOP
5115 017166  017737  163334  002532     2$:     MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5116 017174  017737  163330  002534             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
5117 017202  012777  017314  163316             MOV     #4$,@DRINV      ;SET UP INTERRUPT VECTOR
5118 017210  012777  000340  163312             MOV     #LEVEL7,@DRVS   ;SET UP INTERRUPT PS
5119 017216  012777  010000  163276             MOV     #MA,@CSR        ;MAINT MODE
5120 017224  012737  000400  002660             MOV     #400,TIME       ;SET DELAY COUNTER
5121 017232  052777  000105  163262             BIS     #IE+F2+GO,@CSR  ;IE, FNCT2 AND GO
5122 017240  005337  002660             3$:     DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
5123 017244  00 375                            BNE     3$              ;BRANCH BACK IF NOT ZERO YET
5124 017246  005077  163250                     CLR     @CSR            ;CLEAR CSR TO DO AN INIT
5125 017252  013777  002532  163246             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5126 017260  013777  002534  163242             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5127 017266  013737  177776  002540             MOV     PSW,LEVEL       ;SAVE OLD STATUS LEVEL
5128 017274  005721                             TST     (R1)+           ;INCREMENT R1 TO POINT TO NEXT PRIORITY LEVEL
5129 017276  005300                             DEC     R0              ;DECREMENT LOOP COUNTER AND
5130 017300  001275                             BNE     999$            ;BRANCH BACK FOR ANOTHER TRY IF NOT DONE
5131 017302  104053                             ERROR   +53             ;DR11 FAILED TO INTERRUPT
```

TEST #17 - DR11 INTERRUPTS AT CORRECT BR LEVEL

```
5132 017304  013737  002552  002540          MOV     DRLEV,LEVEL     ;SET LEVEL TO CONTAIN THE ANTICIPATED LEVEL
5133 017312  000422                          BR      TST20           ;BRANCH TO THE NEXT TEST
5134 017314  062706  000004          4$:     ADD     #4,SP           ;RESTORE STACK
5135 017320  005077  163176                  CLR     @CSR            ;CLEAR IE
5136 017324  013777  002532  163174          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5137 017332  013777  002534  163170          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5138 017340  042737  177437  002540          BIC     #177437,LEVEL   ;CLEAR ALL BITS BUT THE BR LEVEL
5139 017346  023737  002540  002552          CMP     LEVEL,DRLEV     ;SEE IF LEVEL INTERRUPTED MATCHES EXPECTED
5140 017354  001401                          BEQ     TST20           ;BRANCH AROUND ERROR CALL IF IT IS AS EXPECTED
5141 017356  104052                          ERROR   +52             ;DR11 INTERRUPTED AT WRONG LEVEL
```

```
5148                                    .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.
                                        ;*
                                        ;;*****************************************************************
      017360                            TST20:
      017360  000004                            SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
      017362  012737  017416  001310            MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
      017370  032777  004000  161742            BIT     #BIT11,@SWR     ;@@ TEST TO SEE IF TEST NUMBER TRACER ENABLED
      017376  001407                            BEQ     1001$           ;@@ BRANCH IF NOT
      017400  104401  017406                    TYPE    ,1000$          ;@@ TYPE: 'T # 20'
      017404  000404                            BR      1001$           ;@@ GET OVER ASCII
      017406     124     C40     043  1000$:    .ASCIZ  /T # 20/<CRLF>  ;@@ THE ASCII MESSAGE
                                                .EVEN
      017416                            1001$:
5149  017416  004737  003646            999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
5150  017422  017737  163100  002532            MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5151  017430  017737  163074  002534            MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
5152  017436  012777  017540  163062            MOV     #2$,@DRINV      ;INTERRUPT VECTOR TO 3$
5153  017444  012777  000140  163056            MOV     #LEVEL3,@DRVS   ;INTERRUPT STATUS TO LEVEL 3
5154  017452  005037  177776                    CLR     PSW             ;LET THE DR11 INTERRUPT
5155  017456  012737  001000  002660            MOV     #1000,TIME      ;MOVE DELAY COUNTER TO LOCATION
5156  017464  012777  010101  163030            MOV     #MA+IE+GO,@CSR  ;SET MAINT, IE AND GO
5157  017472  052777  020004  163022            BIS     #AT+F2,@CSR     ;SET ATTN AND FNCT2
5158  017500  005337  002660            1$:     DEC     TIME            ;DECREMENT UNTIL WE REACH ZERO
5159  017504  001375                            BNE     1$              ;BRANCH IF NOT ZERO YET
5160  017506  013777  002532  163012            MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5161  017514  013777  002534  163006            MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5162  017522  017737  162774  002560            MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5163  017530  104035                            ERROR   +35             ;DR11 FAILED TO INTERRUPT
5164  017532  005077  162764                    CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
5165  017536  000512                            BR      TST21           ;BRANCH TO THE NEXT TEST
5166  017540  062706  000004            2$:     ADD     #4,SP           ;READJUST STACK AFTER THE INTERRUPT
5167  017544  013777  002532  162754            MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5168  017552  013777  002534  162750            MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5169  017560  017737  162736  002560            MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5170  017566  100407                            BMI     3$              ;BRANCH IF ERROR IS SET
5171  017570  013737  002560  002572            MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
5172  017576  052737  100000  002572            BIS     #ER,ECSR        ;SET THE BIT THAT SHOULD HAVE BEEN SET
5173  017604  104037                            ERROR   +37             ;ERROR BIT SHOULD NOT BE CLEAR
5174  017606  017737  162714  002532  3$:       MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5175  017614  017737  162710  002534            MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
5176  017622  012777  017716  162676            MOV     #5$,@DRINV      ;INTERRUPT VECTOR TO 6$
5177  017630  005077  162664                    CLR     @BAR            ;PREVENT CAUSING ANOTHER ERROR
5178  017634  012777  177777  162654            MOV     #-1,@WCR        ;SET-UP WCR
5179  017642  012737  001000  002660            MOV     #1000,TIME      ;LOAD 1000 IN LOCATION TIME FOR WAIT LOOP
5180  017650  052777  000001  162644            BIS     #GO,@CSR        ;SET 'GO' IN CSR
5181  017656  005237  002660            4$:     INC     TIME            ;DELAY - WAIT FOR INTERRUPT
5182  017662  001375                            BNE     4$              ;BRANCH BACK IF NOT ZERO
5183  017664  013777  002532  162634            MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5184  017672  013777  002534  162630            MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5185  017700  017737  162616  002560            MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5186  017706  104035                            ERROR   +35             ;DR11 FAILED TO INTERRUPT
5187  017710  005077  162606                    CLR     @CSR            ;CLEAR CSR TO DO AN INIT
```

TEST #20 - A GO WITHOJT CLEARING ERROR CAUSES INTRPT

```
 5188 017714  000423                          BR      TST21           ;BRANCH TO THE NEXT TEST
 5189 017716  062706  000004          5$:     ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
 5190 017722  013777  002532  162576          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
 5191 017730  013777  002534  162572          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
 5192 017736  017737  162560  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR - IS ERROR CLEAR
 5193 017744  100007                          BPL     TST21           ;BRANCH TO NEXT TEST IF IT IS
 5194 017746  013737  002560  002572          MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
 5195 017754  042737  100000  002572          BIC     #ER,ECSR        ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
 5196 017762  104021                          ERROR   +21             ;ERROR BIT SHOULD HAVE BEEN CLEAR
```

```
                                        .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.
                                        ;*
                                        ;;*****************************************************************
      017764                            TST21:
      017764  000004                            SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
      017766  012737  020036  001310            MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
      017774  032777  004000  161336            BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
      020002  001407                            BEQ     1001$           ;&& BRANCH IF NOT
      020004  104401  020012                    TYPE    ,1000$          ;&& TYPE: "T # 21"
      020010  000404                            BR      1001$           ;&& GET OVER ASCII
      020012     124     040     043    1000$:  .ASCIZ  /T # 21/<CRLF>  ;&& THE ASCII MESSAGE
                                                .EVEN
      020022                            1001$:
 5204 020022  012737  000016  001364            MOV     #16,$TMP2       ;SET UP FUNCTION COUNT COMPARE
 5205 020030  012737  177771  001360            MOV     #-7,$TMP0       ;SET UP WCR LOAD VARIABLE
 5206 020036  004737  003646            999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
 5207 020042  013777  002616  162450            MOV     INBUF,@BAR      ;SET-UP BAR
 5208 020050  017737  162452  002532    1$:     MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
 5209 020056  017737  162446  002534            MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
 5210 020064  012777  020174  162434            MOV     #3$,@DRINV      ;INTERRUPT VECTOR
 5211 020072  013777  002540  162430            MOV     LEVEL,@DRVS     ;INTERRUPT VECTOR PRIORITY TO LEVEL OF DEVICE
 5212 020100  013777  001360  162410            MOV     $TMP0,@WCR      ;SET UP FOR NUMBER OF TRANSFERS IN $TMP0
 5213 020106  005037  177776                    CLR     PSW             ;LET THE DR11 INTERRUPT
 5214 020112  012777  010000  162402            MOV     #MA,@CSR        ;MAINT MODE
 5215 020120  012737  001000  002660            MOV     #1000,TIME      ;MOVE WAIT COUNTER TO LOCATION TIME
 5216 020126  052777  000501  162366            BIS     #IE+CY+GO,@CSR  ;IE, CYCLE & GO
 5217 020134  005337  002660            2$:     DEC     TIME            ;DECREMENT UNTIL ZERO
 5218 020140  001375                            BNE     2$              ;BRANCH BACK IF NOT
 5219 020142  017737  162354  002560            MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
 5220 020150  013777  002532  162350            MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
 5221 020156  013777  002534  162344            MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
 5222 020164  104035                            ERROR   +35             ;DR11 FAILED TO INTERRUPT
 5223 020166  005077  162330                    CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
 5224 020172  000442                            BR      TST22           ;BRANCH TO NEXT TEST
 5225 020174  062706  000004            3$:     ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
 5226 020200  013777  002532  162320            MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
 5227 020206  013777  002534  162314            MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
 5228 020214  017737  162302  002560            MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
 5229 020222  013701  002560                    MOV     RCSR,R1         ;MOVE RECEIVED DATA TO R1 ALSO AND
 5230 020226  042701  177761                    BIC     #CFNC,R1        ;CLEAR ALL BUT THE FUNCTION BITS
 5231 020232  020137  001364                    CMP     R1,$TMP2        ;SEE IF FUNCTION BIT(S) HAD INCREMENTED PROPERLY
 5232 020236  001412                            BEQ     4$              ;BRANCH IF THEY HAD
 5233 020240  013737  002560  002572            MOV     RCSR,ECSR       ;MOVE RECEIVED DATA TO EXPECTED LOCATION
 5234 020246  042737  000016  002572            BIC     #FNC,ECSR       ;CLEAR THE FUNCTION BIT(S) THAT WERE THERE AND
 5235 020254  053737  001364  002572            BIS     $TMP2,ECSR      ;PUT FUNCTION BIT(S) EXPECTED IN THEIR PLACE
 5236 020262  104212                            ERROR   +212            ;FUNCTION BITS DIDN'T INCREMENT IN MAINT MODE
 5237 020264  005237  001360            4$:     INC     $TMP0           ;ADJUST WCR LOAD LOCATION
 5238 020270  162737  000002  001364            SUB     #2,$TMP2        ;SUBTRACT 2 FROM FUNCTION COUNT TEST LOCATION
 5239 020276  001264                            BNE     1$              ;BRANCH BACK FOR ANOTHER TRY
```

```
5245                                    .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.
                                        ;*
                                        ;;*****************************************************************
     020300                             TST22:
     020300  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     020302  012737  020336  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
     020310  032777  004000  161022             BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     020316  001407                             BEQ     1001$           ;&& BRANCH IF NOT
     020320  104401  020326                     TYPE    ,1000$          ;&& TYPE: "T # 22"
     020324  000404                             BR      1001$           ;&& GET OVER ASCII
     020326     124     040     043  1000$:     .ASCIZ  /T # 22/<CRLF>  ;&& THE ASCII MESSAGE
                                                .EVEN
     020336                             1001$:
5246 020336  005077  162160             999$:   CLR     @CSR            ;FORCE ACCESS TO CSR
5247 020342  012737  000010  002622             MOV     #10,BUFLEN      ;BUFLEN=10
5248 020350  013737  002622  002630             MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
5249 020356  005437  002630                     NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
5250 020362  004737  003302                     JSR     PC,LODBUF       ;LOAD IN BUFFER WITH INCREMENTING PATTERN
5251 020366  004737  003330                     JSR     PC,CHKBFF       ;LOAD CHECK BUFFER WITH MODIFIED INCREMENTING PATTERN
5252 020372  013777  002630  162116             MOV     WCLEN,@WCR      ;SET UP WCR
5253 020400  013777  002616  162112             MOV     INBUF,@BAR      ;SET UP BAR
5254 020406  012777  177777  162110             MOV     #-1,@BDR        ;MAINT AIDE
5255 020414  017737  162106  002532             MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5256 020422  017737  162102  002534             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
5257 020430  012777  020532  162070             MOV     #2$,@DRINV      ;INTERRUPT VECTOR
5258 020436  013777  002540  162064             MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
5259 020444  005037  177776                     CLR     PSW             ;LET DR11 INTERRUPT
5260 020450  012777  010000  162044             MOV     #MA,@CSR        ;MAINT MODE
5261 020456  052777  000501  162036             BIS     #IE+CY+GO,@CSR  ;IE, CYCLE & GO
5262 020464  012737  001000  002660             MOV     #1000,TIME      ;SET LOOP COUNTER FOR WAIT
5263 020472  005337  002660          1$:        DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
5264 020476  001375                             BNE     1$              ;BRANCH BACK IF NOT ZERO
5265 020500  013777  002532  162020             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5266 020506  013777  002534  162014             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5267 020514  017737  162002  002560             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5268 020522  104035                             ERROR   +35             ;DR11 FAILED TO INTERRUPT
5269 020524  005077  161772                     CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
5270 020530  000402                             BR      3$              ;BRANCH AROUND THE STACK CLEANUP
5271 020532  062706  000004          2$:        ADD     #4,SP           ;CLEAN UP STACK AFTER THE INTERRUPT
5272 020536  013777  002532  161762  3$:        MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5273 020544  013777  002534  161756             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5274 020552  004737  003356                     JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
5275 020556  104051                             ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
5276 020560  005037  002714                     CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
5277 020564  004737  003526                     JSR     PC,DATCHK       ;CHECK INBUF AFTER A MAINTENANCE MODE OPERATION
5278 020570  104201                             ERROR   +201            ;BUFFER DATA NOT CORRECT
5279 020572  004737  003624                     JSR     PC,DATCH2       ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATCHK
5280
```

```
5287                                    .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.
                                        ;*
                                        ;;*****************************************************************
     020576                             TST23:
     020576  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     020600  012737  020634  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
     020606  032777  004000  160524             BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     020614  001407                             BEQ     1001$           ;&& BRANCH IF NOT
     020616  104401  020624                     TYPE    ,1000$          ;&& TYPE: 'T # 23'
     020622  000404                             BR      1001$           ;&& GET OVER ASCII
     020624     124     040     043  1000$:    .ASCIZ  /T # 23/<CRLF>  ;&& THE ASCII MESSAGE
                                             .EVEN
     020634                             1001$:
5288 020634  004737  003646             999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
5289 020640  012737  001000  002660             MOV     #1000,TIME      ;SET DELAY
5290 020646  012737  000010  002622             MOV     #10,BUFLEN      ;BUFLEN=10
5291 020654  013737  002622  002630             MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
5292 020662  005437  002630                     NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
5293 020666  004737  003302                     JSR     PC,LODBUF       ;LOAD IN BUFFER WITH INCREMENTING PATTERN
5294 020672  004737  003330                     JSR     PC,CHKBFF       ;LOAD CHECK BUFFER WITH MODIFIED INCREMENTING PATTERN
5295 020676  013777  002630  161612             MOV     WCLEN,@WCR      ;SET UP WCR
5296 020704  013777  002616  161606             MOV     INBUF,@BAR      ;SET UP BAR
5297 020712  012777  177777  161604             MOV     #-1,@BDR        ;MAINT AIDE
5298 020720  017737  161602  002532             MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5299 020726  017737  161576  002534             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
5300 020734  012777  021034  161564             MOV     #2$,@DRINV      ;INTERRUPT VECTOR
5301 020742  013777  002540  161560             MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
5302 020750  012777  010000  161544             MOV     #MA,@CSR        ;MAINT MODE
5303 020756  052777  000501  161536             BIS     #IE+CY+GO,@CSR  ;IE, CYCLE & GO
5304 020764  005337  002660          1$:        DEC     TIME            ;WAIT FOR TRANSFERS TO COMPLETE
5305 020770  001375                             BNE     1$              ;BRANCH BACK IF WE ARE STILL WAITING
5306 020772  013777  002532  161526             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5307 021000  013777  002534  161522             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5308 021006  004737  003356                     JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
5309 021012  104051                             ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
5310 021014  005037  002714                     CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
5311 021020  004737  003526                     JSR     PC,DATCHK       ;CHECK INBUF AFTER A MAINTENANCE MODE OPERATION
5312 021024  104201                             ERROR   +201            ;BUFFER DATA NOT CORRECT
5313 021026  004737  003624                     JSR     PC,DATCH2       ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATCHK
5314 021032  000415                             BR      3$              ;BRANCH TO CONTINUE
5315 021034  062706  000004          2$:        ADD     #4,SP           ;CLEAN UP THE STACK FROM THIS INTERRUPT
5316 021040  013777  002532  161460             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5317 021046  013777  002534  161454             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5318 021054  017737  161442  002560             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5319 021062  104036                             ERROR   +36             ;DR11 INTERRUPTED, BUT IT SHOULDN'T HAVE
5320 021064  000523                             BR      TST24           ;BRANCH TO NEXT TEST
5321 021066  012777  010000  161426  3$:        MOV     #MA,@CSR        ;MAINT MODE
5322 021074  012737  001000  002660             MOV     #1000,TIME      ;SET TIME LOOP COUNTER
5323 021102  013777  002630  161406             MOV     WCLEN,@WCR      ;MOVE WCLEN TO WCR
5324 021110  013777  002616  161402             MOV     INBUF,@BAR      ;MOVE INBUF TO BAR
5325 021116  017737  161404  002532             MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5326 021124  017737  161400  002534             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
```

```
5327 021132  012777  021304  161366          MOV     #8$,@DRINV      ;SET UP INTERRUPT VECTOR
5328 021140  012777  010000  161354          MOV     #MA,@CSR        ;MAINT MODE
5329 021146  052777  000501  161346          BIS     #IE+CY+GO,@CSR  ;IE, CYCLE & GO
5330 021154  005337  002660            4$:   DEC     TIME            ;DECREMENT TO ZERO WHILE WAITING
5331 021160  001375                          BNE     4$              ;BRANCH BACK IF NOT ZERO
5332 021162  013777  002532  161336          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5333 021170  013777  002534  161332          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5334 021176  017737  161320  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5335 021204  022737  010700  002560          CMP     #10700,RCSR     ;SEE IF ONLY READY, MAINT, IE & CYCLE ARE SET
5336 021212  001404                          BEQ     5$              ;BRANCH IF THEY ARE
5337 021214  012737  010700  002572          MOV     #10700,ECSR     ;MOVE EXPECTED DATA TO ECSR
5338 021222  104040                          ERROR   +40             ;CSR IS WRONG
5339 021224  017737  161270  002566    5$:   MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
5340 021232  022777  177770  161256          CMP     #-10,@WCR       ;CHECK THAT NO TRANSFERS WERE MADE
5341 021240  001004                          BNE     6$              ;BRANCH TO ERROR IF THERE WERE
5342 021242  023737  002616  002566          CMP     INBUF,RBAR      ;CHECK THAT NO TRANSFERS WERE MADE
5343 021250  001412                          BEQ     7$              ;BRANCH AROUND ERROR IF NONE
5344 021252  017737  161240  002570    6$:   MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
5345 021260  012737  177770  002602          MOV     #-10,EWCR       ;MOVE EXPECTED DATA TO EWCR
5346 021266  013737  002616  002600          MOV     INBUF,EBAR      ;MOVE EXPECTED DATA TO EBAR
5347 021274  104041                          ERROR   +41             ;TRANSFERS SHOULD HAVE BEEN INHIBITED
5348 021276  005077  161220            7$:   CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
5349 021302  000414                          BR      TST24           ;BRANCH TO NEXT TEST
5350 021304  062706  000004            8$:   ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
5351 021310  013777  002532  161210          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5352 021316  013777  002534  161204          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5353 021324  017737  161172  002560          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5354 021332  104042                          ERROR   +42             ;DR11 SHOULD NOT HAVE INTERRUPTED A 2ND TIME
```

```
5360                                    .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.
                                        ;*
                                        ;;*****************************************************************
     021334                             TST24:
     021334  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     021336  012737  021372  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
     021344  032777  004000  157766             BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     021352  001407                             BEQ     1001$           ;&& BRANCH IF NOT
     021354  104401  021362                     TYPE    ,1000$          ;&& TYPE: "T # 24"
     021360  000404                             BR      1001$           ;&& GET OVER ASCII
     021362     124     040     043  1000$:     .ASCIZ  /T # 24/<CRLF>  ;&& THE ASCII MESSAGE
                                                .EVEN
     021372                             1001$:
5361 021372  004737  003646             999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
5362 021376  005077  161120                     CLR     @CSR            ;FORCE ACCESS TO CSR
5363 021402  012737  000200  002622             MOV     #200,BUFLEN     ;LENGTH OF BUFFER = 200
5364 021410  013737  002622  002630             MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
5365 021416  005437  002630                     NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
5366 021422  004737  003302                     JSR     PC,LODBUF       ;LOAD INBUF WITH INCREMENTING PATTERN
5367 021426  004737  003330                     JSR     PC,CHKBFF       ;LOAD CHKBUF WITH MODIFIED INCREMENTED PATTERN
5368 021432  013777  002630  161056             MOV     WCLEN,@WCR      ;SET UP WCR
5369 021440  013777  002616  161052             MOV     INBUF,@BAR      ;SET UP BAR
5370 021446  012777  177777  161050             MOV     #-1,@BDR        ;MAINT AIDE
5371 021454  017737  161046  002532             MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5372 021462  017737  161042  002534             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
5373 021470  012777  021572  161030             MOV     #2$,@DRINV      ;INT VECTOR
5374 021476  013777  002540  161024             MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
5375 021504  005037  177776                     CLR     PSW             ;LET THE DR11 INTERRUPT
5376 021510  012777  010000  161004             MOV     #MA,@CSR        ;MAINT MODE
5377 021516  012737  001000  002660             MOV     #1000,TIME      ;SET WAIT LOOP COUNTER
5378 021524  052777  000501  160770             BIS     #IE+CY+GO,@CSR  ;IE, CYCLE & GO
5379 021532  005337  002660          1$:        DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
5380 021536  001375                             BNE     1$              ;BRANCH BACK IF NOT ZERO
5381 021540  017737  160756  002560             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5382 021546  013777  002532  160752             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5383 021554  013777  002534  160746             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5384 021562  104043                             ERROR   +43             ;EXPECTED INTERRUPT DID NOT OCCUR
5385 021564  005077  160732                     CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
5386 021570  000402                             BR      3$              ;BRANCH AROUND THE STACK CLEANUP
5387 021572  062706  000004          2$:        ADD     #4,SP           ;CLEAN UP THE STACK AFTER INTERRUPT
5388 021576  013777  002532  160722  3$:        MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5389 021604  013777  002534  160716             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5390 021612  004737  003356                     JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
5391 021616  104051                             ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORRECT
5392 021620  005037  002714                     CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
5393 021624  004737  003526                     JSR     PC,DATCHK       ;CHECK INBUF AFTER A MAINTENANCE MODE OPERATION
5394 021630  104201                             ERROR   +201            ;BUFFER DATA NOT CORRECT
5395 021632  004737  003624                     JSR     PC,DATCH2       ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATCHK
```

```
                                .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.
                                ;*
                                ;*****************************************************************
     021636                     TST25:
     021636  000004                     SCOPE                           ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     021640  012737  021674  001310     MOV     #999$,$LPERR            ;SET LOOP ON ERROR TO 999$
     021646  032777  004000  157464     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 # 25"
     021662  000404                     BR      1001$                   ;&& GET OVER ASCII
     021664     124     040     043  1000$:  .ASCIZ  /T # 25/<CRLF>      ;&& THE ASCII MESSAGE
                                        .EVEN
     021674                     1001$:
5403 021674  004737  003646     999$:   JSR     PC,CLENUP               ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
5404 021700  005077  160616             CLR     @CSR                    ;FORCE ACCESS TO CSR
5405 021704  012777  177776  160604     MOV     #-2,@WCR                ;SET UP WCR
5406 021712  013777  002614  160600     MOV     DIOMEM,@BAR             ;SET UP BAR
5407 021720  017737  160602  002532     MOV     @DRINV,SDRINV           ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5408 021726  017737  160576  002534     MOV     @DRVS,SDRVS             ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
5409 021734  012777  022044  160564     MOV     #2$,@DRINV              ;INTERRUPT VECTOR TO 3$
5410 021742  013777  002540  160560     MOV     LEVEL,@DRVS             ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
5411 021750  005037  177776             CLR     PSW                     ;LET THE DR11 INTERRUPT
5412 021754  012777  010000  160540     MOV     #MA,@CSR                ;MAINT MODE
5413 021762  052777  000062  160532     BIS     #F1+X6+X7,@CSR          ;SET FNCT1, XBA16, AND XBA17
5414 021770  052777  000501  160524     BIS     #IE+CY+GO,@CSR          ;SET IE, CYCLE, AND GO
5415 021776  012737  001000  002660     MOV     #1000,TIME              ;SET DELAY COUNTER
5416 022004  005337  002660        1$:  DEC     TIME                    ;DECREMENT UNTIL ZERO
5417 022010  001375                     BNE     1$                      ;BRANCH BACK IF NOT
5418 022012  013777  002532  160506     MOV     SDRINV,@DRINV           ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5419 022020  013777  002534  160502     MOV     SDRVS,@DRVS             ;RESTORE LOCATION USED AS THE INTERRUPT PS
5420 022026  017737  160470  002560     MOV     @CSR,RCSR               ;MOVE RECEIVED DATA TO RCSR
5421 022034  104035                     ERROR   +35                     ;DR11 FAILED TO INTERRUPT
5422 022036  005077  160460             CLR     @CSR                    ;CLEAR THE CSR TO DO AN INIT
5423 022042  000431                     BR      TST26                   ;BRANCH TO THE NEXT TEST
5424 022044  062706  000004        2$:  ADD     #4,SP                   ;RESTORE THE STACK
5425 022050  013777  002532  160450     MOV     SDRINV,@DRINV           ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5426 022056  013777  002534  160444     MOV     SDRVS,@DRVS             ;RESTORE LOCATION USED AS THE INTERRUPT PS
5427 022064  017737  160432  002560     MOV     @CSR,RCSR               ;MOVE CSR DATA TO RCSR
5428 022072  013701  002560             MOV     RCSR,R1                 ;MOVE DATA TO R1 FOR CHECKING
5429 022076  042701  037577             BIC     #37577,R1               ;CLEAR ALL BUT ERROR, NEX AND READY BITS
5430 022102  022701  140200             CMP     #ER+NX+RY,R1            ;SEE IF ALL THESE BITS ARE SET
5431 022106  001407                     BEQ     TST26                   ;BRANCH TO THE NEXT TEST IF THEY ARE ALL SET
5432 022110  013737  002560  002572     MOV     RCSR,ECSR               ;MOVE EXPECTED DATA TO ECSR
5433 022116  052737  140200  002572     BIS     #ER+NX+RY,ECSR          ;SET THE BIT THAT SHOULD HAVE BEEN SET
5434 022124  104040                     ERROR   +40                     ;CSR IS WRONG
```

TEST #26 - CROSSING 32K DOESN'T CAUSE BAOF OR FORCE ERROR

```
5441                                    .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.
                                        ;*
                                        ;;*****************************************************************
     022126                             TST26:
     022126  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     022130  012737  022164  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
     022136  032777  004000  157174             BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     022144  001407                             BEQ     1001$           ;&& BRANCH IF NOT
     022146  104401  022154                     TYPE    ,1000$          ;&& TYPE: "T # 26"
     022152  000404                             BR      1001$           ;&& GET OVER ASCII
     022154     124     040     043  1000$:    .ASCIZ  /T # 26/<CRLF>  ;&& THE ASCII MESSAGE
                                             .EVEN
     022164                             1001$:
5442 022164  004737  003646             999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
5443 022170  012777  177760  160320             MOV     #-20,@WCR       ;SET UP WCR
5444 022176  012777  177776  160314             MOV     #-2,@BAR        ;SET UP BAR FOR PROC STATUS ADDRESS
5445 022204  017737  160316  002532             MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5446 022212  017737  160312  002534             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
5447 022220  012777  022316  160300             MOV     #2$,@DRINV      ;INTERRUPT VECTOR TO 3$
5448 022226  013777  002540  160274             MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
5449 022234  012737  001000  002660             MOV     #1000,TIME      ;SET WAIT LOOP COUNTER
5450 022242  005037  177776                     CLR     PSW             ;LET THE DR11 INTERRUPT
5451 022246  012777  010000  160246             MOV     #MA,@CSR        ;MAINT MODE
5452 022254  052777  000563  160240             BIS     #563,@CSR       ;CYCLE, IE, FNCT1, XBA17, XBA16, AND GO TO CSR
5453 022262  005337  002660          1$:        DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
5454 022266  001375                             BNE     1$              ;BRANCH BACK IF NOT ZERO
5455 022270  013777  002532  160230             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5456 022276  013777  002534  160224             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5457 022304  017737  160212  002560             MOV     @CSR,RCSR       ;MOVE CSR CONTENTS TO RCSR
5458 022312  104035                             ERROR   +35             ;DR11 FAILED TO INTERRUPT
5459 022314  000433                             BR      TST27           ;BRANCH TO NEXT TEST
5460 022316  062706  000004          2$:        ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
5461 022322  013777  002532  160176             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5462 022330  013777  002534  160172             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5463 022336  017737  160160  002560             MOV     @CSR,RCSR       ;MOVE CSR CONTENTS TO RCSR
5464 022344  013701  002560                     MOV     RCSR,R1         ;MOVE CONTENTS TO R1 FOR TESTING
5465 022350  042701  037577                     BIC     #37577,R1       ;CLEAR ALL BUT THE ERROR AND READY BITS
5466 022354  022701  140200                     CMP     #ER+RY+NX,R1    ;SEE IF ERROR, READY AND NEX ARE SET
5467 022360  001411                             BEQ     TST27           ;BRANCH TO NEXT TEST IF THEY ARE
5468 022362  013737  002560  002572             MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
5469 022370  052737  140200  002572             BIS     #ER+RY+NX,ECSR  ;SET THE BITS THAT SHOULD HAVE BEEN SET
5470 022376  104040                             ERROR   +40             ;CSR IS WRONG
5471 022400  005077  160116                     CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
```

TEST #27 - CHECK ACTUAL POSITION OF 2-N BURST SWITCH

```
5478                                    .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.
                                        ;*
                                        ;;*****************************************************************
     022404                             TST27:
     022404  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     022406  012737  022452  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
     022414  032777  004000  156716             BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     022422  001407                             BEQ     1001$           ;&& BRANCH IF NOT
     022424  104401  022432                     TYPE    ,1000$          ;&& TYPE: "T # 27"
     022430  000404                             BR      1001$           ;&& GET OVER ASCII
     022432     124     040     043  1000$:     .ASCIZ  /T # 27/<CRLF>  ;&& THE ASCII MESSAGE
                                                .EVEN
     022442                             1001$:
5479 022442  032737  000001  002610             BIT     #BIT0,BORW      ;TESTING A "B" OR A "W"?
5480 022450  001456                             BEQ     INOOUT          ;BRANCH TO INOOUT IF DR11-B
5481 022452  004737  003646             999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
5482 022456  012777  100000  160036             MOV     #EIR,@CSR       ;GO TO EIR MODE
5483 022464  017737  160032  002562             MOV     @CSR,REIR       ;MOVE EIR DATA TO REIR
5484 022472  005077  160024                     CLR     @CSR            ;GO BACK TO CSR
5485 022476  013701  002562                     MOV     REIR,R1         ;MOVE DATA TO R1 ALSO
5486 022502  000301                             SWAB    R1              ;GET BIT 8 INTO BIT 0 BY SWAPPING BYTES
5487 022504  006301                             ASL     R1              ;MOVE BIT 0 INTO BIT 1
5488 022506  042701  177775                     BIC     #CBIT1,R1       ;CLEAR ALL BUT BIT 1
5489 022512  013702  002720                     MOV     DDW,R2          ;PUT DEVICE DESCRIPTOR WORD IN R2
5490 022516  042702  177775                     BIC     #CBIT1,R2       ;CLEAR ALL BUT BIT 1
5491 022522  001402                             BEQ     1$              ;BRANCH IF IT IS CLEAR
5492 022524  005002                             CLR     R2              ;CLEAR THE BIT
5493 022526  000402                             BR      2$              ;GO TEST THE BIT
5494 022530  012702  000002             1$:     MOV     #BIT1,R2        ;SET THE BIT
5495 022534  020102                     2$:     CMP     R1,R2           ;SEE IF RECEIVED MATCHES EXPECTED
5496 022536  001017                             BNE     5$              ;BRANCH TO CHECK FOR LOOP ON TEST
5497 022540  013737  002562  002574             MOV     REIR,EEIR       ;MOVE EXPECTED DATA TO EEIR
5498 022546  032737  000400  002574             BIT     #BIT8,EEIR      ;TEST STATE OF BIT 8
5499 022554  001404                             BEQ     3$              ;BRANCH IF IT IS CLEAR
5500 022556  042737  000400  002574             BIC     #BIT8,EEIR      ;REVERSE STATE - EXPECTED CLEAR
5501 022564  000403                             BR      4$              ;GO CALL ERROR
5502 022566  052737  000400  002574     3$:     BIS     #BIT8,EEIR      ;REVERSE STATE - EXPECTED SET
5503 022574  104054                     4$:     ERROR   +54             ;2-N CYCLE BURST SWITCH IN WRONG POSITION
5504 022576  032777  040400  156534     5$:     BIT     #BIT14+BIT8,@SWR;SEE IF WE SHOULD LOOP ON THIS TEST
5505 022604  001322                             BNE     999$            ;BRANCH BACK IF SO
```

```
                                        .SBTTL  CODE TO CHECK CABLE STATUS FOR EXECUTION OF CABLE TESTS
                                ;               CABLE MODE TESTING (WRAP-AROUND CABLE IN USER SLOTS)

                                ;       TESTS 30 THRU 37 ARE PERFORMED IF BIT 2 OF DEVICE DESCRIPTOR WORD IS
                                ;       SET, INDICATING CABLE IS IN.

022606  032737  000004  002720  INOOUT: BIT     #BIT2,DDW       ;SEE IF CABLE IS IN
022614  001002                          BNE     TST30           ;BRANCH TO NEXT TEST IF CABLE IS IN
022616  000137  025622                  JMP     ENDEV           ;JUMP TO ENDEV - TESTS ARE NOT TO BE DONE
```

```
5522                                    .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'.
                                        ;*
                                        ;*****************************************************************
022622                                  TST30:
022622  000004                                  SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
022624  012737  022706  000310                  MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
022632  032777  004000  156500                  BIT     #BIT11,@SWR     ;@@ TEST TO SEE IF TEST NUMBER TRACER ENABLED
022640  001407                                  BEQ     1001$           ;@@ BRANCH IF NOT
022642  104401  022650                          TYPE    ,1000$          ;@@ TYPE: "T # 30"
022646  000404                                  BR      1001$           ;@@ GET OVER ASCII
022650     124     040     043  1000$:         .ASCIZ  /T # 30/<CRLF>  ;@@ THE ASCII MESSAGE
                                                .EVEN
022660                                  1001$:
5523 022660  004737  003646                    JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
5524 022664  005037  002714                    CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +202
5525 022670  012700  000002                    MOV     #2,R0           ;DO 2 SETS OF 200 PATTERNS
5526 022674  012701  007220                    MOV     #MAICLR,R1      ;MOVE ADDRESS OF EXPECTED PATTERNS TO R1
5527 022700  005002                            CLR     R2              ;START WITH PATTERN ZERO
5528 022702  012703  000200            1$:     MOV     #200,R3         ;MOVE 200 TO THE LOOP COUNTER
5529 022706  052777  010000  157606    999$:   BIS     #MA,@CSR        ;SET MAINTENANCE AND
5530 022714  005077  157602                    CLR     @CSR            ;CLEAR TO DO AN INIT
5531 022720  017737  157576  002560            MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5532 022726  022737  000200  002560            CMP     #RY,RCSR        ;MAKE SURE READY BIT IS THE ONLY BIT SET
5533 022734  001404                            BEQ     2$              ;BRANCH IF SO
5534 022736  012737  000200  002572            MOV     #RY,ECSR        ;MOVE EXPECTED DATA TO ECSR
5535 022744  104032                            ERROR   +32             ;READY IS NOT THE ONLY BIT SET
5536 022746  012777  177777  157542    2$:     MOV     #-1,@WCR        ;MOVE 1 WORD COUNT TO WCR IN CASE OF IE ENABLED
5537 022754  012777  056104  157536            MOV     #NOCARE,@BAR    ;MOVE A NOT-CARE ADDRESS TO BAR FOR SAME REASON
5538 022762  010277  157534                    MOV     R2,@CSR         ;SET THE PARTICULAR FUNCTION BITS IN CSR
5539 022766  017737  157530  002560            MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5540 022774  011137  002572                    MOV     (R1),ECSR       ;MOVE EXPECTED DATA TO ECSR
5541 023000  023737  002572  002560            CMP     ECSR,RCSR       ;COMPARE EXPECTED WITH RECEIVED
5542 023006  001427                            BEQ     6$              ;BRANCH IF OK
5543 023010  012737  000401  023020            MOV     #CY+GO,3$       ;REESTABLISH BIT PATTERN
5544 023016  040227                            BIC     R2,(PC)+        ;SEE IF BOTH CYCLE AND GO WERE SET
5545 023020  000401                    3$:     .WORD   CY+GO           ;LOCATION TO HOLD BOTH CYCLE AND GO BITS
5546 023022  001016                            BNE     5$              ;BRANCH TO ERROR ONLY IF EITHER OR BOTH BITS WERE CLEAR
5547 023024  005737  002712                    TST     MEMGMT          ;SEE IF MEMORY MANAGEMENT IS OUT THERE
5548 023030  001404                            BEQ     4$              ;BRANCH IF NOT
5549 023032  032737  000060  002572            BIT     #X6+X7,ECSR     ;SEE IF EITHER XBA16 OR XBA17 ARE SET
5550 023040  001407                            BEQ     5$              ;BRANCH TO ERROR IF BOTH ARE CLEAR
5551 023042  052737  140000  002572    4$:     BIS     #ER+NX,ECSR     ;SET THE ERROR AND NEX BITS - EXPECT THEM TO SET
5552 023050  023737  002572  002560            CMP     ECSR,RCSR       ;NOW SEE IF DATA MATCHES
5553 023056  001403                            BEQ     6$              ;BRANCH AROUND ERROR IF IT DOES
5554 023060  010237  002536            5$:     MOV     R2,BUT          ;MOVE THE BITS SET INTO CSR TO THE BUT LOCATION
5555 023064  104202                            ERROR   +202            ;CSR PATTERN NOT CORRECT
5556 023066  062701  000002            6$:     ADD     #2,R1           ;INCREMENT R1 TO NEXT EXPECTED PATTERN
5557 023072  005202                            INC     R2              ;INCREMENT THE PATTERN
5558 023074  005303                            DEC     R3              ;DECREMENT THE LOOP COUNTER
5559 023076  001303                            BNE     999$            ;BRANCH BACK IF NOT DONE
5560 023100  062702  000200                    ADD     #200,R2         ;ADD 200 TO PATTERN LOCATION
```

'EST #30 - CHECK CSR BIT PATTERNS WITH MAINT BIT CLEAR

```
5561 023104  005300                    DEC     R0          ;DECREMENT THE LOOP COUNTER AND
5562 023106  001275                    BNE     1$          ;BRANCH BACK IF 2ND OCTAL GROUP NOT DONE
```

TEST #31 - CHECK BAR WITH CSR CLEAR

```
                                        .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).
                                        ;*
                                        ;*********************************************************
 023110                                 TST31:
 023110  000004                        SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
 023112  012737  023146  001310         MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
 023120  032777  004000  156212         BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
 023126  001407                         BEQ     1001$           ;&& BRANCH IF NOT
 023130  104401  023136                 TYPE    ,1000$          ;&& TYPE: "T # 31"
 023134  000404                         BR      1001$           ;&& GET OVER ASCII
 023136     124     040     043 1000$:  .ASCIZ   /T # 31/<CRLF>  ;&& THE ASCII MESSAGE
                                        .EVEN
 023146                                 1001$:
5569 023146  004737  003646             999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
5570 023152  012777  001360  157340             MOV     #$TMP0,@BAR     ;PUT AN ADDRESS IN THE BAR
5571 023160  012777  000001  157334             MOV     #GO,@CSR        ;SET JUST THE GO BIT TO CLEAR THE READY BIT
5572 023166  017737  157330  002560             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5573 023174  001403                             BEQ     1$              ;BRANCH AROUND ERROR IF EQUAL TO ZERO
5574 023176  005037  002572                     CLR     ECSR            ;MOVE EXPECTED DATA TO ECSR
5575 023202  104040                             ERROR   +40             ;CSR IS WRONG
5576 023204  017737  157310  002566     1$:     MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
5577 023212  032737  000001  002566             BIT     #BIT0,RBAR      ;SEE IF THIS BIT IS CLEAR
5578 023220  001407                             BEQ     TST32           ;BRANCH TO NEXT TEST IF IT WAS
5579 023222  013737  002566  002600             MOV     RBAR,EBAR       ;MOVE EXPECTED DATA TO EBAR
5580 023230  042737  000001  002600             BIC     #BIT0,EBAR      ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
5581 023236  104044                             ERROR   +44             ;BAR IS WRONG
```

```
5588                                    .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.
                                        ;*
                                        ;;*****************************************************************
     023240                             TST32:
     023240  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     023242  012737  023312  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
     023250  032777  004000  156062             BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     023256  001407                             BEQ     1001$           ;&& BRANCH IF NOT
     023260  104401  023266                     TYPE    ,1000$          ;&& TYPE: "T # 32"
     023264  000404                             BR      1001$           ;&& GET OVER ASCII
     023266     124     040     043  1000$:    .ASCIZ  /T # 32/<CRLF>  ;&& THE ASCII MESSAGE
                                             .EVEN
     023276                             1001$:
5589 023276  012702  000007             1$:     MOV     #7,R2           ;SET UP LOOP COUNTER - DO 7 BIT PATTERNS
5590 023302  005037  002714                     CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
5591 023306  012703  007202                     MOV     #PATRNS,R3      ;MOVE ADDRESS OF PATTERNS TO R3
5592 023312  017737  157210  002532  999$:     MOV     @DRINV,$DRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5593 023320  017737  157204  002534             MOV     @DRVS,$DRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
5594 023326  012777  023470  157172             MOV     #3$,@DRINV      ;INTERRUPT VECTOR TO 4$
5595 023334  013777  002540  157166             MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
5596 023342  005037  177776                     CLR     PSW             ;LET THE DR11 INTERRUPT
5597 023346  012777  010000  157146             MOV     #MA,@CSR        ;DO AN INIT BY SETTING AND
5598 023354  005077  157142                     CLR     @CSR            ;CLEARING THE CSR MAINTENANCE BIT
5599 023360  012777  177777  157130             MOV     #-1,@WCR        ;SET UP FOR 1 TRANSFER
5600 023366  012777  002612  157124             MOV     #NPR1,@BAR      ;TRANSFER FROM BUS ADDRESS IN NPR1
5601 023374  005077  157124                     CLR     @BDR            ;GET READY TO RECEIVE DATA
5602 023400  011337  002612                     MOV     (R3),NPR1       ;SET UP TRANSFER DATA
5603 023404  012777  000110  157110             MOV     #F3+IE,@CSR     ;SET THE NON BURST (FNCT3) AND IE
5604 023412  052777  000401  157102             BIS     #CY+GO,@CSR     ;SET THE CYCLE AND GO BITS
5605 023420  005037  002660                     CLR     TIME            ;CLEAR THE TIME LOCATION FOR WAIT LOOP
5606 023424  005237  002660             2$:     INC     TIME            ;INCREMENT UNTIL WE GET TO ZERO AGAIN
5607 023430  001375                             BNE     2$              ;BRANCH BACK IF WE AREN'T THERE YET
5608 023432  013777  002532  157066             MOV     $DRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5609 023440  013777  002534  157062             MOV     $DRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5610 023446  017737  157050  002560             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5611 023454  011337  002604                     MOV     (R3),ENPR1      ;MOVE PATTERN TO ENPR1
5612 023460  104035                             ERROR   +35             ;DR11 FAILED TO INTERRUPT
5613 023462  005077  157034                     CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
5614 023466  000450                             BR      5$              ;BRANCH TO SEE IF THERE ARE ANY MORE PATTERNS TO CHECK
5615 023470  062706  000004             3$:     ADD     #4,SP           ;CLEAN STACK AFTER INTERRUPT
5616 023474  013777  002532  157024             MOV     $DRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5617 023502  013777  002534  157020             MOV     $DRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5618 023510  004737  004064                     JSR     PC,ERRCHK       ;CLEAR IE, CHECK FOR ERROR
5619 023514  104021                             ERROR   +21             ;ERROR BIT SHOULD HAVE BEEN CLEAR
5620 023516  017737  156776  002566             MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
5621 023524  012737  002613  002600             MOV     #NPR1+3,EBAR    ;MOVE EXPECTED DATA TO EBAR
5622 023532  017737  156766  002564             MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
5623 023540  011337  002576                     MOV     (R3),EBDR       ;MOVE EXPECTED DATA TO EBDR
5624 023544  017737  156746  002570             MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
5625 023552  005037  002602                     CLR     EWCR            ;MOVE EXPECTED DATA TO EWCR
5626 023556  023737  002566  002600             CMP     RBAR,EBAR       ;COMPARE RECEIVED WITH EXPECTED
5627 023564  001010                             BNE     4$              ;BRANCH IF WRONG
```

TEST #32 - TEST 7 SINGLE DATI NON BURST MODE TRANSFERS

```
5628 023566  023737  002564  002576          CMP     RBDR,EBDR       ;COMPARE RECEIVED WITH EXPECTED
5629 023574  001004                          BNE     4$              ;BRANCH IF WRONG
5630 023576  023737  002570  002602          CMP     RWCR,EWCR       ;COMPARE RECEIVED WITH EXPECTED
5631 023604  001401                          BEQ     5$              ;BRANCH IF OK
5632 023606  104211                  4$:     ERROR   +211            ;CSR AND-OR WCR AND-OR BAR ARE INCORRECT
5633 023610  062703  000002          5$:     ADD     #2,R3           ;INCREMENT TO NEXT PATTERN
5634 023614  005302                          DEC     R2              ;DECREMENT THE LOOP COUNTER
5635 023616  001235                          BNE     999$            ;BRANCH BACK IF NOT ZERO YET
```

TEST #33 - TEST STRING OF 200 DATIS BURST MODE XFERS

```
                                        .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.
                                        ;*
                                        ;;*****************************************************************
 023620                                 TST33:
 023620  000004                        SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
 023622  012737  023656  001310        MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
 023630  032777  004000  155502        BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
 023636  001407                        BEQ     1001$           ;&& BRANCH IF NOT
 023640  104401  023646                TYPE    ,1000$          ;&& TYPE: "T # 33"
 023644  000404                        BR      1001$           ;&& GET OVER ASCII
 023646     124     040     043 1000$: .ASCIZ  /T # 33/<CRLF>  ;&& THE ASCII MESSAGE
                                        .EVEN
 023656                                 1001$:
5642 023656  004737  003646            999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
5643 023662  012737  000200  002622            MOV     #200,BUFLEN     ;LENGTH OF BUFFER=200
5644 023670  004737  003302                    JSR     PC,LODBUF       ;LOAD THE BUFFER WITH INCREMENTING PATTERN
5645 023674  012737  177600  002630            MOV     #-200,WCLEN     ;PREPARE NUMBER FOR WCR
5646 023702  013777  002630  156606            MOV     WCLEN,@WCR      ;SET UP WCR
5647 023710  013777  002616  156602            MOV     INBUF,@BAR      ;SET UP BAR
5648 023716  012777  177777  156600            MOV     #-1,@BDR        ;MAINT AIDE
5649 023724  017737  156576  002532            MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5650 023732  017737  156572  002534            MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
5651 023740  012777  024050  156560            MOV     #2$,@DRINV      ;INT VECTOR
5652 023746  013777  002540  156554            MOV     LEVEL,@DRVS     ;INTERRUPT STATUS TO LEVEL OF DEVICE
5653 023754  005037  177776                    CLR     PSW             ;LET THE DR11 INTERRUPT
5654 023760  012777  000100  156534            MOV     #IE,@CSR        ;SET INTERRUPT ENABLE
5655 023766  052777  000401  156526            BIS     #CY+GO,@CSR     ;CYCLE, GO
5656 023774  012737  001000  002660            MOV     #1000,TIME      ;WAIT FOR INTERRUPT
5657 024002  005337  002660            1$:     DEC     TIME            ;DECREMENT UNTIL WE REACH ZERO
5658 024006  001375                    BNE     1$              ;BRANCH BACK IF NOT ZERO
5659 024010  013777  002532  156510            MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5660 024016  013777  002534  156504            MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5661 024024  017737  156472  002560            MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5662 024032  104035                    ERROR   +35             ;DR11 FAILED TO INTERRUPT
5663 024034  012777  010000  156460            MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
5664 024042  005077  156454                    CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
5665 024046  000426                    BR      TST34           ;BRANCH TO NEXT TEST
5666 024050  062706  000004            2$:     ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
5667 024054  013777  002532  156444            MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5668 024062  013777  002534  156440            MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5669 024070  004737  003356                    JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
5670 024074  104051                    ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
5671 024076  017737  156422  002564            MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
5672 024104  022737  000177  002564            CMP     #177,RBDR       ;CHECK THAT WORD #200 OF INBUF IS IN BDR
5673 024112  001404                    BEQ     TST34           ;BRANCH TO NEXT TEST IF IT IS
5674 024114  012737  000177  002576            MOV     #177,EBDR       ;MOVE EXPECTED DATA TO EBDR
5675 024122  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.
                                ;*
                                ;*****************************************************************
      024124                    TST34:
      024124  000004                    SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
      024126  012737  024176  001310    MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
      024134  032777  004000  155176    BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
      024142  001407                    BEQ     1001$           ;&& BRANCH IF NOT
      024144  104401  024152            TYPE    ,1000$          ;&& TYPE: "T # 34"
      024150  000404                    BR      1001$           ;&& GET OVER ASCII
      024152     124     040     043    1000$:  .ASCIZ  /T # 34/<CRLF>  ;&& THE ASCII MESSAGE
                                        .EVEN
      024162                    1001$:
 5683 024162  012702  000007    1$:     MOV     #7,R2           ;DO 7 BIT PATTERNS
 5684 024166  005037  002714            CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
 5685 024172  012703  007202            MOV     #PATRNS,R3      ;MOVE ADDRESS OF PATTERNS TO R3
 5686 024176  017737  156324  002532    999$:   MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
 5687 024204  017737  156320  002534            MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
 5688 024212  012777  024364  156306            MOV     #3$,@DRINV      ;INTERRUPT VECTOR TO 3$
 5689 024220  013777  002540  156302            MOV     LEVEL,@DRVS     ;INTERRUPT STATUS TO LEVEL OF DEVICE
 5690 024226  005037  177776            CLR     PSW             ;LET DR11 INTERRUPT
 5691 024232  012777  010000  156262            MOV     #MA,@CSR        ;DO AN INIT
 5692 024240  005077  156256            CLR     @CSR            ;FORCE ACCESS TO CSR
 5693 024244  012777  177777  156244            MOV     #-1,@WCR        ;SET UP FOR 1 TRANSFER
 5694 024252  012777  002612  156240            MOV     #NPR1,@BAR      ;TRANSFER TO BUS ADDRESS IN NPR1
 5695 024260  005037  002612            CLR     NPR1            ;GET READY TO RECEIVE DATA
 5696 024264  011377  156234            MOV     (R3),@BDR       ;SET UP TO TRANSFER DATA
 5697 024270  012777  000112  156224            MOV     #F1+F3+IE,@CSR  ;DATO (FNCT1), FNCT3, IE
 5698 024276  052777  000401  156216            BIS     #CY+GO,@CSR     ;CYCLE, GO
 5699 024304  012737  001000  002660            MOV     #1000,TIME      ;CLEAR THE TIME LOCATION FOR WAIT LOOP
 5700 024312  005337  002660    2$:     DEC     TIME            ;DECREMENT UNTIL WE GET BACK TO ZERO
 5701 024316  001375            BNE     2$              ;BRANCH BACK IF NOT ZERO
 5702 024320  013777  002532  156200            MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
 5703 024326  013777  002534  156174            MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
 5704 024334  017737  156162  002560            MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
 5705 024342  011337  001360            MOV     (R3),$TMP0      ;MOVE PATTERN TO $TMP0
 5706 024346  104035            ERROR   +35             ;DR11 FAILED TO INTERRUPT
 5707 024350  012777  010000  156144            MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
 5708 024356  005077  156140            CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
 5709 024362  000445            BR      6$              ;BRANCH TO DO NEXT PATTERN
 5710 024364  062706  000004    3$:     ADD     #4,SP           ;CLEAN UP STACK FROM INTERRUPT
 5711 024370  013777  002532  156130            MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
 5712 024376  013777  002534  156124            MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
 5713 024404  004737  004064            JSR     PC,ERRCHK       ;CLEAR IE, CHECK FOR ERROR
 5714 024410  104021            ERROR   +21             ;ERROR BIT SHOULD HAVE BEEN CLEAR
 5715 024412  017737  156100  002570            MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
 5716 024420  001403            BEQ     4$              ;BRANCH IF IT IS EQUAL TO ZERO
 5717 024422  011337  001362            MOV     (R3),$TMP1      ;MOVE PATTERN TO $TMP0
 5718 024426  104206            ERROR   +206            ;WCR NOT EQUAL TO ZERO
 5719 024430  017737  156064  002566    4$:     MOV     @BAR,RBAR       ;MOVE RECEIVED ADDRESS TO RBAR
 5720 024436  022737  002615  002566            CMP     #NPR1+3,RBAR    ;COMPARE CORRECT BAR WITH BAR CABLE MODE TESTING LEAVES
 5721                                                                    ;BIT 0 OF BAR SET. THEREFORE MUST CHECK FOR ODD ADDRESS
```

```
 5722 024444  001406                          BEQ     5$              ;BRANCH IF IT IS OK
 5723 024446  012737  002615  002600          MOV     #NPR1+3,EBAR    ;MOVE EXPECTED DATA TO EBAR
 5724 024454  011337  001362                  MOV     (R3),$TMP1      ;MOVE PATTERN TO $TMP0
 5725 024460  104207                          ERROR   +207            ;BAR IS WRONG
 5726 024462  021337  002612          5$:     CMP     (R3),NPR1       ;CHECK FOR CORRECT DATA
 5727 024466  001403                          BEQ     6$              ;BRANCH IF CORRECT DATA WAS TRANSFERRED
 5728 024470  011337  002604                  MOV     (R3),ENPR1      ;MOVE EXPECTED DATA TO ENPR1
 5729 024474  104210                          ERROR   +210            ;DATA NOT TRANSFERED CORRECTLY
 5730 024476  062703  000002          6$:     ADD     #2,R3           ;POINT TO NEXT BIT PATTERN
 5731 024502  005302                          DEC     R2              ;COUNT 1 PATTERN DONE
 5732 024504  001234                          BNE     999$            ;BRANCH BACK IF NOT DONE
```

```
5738                                    .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.
                                        ;*
                                        ;;*****************************************************************
     024506                             TST35:
     024506  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     024510  012737  024544  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
     024516  032777  004000  154614             BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     024524  001407                             BEQ     1001$           ;&& BRANCH IF NOT
     024526  104401  024534                     TYPE    ,1000$          ;&& TYPE: "T # 35"
     024532  000404                             BR      1001$           ;&& GET OVER ASCII
     024534     124     040     043  1000$:     .ASCIZ  /T # 35/<CRLF>  ;&& THE ASCII MESSAGE
                                                .EVEN
     024544                             1001$:
5739 024544  005077  155752             999$:   CLR     @CSR            ;FORCE ACCESS TO CSR
5740 024550  012737  000200  002622             MOV     #200,BUFLEN     ;LENGTH OF BUFFER=200
5741 024556  004737  003302                     JSR     PC,LODBUF       ;LOAD THE BUFFER WITH INCREMENTING PATTERN
5742 024562  013737  002622  002630             MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
5743 024570  005437  002630                     NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
5744 024574  013777  002630  155714             MOV     WCLEN,@WCR      ;SET UP WCR
5745 024602  013777  002616  155710             MOV     INBUF,@BAR      ;SET UP BAR
5746 024610  012777  052525  155706             MOV     #52525,@BDR     ;SET UP BDR
5747 024616  017737  155704  002532             MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5748 024624  017737  155700  002534             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
5749 024632  012777  024742  155666             MOV     #2$,@DRINV      ;INTERRUPT VECTOR
5750 024640  013777  002540  155662             MOV     LEVEL,@DRVS     ;INTERRUPT STATUS TO LEVEL OF DEVICE
5751 024646  005037  177776                     CLR     PSW             ;LET THE DR11 INTERRUPT
5752 024652  012777  000102  155642             MOV     #IE+F1,@CSR     ;IE, FNCT1
5753 024660  052777  000401  155634             BIS     #CY+GO,@CSR     ;CYCLE, GO
5754 024666  012737  001000  002660             MOV     #1000,TIME      ;MOVE WAIT LOOP VALUE TO TIME LOCATION
5755 024674  005337  002660          1$:        DEC     TIME            ;DECREMENT UNTIL WE REACH ZERO
5756 024700  001375                             BNE     1$              ;BRANCH BACK IF NOT ZERO
5757 024702  013777  002532  155616             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5758 024710  013777  002534  155612             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5759 024716  017737  155600  002560             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5760 024724  104035                             ERROR   +35             ;DR11 FAILED TO INTERRUPT
5761 024726  012777  010000  155566             MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
5762 024734  005077  155562                     CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
5763 024740  000421                             BR      TST36           ;BRANCH TO NEXT TEST
5764 024742  062706  000004          2$:        ADD     #4,SP           ;CLEAN STACK AFTER INTERRUPT
5765 024746  013777  002532  155552             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5766 024754  013777  002534  155546             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5767 024762  004737  003356                     JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
5768 024766  104051                             ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
5769 024770  004737  003716                     JSR     PC,DATOCK       ;CHECK INBUF
5770 024774  104046                             ERROR   +46             ;BUFFER DATA NOT CORRECT
5771 024776  004737  004000                     JSR     PC,DATOC2       ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATOCK
5772 025002  104047                             ERROR   +47             ;TOO MANY WORDS WERE TRANSFERED
```

TEST #36 - TEST STRING OF 200 DATIS NON-BURST MODE

```
                                        .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.
                                        ;*
                                        ;;*****************************************************************
025004                                  TST36:
025004  000004                                 SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
025006  012737  025042  001310                 MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
025014  032777  004000  154316                 BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
025022  001407                                 BEQ     1001$           ;&& BRANCH IF NOT
025024  104401  025032                         TYPE    ,1000$          ;&& TYPE: "T # 36"
025030  000404                                 BR      1001$           ;&& GET OVER ASCII
025032     124     040     043        1000$:   .ASCIZ  /T # 36/<CRLF>  ;&& THE ASCII MESSAGE
                                               .EVEN
025042                                  1001$:
5779 025042  004737  003646            999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
5780 025046  005077  155450                     CLR     @CSR            ;FORCE ACCESS TO CSR
5781 025052  012737  000200  002622             MOV     #200,BUFLEN     ;LENGTH OF BUFFER=200
5782 025060  004737  003302                     JSR     PC,LODBUF       ;LOAD THE BUFFER WITH INCREMENTING PATTERN
5783 025064  013737  002622  002630             MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
5784 025072  005437  002630                     NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
5785 025076  013777  002630  155412             MOV     WCLEN,@WCR      ;SET-UP WCR
5786 025104  013777  002616  155406             MOV     INBUF,@BAR      ;SET-UP BAR
5787 025112  012777  177777  155404             MOV     #-1,@BDR        ;MAINT AIDE
5788 025120  017737  155402  002532             MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5789 025126  017737  155376  002534             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
5790 025134  012777  025244  155364             MOV     #2$,@DRINV      ;INT VECTOR
5791 025142  013777  002540  155360             MOV     LEVEL,@DRVS     ;INTERRUPT STATUS TO LEVEL OF DEVICE
5792 025150  005037  177776                     CLR     PSW             ;LET THE DR11 INTERRUPT
5793 025154  012777  000110  155340             MOV     #F3+IE,@CSR     ;FNCT3, IE
5794 025162  052777  000401  155332             BIS     #CY+GO,@CSR     ;CYCLE, GO
5795 025170  012737  001000  002660             MOV     #1000,TIME      ;SET WAIT LOOP COUNTER
5796 025176  005337  002660          1$:        DEC     TIME            ;DECREMENT UNTIL WE REACH ZERO
5797 025202  001375                             BNE     1$              ;BRANCH BACK IF NOT ZERO
5798 025204  013777  002532  155314             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5799 025212  013777  002534  155310             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5800 025220  017737  155276  002560             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5801 025226  104035                             ERROR   +35             ;DR11 FAILED TO INTERRUPT
5802 025230  012777  010000  155264             MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
5803 025236  005077  155260                     CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
5804 025242  000426                             BR      TST37           ;BRANCH TO NEXT TEST
5805 025244  062706  000004          2$:        ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
5806 025250  013777  002532  155250             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5807 025256  013777  002534  155244             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5808 025264  004737  003356                     JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
5809 025270  104051                             ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
5810 025272  017737  155226  002564             MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
5811 025300  022737  000177  002564             CMP     #177,RBDR       ;CHECK THAT WORD #200 OF INBUF IS IN BDR
5812 025306  001404                             BEQ     TST37           ;BRANCH TO NEXT TEST IF OK
5813 025310  012737  000177  002576             MOV     #177,EBDR       ;MOVE EXPECTED DATA TO EBDR
5814 025316  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.
                                        ;*
                                        ;;*****************************************************************
     025320                             TST37:
     025320  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     025322  012737  025356  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
     025330  032777  004000  154002             BIT     #BIT11,@SWR     ;&& TEST TO SEE IF TEST NUMBER TRACER ENABLED
     025336  001407                             BEQ     1001$           ;&& BRANCH IF NOT
     025340  104401  025346                     TYPE    ,1000$          ;&& TYPE: "T # 37"
     025344  000404                             BR      1001$           ;&& GET OVER ASCII
     025346     124     040     043  1000$:     .ASCIZ  /T # 37/<CRLF>  ;&& THE ASCII MESSAGE
                                             .EVEN
     025356                             1001$:
5821 025356  004737  003646             999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
5822 025362  005077  155134                     CLR     @CSR            ;FORCE ACCESS TO CSR
5823 025366  012737  000200  002622             MOV     #200,BUFLEN     ;LENGTH OF BUFFER=200
5824 025374  004737  003302                     JSR     PC,LODBUF       ;LOAD THE BUFFER WITH INCREMENTING PATTERN
5825 025400  013737  002622  002630             MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
5826 025406  005437  002630                     NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
5827 025412  013777  002630  155076             MOV     WCLEN,@WCR      ;SET UP WCR
5828 025420  013777  002616  155072             MOV     INBUF,@BAR      ;SET UP BAR
5829 025426  012777  052525  155070             MOV     #52525,@BDR     ;SET UP BDR
5830 025434  017737  155066  002532             MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
5831 025442  017737  155062  002534             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
5832 025450  012777  025560  155050             MOV     #2$,@DRINV      ;INTERRUPT VECTOR
5833 025456  013777  002540  155044             MOV     LEVEL,@DRVS     ;INTERRUPT STATUS TO LEVEL OF DEVICE
5834 025464  005037  177776                     CLR     PSW             ;LET THE DR11 INTERRUPT
5835 025470  012777  000112  155024             MOV     #F1+F3+IE,@CSR  ;FNCT1, FNCT3, IE
5836 025476  052777  000401  155016             BIS     #CY+GO,@CSR     ;CYCLE, GO
5837 025504  012737  001000  002660             MOV     #1000,TIME      ;SET WAIT LOOP COUNTER
5838 025512  005337  002660         1$:         DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
5839 025516  001375                             BNE     1$              ;BRANCH BACK IF NOT ZERO
5840 025520  013777  002532  155000             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5841 025526  013777  002534  154774             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5842 025534  017737  154762  002560             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
5843 025542  104035                             ERROR   +35             ;DR11 FAILED TO INTERRUPT
5844 025544  012777  010000  154750             MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
5845 025552  005077  154744                     CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
5846 025556  000421                             BR      TST40           ;BRANCH TO NEXT TEST
5847 025560  062706  000004         2$:         ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
5848 025564  013777  002532  154734             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
5849 025572  013777  002534  154730             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
5850 025600  004737  003356                     JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
5851 025604  104051                             ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
5852 025606  004737  003716                     JSR     PC,DATOCK       ;CHECK INBUF
5853 025612  104046                             ERROR   +46             ;BUFFER DATA NOT CORRECT
5854 025614  004737  004000                     JSR     PC,DATOC2       ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATOCK
5855 025620  104047                             ERROR   +47             ;TOO MANY WORDS WERE TRANSFERED
```

```
                                .SBTTL  END OF DEVICE PASS ROUTINE
                                ;;*****************************************************************
025622  000004                  ENDEV:  SCOPE                   ;FOR POSSIBLE LOOP ON TEST
025624  005237  026010                  INC     PREOP           ;INCREMENT DEVICE PASS COUNTER          ;DPM001
025630  005037  001302                  CLR     $TSTNM          ;CLEAR TEST NO. COUNT FOR SCOPE ROUTINE
025634  022737  000001  002414          CMP     #1,QTYBRD       ;IS THERE MORE THAN 1 BOARD UNDER TEST?
025642  001450                          BEQ     5$              ;BRANCH IF NO
025644  005737  001312                  TST     $ERTTL          ;SEE IF THERE WERE ANY ERRORS
025650  001013                          BNE     2$              ;BRANCH AROUND EOP TEST IF SO
025652  004737  026012                  JSR     PC,DOWEPR       ;GO SEE IF WE SHOULD PRINT THE EOP      ;DPM001
025656  000442                          BR      5$              ;HERE IF NOT                            ;DPM001
025660  000411                          BR      3$              ;HERE IF SO                             ;DPM001
025662  032737  000077  026010          BIT     #77,PREOP       ;HERE IF ENABLED - DO WE PRINT THIS EOP?;DPM001
025670  001035                          BNE     5$              ;BRANCH IF SO                           ;DPM001
025672  005037  026010                  CLR     PREOP           ;CLEAR THE DEVICE PASS COUNTER          ;DPM001
025676  000402                          BR      3$              ;BRANCH                                 ;DPM001
025700  104401  001405          2$:     TYPE    ,$CRLF          ;TYPE A <CRLF>
025704  104401  037424          3$:     TYPE    ,BOARD          ;TYPE:  'BOARD # '
025710  013746  001422                  MOV     $UNIT,-(SP)     ;;SAVE $UNIT FOR TYPEOUT
025714  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
025716  104401  040341                  TYPE    ,TSTCOM         ;TYPE:  ' TESTING COMPLETE, PASS #'     ;DPM001
025722  013746  002716                  MOV     $PASS2,-(SP)    ;MOVE MAJOR PASS COUNTER TO STACK       ;DPM001
025726  013746  001416                  MOV     $PASS,-(SP)     ;MOVE PASS NUMBER TO STACK              ;DPM001
025732  104414                          TYPDE                   ;TYPE NUMBER IN EXTENDED DECIMAL        ;DPM001
025734  005737  001312                  TST     $ERTTL          ;SEE IF ANY ERRORS THIS DEVICE
025740  001407                          BEQ     4$              ;BRANCH AROUND TOTAL ERRORS MESSAGE IF NONE
025742  104401  036350                  TYPE    ,ETDEV          ;TYPE:  ' - TOTAL ERRORS THIS DEVICE = '
025746  013746  001312                  MOV     $ERTTL,-(SP)    ;MOVE NUMBER OF ERRORS TO THE STACK
025752  104405                          TYPDS                   ;GO TYPE THE NUMBER
025754  004737  003126                  JSR     PC,ERCAPT       ;GO LOG THE UNIT #, PASS # & # OF ERRORS THIS DEVICE
025760  104401  001405          4$:     TYPE    ,$CRLF          ;TYPE A <CRLF>
025764  005237  001420          5$:     INC     $DEVCT          ;INCREMENT DEVICE COUNTER
025770  023737  002414  001420          CMP     QTYBRD,$DEVCT   ;ALL DEVICES TESTED?
025776  001423                          BEQ     $EOP            ;GO TO END OF PASS ROUTINE IF SO
026000  005237  001422                  INC     $UNIT           ;INCREMENT THE UNIT NUMBER
026004  000137  012370                  JMP     TSTDEV          ;GO TEST NEXT DEVICE
026010  000000                  PREOP:  .WORD   0               ;DEVICE PASS COUNTER                    ;DPM001

026012  105737  002706          DOWEPR: TSTB    EOPLOC          ;SEE IF EOP MESSAGES ARE TO PRINT       ;DPM001
026016  001406                          BEQ     1$              ;BRANCH IF SO                           ;DPM001
026020  105737  032555                  TSTB    CHARCT          ;SEE IF A CHARACTER IS WAITING          ;DPM001
026024  001407                          BEQ     3$              ;BRANCH IF NOT                          ;DPM001
026026  062716  000002                  ADD     #2,(SP)         ;RETURN TO 2ND RETURN LOCATION          ;DPM001
026032  000402                          BR      2$              ;RETURN                                 ;DPM001
026034  062716  000004          1$:     ADD     #4,(SP)         ;FUDGE RETURN TO TEST FOR EOP PRINT     ;DPM001
026040  105037  032555          2$:     CLRB    CHARCT          ;CLEAR CHARCT                           ;DPM001
026044  000207                  3$:     RTS     PC              ;RETURN TO USER                         ;DPM001
```

```
 5902                                   .SBTTL  END OF PASS ROUTINE

                                        ;;*****************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY"
                                        ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO GOAGIN

026046                                  $EOP:
026046  000004                                  SCOPE
026050  005037  001302                          CLR     $TSTNM          ;;ZERO THE TEST NUMBER
026054  005237  001416                          INC     $PASS           ;;INCREMENT THE PASS NUMBER
026060  100004                                  BPL     1$              ;BRANCH IF STILL POSITIVE
026062  005037  001416                          CLR     $PASS           ;CLEAR $PASS AND
026066  005237  002716                          INC     $PASS2          ;INCREMENT OVERFLOW LOCATION
026072  005327                          1$:     DEC     (PC)+           ;;LOOP?
026074  000001                          $EOPCT: .WORD   1
026076  003116                                  BGT     $DOAGN          ;;YES
026100  012737                                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
026102  000001                          $ENDCT: .WORD   1
026104  026074                                  .WORD   $EOPCT
026106  005737  001312                          TST     $ERTTL          ;SEE IF ANY ERRORS THIS PASS            ;DPM001
026112  001012                                  BNE     2$              ;BRANCH IF SO TO PRINT EOP              ;DPM001
026114  004737  026012                          JSR     PC,DOWEPR       ;GO SEE IF EOP IS TO PRINT              ;DPM001
026120  000475                                  BR      $GET42          ;HERE IF NOT                            ;DPM001
026122  000406                                  BR      2$              ;HERE IF SO                             ;DPM001
026124  032737  000077  026010                  BIT     #77,PREOP       ;HERE IF ENABLED - DO WE PRINT THIS EOP?;DPM001
026132  001070                                  BNE     $GET42          ;BRANCH IF SO                           ;DPM001
026134  005037  026010                          CLR     PREOP           ;CLEAR THE DEVICE PASS COUNTER          ;DPM001
026140                                  2$:
026140  104401  026146                          TYPE    ,65$            ;;TYPE ASCIZ STRING
026144  000407                                  BR      64$             ;;GET OVER THE ASCIZ
026146     012     015     105          65$:    .ASCIZ  <12><15>/END PASS #/
                                                .EVEN
026164                                  64$:
026164  013746  002716                          MOV     $PASS2,-(SP)    ;MOVE UPPER PASS COUNT TO STACK         ;DPM001
026170  013746  001416                          MOV     $PASS,-(SP)     ;MOVE PASS COUNT TO STACK               ;DPM001
026174  104414                                  TYPDE                   ;GO TYPE PASS COUNT IN EXTENDED DECIMAL ;DPM001
026176  005737  001312                          TST     $ERTTL          ;SEE IF ANY ERRORS THIS PASS            ;DPM001
026202  001442                                  BEQ     $GET            ;BRANCH IF NOT                          ;DPM001
026204  022737  000001  002414                  CMP     #1,QTYBRD       ;ARE WE TESTING 1 BOARD?                ;DPM001
026212  001002                                  BNE     3$              ;BRANCH IF NOT - LOGGING IS DONE        ;DPM001
026214  004737  003126                          JSR     PC,ERCAPT       ;GO LOG THE UNIT #, PASS # & # OF ERRORS THIS DEVICE
026220                                  3$:
026220  104401  026226                          TYPE    ,67$            ;;TYPE ASCIZ STRING
026224  000424                                  BR      66$             ;;GET OVER THE ASCIZ
026226     040     040     124          67$:    .ASCIZ  /  TOTAL ERRORS THIS PASS ALL MODULE(S) /
                                                .EVEN
026276                                  66$:
026276  013746  001312                          MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                                                        ;;TOTAL NUMBER OF ERRORS
026302  104405                                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
026304  005037  001312                          CLR     $ERTTL          ;;CLEAR ERROR TOTAL
026310  104401  001405                  $GET:   TYPE    ,$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
026314  013700  000042                  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
026320  001405                                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
```

END OF PASS ROUTINE

```
        026322  000005                          RESET                   ;;CLEAR THE WORLD
        026324  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
        026326  000240                          NOP                     ;;SAVE ROOM
        026330  000240                          NOP                     ;;FOR
        026332  000240                          NOP                     ;;ACT11
        026334                          $DOAGN:
        026334  000137                          JMP     @(PC)+          ;;RETURN
        026336  026344                  $RTNAD: .WORD   GOAGIN
        026340     377      377     000 $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
                                            .EVEN
  5903  026344  005037  001420          GOAGIN: CLR     $DEVCT          ;CLEAR DEVICE COUNT
  5904  026350  022737  000001  002414          CMP     #1,QTYBRD       ;IS THERE ONLY ONE DEVICE UNDER TEST?
  5905  026356  001002                          BNE     RSTRT           ;BR, IF NOT
  5906  026360  000137  012556                  JMP     REINIT          ;GO DO ANOTHER PASS
  5907  026364  005037  001422          RSTRT:  CLR     $UNIT           ;CLEAR UNIT NUMBER
  5908  026370  000137  012356                  JMP     BEGIN1          ;GO BEGIN TEST OF NEXT DEVICE
```

```
5909                                    .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
                                        ;*

026374  105737  001357          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
026400  100002                          BPL     1$              ;;BR IF YES
026402  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
026404  000430                          BR      3$              ;;LEAVE
026406  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
026410  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
026414  122737  000001  001430          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
026422  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
026424  132737  000100  001431          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
026432  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
026434  010037  026444                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
026440  004737  033634                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
026444  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
026446  132737  000040  001431  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
026454  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
026456  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
026460  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
026462  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
026464  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
026466  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
026472  000002                          RTI                     ;;RETURN
026474  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
026500  001430                          BEQ     8$
026502  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
026506  001006                          BNE     5$
026510  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
026512  104401                          TYPE                    ;;TYPE A CR AND LF
026514  001405                          $CRLF
026516  105037  026736                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
026522  000755                          BR      2$              ;;GET NEXT CHARACTER
026524  004737  026606          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
026530  123726  001356          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
026534  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
026536  013746  001354                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
026542  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
026546  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
026550  004737  026606                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
026554  105337  026736                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
026560  000770                          BR      7$              ;;LOOP
```

```
                                ;HORIZONTAL TAB PROCESSOR

026562  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
026566  004737  026606          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
026572  132737  000007  026736          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
026600  001372                          BNE     9$              ;;TAB STOP
026602  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
026604  000724                          BR      2$              ;;GET NEXT CHARACTER
026606  105777  152536          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
026612  100375                          BPL     $TYPEC
026614  116677  000002  152530          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
026622  105777  152516                  TSTB    @$TKS           ;;SEE IF KEYBOARD IS TALKING.
026626  100027                          BPL     2$              ;;BRANCH IF IT ISN'T.
026630  117737  152512  032555          MOVB    @$TKB,CHARCT    ;&& PUT CHARACTER IN CHARCT
026636  142737  000200  032555          BICB    #200,CHARCT     ;&& BIT CLEAR PARITY BIT.
026644  122737  000023  032555          CMPB    #23,CHARCT      ;&& SEE IF THIS IS A ^S.
026652  001015                          BNE     2$              ;;BRANCH TO CONTINUE IF IT ISN'T.
026654  105777  152464          3$:     TSTB    @$TKS           ;;WAIT FOR ANOTHER INPUT.
026660  100375                          BPL     3$              ;;BRANCH BACK IF NOT READY.
026662  117737  152460  032555          MOVB    @$TKB,CHARCT    ;&& PUT CHARACTER IN CHARCT
026670  142737  000200  032555          BICB    #200,CHARCT     ;&& BIT CLEAR PARITY BIT.
026676  122737  000021  032555          CMPB    #21,CHARCT      ;&& SEE IF THIS IS A ^Q.
026704  001363                          BNE     3$              ;;BRANCH BACK FOR MORE WAIT IF NOT.
026706  122766  000015  000002  2$:     CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
026714  001003                          BNE     1$              ;;BRANCH IF NO
026716  105037  026736                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
026722  000406                          BR      $TYPEX          ;;EXIT
026724  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
026732  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
026734  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
026736  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
026740  000207          $TYPEX: RTS     PC
```

```
 5911                                   .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 $TYPOC
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPCN                   ;;CALL FOR TYPEOUT
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT

 026742  017646  000000         $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
 026746  116637  000001 027165          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
 026754  112637  027167                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
 026760  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
 026764  000406                          BR      $TYPON
 026766  112737  000001 027165 $TYPOC:  MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
 026774  112737  000006 027167          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
 027002  112737  000005 027164 $TYPON:  MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
 027010  010346                          MOV     R3,-(SP)        ;;SAVE R3
 027012  010446                          MOV     R4,-(SP)        ;;SAVE R4
 027014  010546                          MOV     R5,-(SP)        ;;SAVE R5
 027016  113704  027167                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
 027022  005404                          NEG     R4
 027024  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
 027030  110437  027166                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
 027034  113704  027165                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
 027040  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
 027044  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
 027046  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
 027050  000404                          BR      3$              ;;GO DO MSB
 027052  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
 027054  006105                          ROL     R5
 027056  006105                          ROL     R5
 027060  010503                          MOV     R5,R3
 027062  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
 027064  105337  027166                  DECB    $OMODE          ;;TYPE THIS DIGIT?
 027070  100016                          BPL     7$              ;;BR IF NO
 027072  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
 027076  001002                          BNE     4$              ;;TEST FOR 0
 027100  005704                          TST     R4              ;;SUPPRESS THIS 0?
 027102  001403                          BEQ     5$              ;;BR IF YES
 027104  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
```

BINARY TO OCTAL (ASCII) AND TYPE

```
027106  052703  000060            BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
027112  052703  000040     5$:    BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
027116  110337  027162            MOVB    R3,8$           ;;SAVE FOR TYPING
027122  104401  027162            TYPE    ,8$             ;;GO TYPE THIS DIGIT
027126  105337  027164     7$:    DECB    $OCNT           ;;COUNT BY 1
027132  003347                    BGT     2$              ;;BR IF MORE TO DO
027134  002402                    BLT     6$              ;;BR IF DONE
027136  005204                    INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
027140  000744                    BR      2$              ;;GO DO THE LAST DIGIT
027142  012605             6$:    MOV     (SP)+,R5        ;;RESTORE R5
027144  012604                    MOV     (SP)+,R4        ;;RESTORE R4
027146  012603                    MOV     (SP)+,R3        ;;RESTORE R3
027150  016666  000002  000004    MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
027156  012616                    MOV     (SP)+,(SP)
027160  000002                    RTI                     ;;RETURN
027162     000     8$:    .BYTE   0               ;;STORAGE FOR ASCII DIGIT
027163     000            .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
027164     000     $OCNT: .BYTE   0               ;;OCTAL DIGIT COUNTER
027165     000     $OFILL:.BYTE   0               ;;ZERO FILL SWITCH
027166  000000     $OMODE:.WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
 5913                                   .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

 027170  012737  000001  001364  $TYPDE: MOV     #1,$TMP2        ;&& SHOW ENTRY AT $TYPDE
 027176  000402                          BR      $TYPD           ;&& GO TO ROUTINE
 027200  005037  001364          $TYPDS: CLR     $TMP2           ;&& SHOW ENTRY AT $TYPDS
 027204                          $TYPD:
 027204  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 027206  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 027210  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 027212  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 027214  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 027216  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
 027222  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
 027226  100004                          BPL     1$              ;;BR IF INPUT IS POS.
 027230  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
 027232  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
 027240  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
 027242  012703  027504                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
 027246  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
 027252  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
 027254  016001  027474                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
 027260  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
 027262  002402                          BLT     4$              ;;BR IF DONE
 027264  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
 027266  000774                          BR      3$
 027270  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
 027272  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
 027274  001002                          BNE     5$              ;;FALL THROUGH IF 0
 027276  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
 027300  100407                          BMI     7$              ;;BR IF YES
 027302  106316                  5$:     ASLB    (SP)            ;;MSD?
 027304  103003                          BCC     6$              ;;BR IF NO
 027306  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
 027314  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
 027320  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
 027324  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
 027326  005720                          TST     (R0)+           ;;JUST INCREMENTING
```

```
027330  020027  000010              CMP     R0,#10          ;;CHECK THE TABLE INDEX
027334  002746                      BLT     2$              ;;GO DO THE NEXT DIGIT
027336  003002                      BGT     8$              ;;GO TO EXIT
027340  010502                      MOV     R5,R2           ;;GET THE LSD
027342  000764                      BR      6$              ;;GO CHANGE TO ASCII
027344  105726                8$:   TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
027346  100003                      BPL     9$              ;;BR IF NO
027350  116663  177777  177776      MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
027356  105013                9$:   CLRB    (R3)            ;;SET THE TERMINATOR
027360  005737  001364              TST     $TMP2           ;&& WAS ENTRY AT $TYPDS?
027364  001405                      BEQ     10$             ;&& BRANCH IF SO
027366  005766  000020              TST     20(SP)          ;&& TEST THE OVERFLOW LOCATION
027372  001402                      BEQ     10$             ;&& BRANCH IF NON-ZERO
027374  004737  027516              JSR     PC,EXPAND       ;&& GO EXPAND THE DECIMAL NUMBER
027400                        10$:
027400  012605                      MOV     (SP)+,R5        ;;POP STACK INTO R5
027402  012603                      MOV     (SP)+,R3        ;;POP STACK INTO R3
027404  012602                      MOV     (SP)+,R2        ;;POP STACK INTO R2
027406  012601                      MOV     (SP)+,R1        ;;POP STACK INTO R1
027410  012600                      MOV     (SP)+,R0        ;;POP STACK INTO R0
027412  104401  027504              TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
027416  005737  001364              TST     $TMP2           ;&& SEE IF ENTRY WAS AT $TYPDS
027422  001417                      BEQ     11$             ;&& BRANCH IF SO
027424  016666  000002  000006      MOV     2(SP),6(SP)     ;&& ADJUST THE STACK
027432  012666  000002              MOV     (SP)+,2(SP)     ;&&
027436  005726                      TST     (SP)+           ;&&
027440  105037  027513              CLRB    $DBLK+7         ;&& REPLACE ORIGINAL TERMINATOR
027444  112737  000040  027512      MOVB    #' ,$DBLK+6     ;&& REPLACE ORIGINAL SPACE CHARACTER
027452  112737  000040  027504      MOVB    #' ,$DBLK       ;&& REPLACE ORIGINAL SPACE CHARACTER
027460  000002                      RTI                     ;&& RETURN TO USER
027462  016666  000002  000004 11$:  MOV     2(SP),4(SP)     ;;ADJUST THE STACK
027470  012616                      MOV     (SP)+,(SP)
027472  000002                      RTI                     ;;RETURN TO USER
027474  023420  001750  000144 $DTBL: .WORD   10000.,1000.,100.,10.
027504                   $DBLK:  .BLKW  5
```

```
                                        .SBTTL  SUBROUTINE TO EXPAND DECIMAL TO LARGER THAN 32767
                                    ;;**********************************************************************
027516  012702  027512          EXPAND: MOV     #$DBLK+6,R2     ;&& MOVE LOCATION OF LCD+1 TO R2
027522  012703  027516                  MOV     #$DBLK+12,R3    ;&& MOVE NEW LOCATION OF LCD+2 TO R3
027526  105043                          CLRB    -(R3)           ;&& MAKE SURE TERMINATOR IS THERE
027530  114243                          MOVB    -(R2),-(R3)     ;&& MOVE THE 5 ASCII'S TO THEIR NEW LOCATIONS
027532  114243                          MOVB    -(R2),-(R3)     ;&&
027534  114243                          MOVB    -(R2),-(R3)     ;&&
027536  114243                          MOVB    -(R2),-(R3)     ;&&
027540  114243                          MOVB    -(R2),-(R3)     ;&&
027542  112743  000040                  MOVB    #' ,-(R3)       ;&& MOVE 4 SPACE CHARACTERS TO THE 4 NEW LOCATIONS
027546  112743  000040                  MOVB    #' ,-(R3)       ;&&
027552  112743  000040                  MOVB    #' ,-(R3)       ;&&
027556  112743  000040                  MOVB    #' ,-(R3)       ;&&
027562  013746  001360                  MOV     $TMP0,-(SP)     ;&& SAVE $TMP0
027566  005037  001360                  CLR     $TMP0           ;&& CLEAR LOCATION TO USE AS ACCUMULATOR
027572  013746  001362                  MOV     $TMP1,-(SP)     ;&& SAVE $TMP1
027576  005037  001362                  CLR     $TMP1           ;&& CLEAR LOCATION TO USE AS 2ND ACCUMULATOR
027602  012701  027513                  MOV     #$DBLK+7,R1     ;&& MOVE ADDRESS OF LCD TO R2
027606  012702  027762                  MOV     #$NUMS+10,R2    ;&& MOVE ADDRESS+10 OF WORD STREAM OF 8^6 TO R2
027612  016600  000026          1$:     MOV     26(SP),R0       ;&& MOVE OVERFLOW LOCATION CONTENTS TO R0
027616  111137  001360                  MOVB    (R1),$TMP0      ;&& MOVE ASCII TO THE TEMPORARY LOCATION
027622  052737  000060  001360          BIS     #'0,$TMP0       ;&& MAKE LOCATION AN ASCII IF NOT ALREADY
027630  061237  001360          2$:     ADD     (R2),$TMP0      ;&& ADD THE NUMBER TO THE ASCII
027634  022737  000071  001360          CMP     #'9,$TMP0       ;&& HAVE WE SURPASSED ASCII '9'?
027642  002025                          BGE     4$              ;&& BRANCH IF NOT
027644  162737  000012  001360          SUB     #10.,$TMP0      ;&& SUBTRACT 10 FROM THE ASCII AND
027652  010103                          MOV     R1,R3           ;&& MOVE PRESENT ASCII ADDRESS TO R3
027654  005303                  3$:     DEC     R3              ;&& DECREMENT TO NEXT SIGNIFICANT ASCII DIGIT
027656  105213                          INCB    (R3)            ;&& ADD THE CARRY TO THE ASCII
027660  152713  000060                  BISB    #'0,(R3)        ;&& SET BITS TO MAKE IT A NUMBER ASCII
027664  122713  000071                  CMPB    #'9,(R3)        ;&& SEE IF CARRY NEEDS TO BE TRANSFERED
027670  002012                          BGE     4$              ;&& BRANCH IF NOT
027672  111337  001362                  MOVB    (R3),$TMP1      ;&& MOVE BYTE TO LOCATION $TMP1
027676  162737  000012  001362          SUB     #10.,$TMP1      ;&& SUBTRACT 10 DECIMAL AND
027704  113713  001362                  MOVB    $TMP1,(R3)      ;&& MOVE IT BACK
027710  022703  027504                  CMP     #$DBLK,R3       ;&& SEE IF ALL POSITIONS HAVE BEEN DONE
027714  001357                          BNE     3$              ;&& BRANCH BACK IF NOT
027716  005300                  4$:     DEC     R0              ;&& DECREMENT THE OVERFLOW LOCATION R0
027720  001343                          BNE     2$              ;&& BRANCH IF NOT ALL ADDED
027722  113711  001360                  MOVB    $TMP0,(R1)      ;&& MOVE NEW NUMBER TO LOCATION
027726  005301                          DEC     R1              ;&& POINT R1 TO THE NEXT ASCII LOCATION
027730  005742                          TST     -(R2)           ;&& POINT R2 TO NEXT DIGIT TO ADD
027732  022701  027507                  CMP     #$DBLK+3,R1     ;&& SEE IF ALL DIGITS HAVE BEEN ADDED
027736  001325                          BNE     1$              ;&& BRANCH IF NOT DONE
027740  012637  001362                  MOV     (SP)+,$TMP1     ;&& RESTORE $TMP1
027744  012637  001360                  MOV     (SP)+,$TMP0     ;&& RESTORE $TMP0
027750  000207                          RTS     PC              ;&& RETURN
027752  000003  000002  000007  $NUMS:  .WORD   3,2,7,6,8.
```

```
 5915                                   .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.
027764  022737  000176  001340  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
027772  001112                          BNE     15$             ;;BRANCH IF NO
027774  105737  032555                  TSTB    CHARCT          ;&& SEE IF CHARACTER WAS INPUTED DURING PRINT
030000  001405                          BEQ     1$              ;&& BRANCH IF NOT
030002  113746  032555                  MOVB    CHARCT,-(SP)    ;&& MOVE CHARACTER TO THE STACK
030006  105037  032555                  CLRB    CHARCT          ;&& CLEAR THE CHARACTER FROM CHARCT
030012  000405                          BR      2$              ;&& GO CHECK IT OUT
030014  105777  151324          1$:     TSTB    @$TKS           ;;CHAR THERE?
030020  100077                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
030022  117746  151320                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
030026  042716  177600          2$:     BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
030032  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
030036  001404                          BEQ     3$              ;&& YES, BRANCH AROUND RETURN TO USER SETUP
030040  116637  177776  032555          MOVB    -2(SP),CHARCT   ;&& MOVE CHARACTER BACK TO CHARCT
030046  000464                          BR      15$             ;&& RETURN TO USER
030050  123727  001334  000001  3$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
030056  001460                          BEQ     15$             ;;BRANCH IF YES

030060  104401  030544                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
030064  104401  030551          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
030070  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
030074  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
030076  104401  030562                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
030102  105037  032555          19$:    CLRB    CHARCT          ;&& CLEAR ANY CHARACTER THAT WAS INPUTED DURING PRINT
030106  005046                          CLR     -(SP)           ;;CLEAR COUNTER
030110  005046                          CLR     -(SP)           ;;THE NEW SWR
030112  105777  151226          7$:     TSTB    @$TKS           ;;CHAR THERE?
030116  100375                          BPL     7$              ;;IF NOT TRY AGAIN

030120  117746  151222                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
030124  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII


030130  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
030134  001005                          BNE     10$             ;;BRANCH IF NOT
030136  104401  030537                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
030142  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
030146  000746                          BR      $GTSWR          ;;LET'S TRY IT AGAIN


030150  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
030154  001022                          BNE     16$             ;;BRANCH IF NO
030156  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
030162  001403                          BEQ     11$             ;;BRANCH IF YES
030164  016677  000002  151146          MOV     2(SP),@SWR      ;;SAVE NEW SWR
030172  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
```

TTY INPUT ROUTINE

```
030176  104401  001405          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
030202  123727  001335  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
030210  001003                          BNE     15$             ;;BRANCH IF NOT
030212  012777  000100  151124          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
030220  000002                  15$:    RTI                     ;;RETURN
030222  004737  026606          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
030226  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
030232  002420                          BLT     16$             ;;BRANCH IF YES
030234  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
030240  003015                          BGT     18$             ;;BRANCH IF YES
030242  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
030246  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
030252  001403                          BEQ     17$             ;;BRANCH IF YES
030254  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
030256  006316                          ASL     (SP)            ;;    CHAR OVER TO MAKE
030260  006316                          ASL     (SP)            ;;    ROOM FOR NEW ONE.
030262  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
030266  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
030272  000707                          BR      7$              ;;GET THE NEXT ONE
030274  104401  001404          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
030300  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
                                ;

030302  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
030304  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
030312  105777  151026          1$:     TSTB    @$TKS           ;;WAIT FOR
030316  100375                          BPL     1$              ;;A CHARACTER
030320  117766  151022  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
030326  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
030334  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
030342  001013                          BNE     3$              ;;BRANCH IF NO
030344  105777  150774          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
030350  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
030352  117746  150770                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
030356  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
030362  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
030366  001366                          BNE     2$              ;;IF NOT DISCARD IT
030370  000750                          BR      1$              ;;YES, RESUME
030372  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
030400  002407                          BLT     4$              ;;BRANCH IF YES
030402  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
030410  003003                          BGT     4$              ;;BRANCH IF YES
030412  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
030420  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

030422  010346                      $RDLIN: MOV     R3,-(SP)            ;;SAVE R3
030424  012703  030530              1$:     MOV     #$TTYIN,R3          ;;GET ADDRESS
030430  022703  030537              2$:     CMP     #$TTYIN+7.,R3       ;;BUFFER FULL?
030434  101405                              BLOS    4$                  ;;BR IF YES
030436  104410                              RDCHR                       ;;GO READ ONE CHARACTER FROM THE TTY
030440  112613                              MOVB    (SP)+,(R3)          ;;GET CHARACTER
030442  122713  000177              10$:    CMPB    #177,(R3)           ;;IS IT A RUBOUT
030446  001003                              BNE     3$                  ;;SKIP IF NOT
030450  104401  001404              4$:     TYPE    ,$QUES              ;;TYPE A '?'
030454  000763                              BR      1$                  ;;CLEAR THE BUFFER AND LOOP
030456  111337  030526              3$:     MOVB    (R3),9$             ;;ECHO THE CHARACTER
030462  104401  030526                      TYPE    ,9$
030466  122723  000015                      CMPB    #15,(R3)+           ;;CHECK FOR RETURN
030472  001356                              BNE     2$                  ;;LOOP IF NOT RETURN
030474  105063  177777                      CLRB    -1(R3)              ;;CLEAR RETURN (THE 15)
030500  104401  001406                      TYPE    ,$LF                ;;TYPE A LINE FEED
030504  012603                              MOV     (SP)+,R3            ;;RESTORE R3
030506  011646                              MOV     (SP),-(SP)          ;;ADJUST THE STACK AND PUT ADDRESS OF THE
030510  016666  000004  000002              MOV     4(SP),2(SP)         ;;      FIRST ASCII CHARACTER ON IT
030516  012766  030530  000004              MOV     #$TTYIN,4(SP)
030524  000002                              RTI                         ;;RETURN
030526     000                  9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
030527     000                          .BYTE   0                       ;;TERMINATOR
030530                          $TTYIN: .BLKB   7.                      ;;RESERVE 7. BYTES FOR TTY INPUT
030537     136     125     015  $CNTLU: .ASCIZ  /^U/<15><12>            ;;CONTROL "U"
030544     136     107     015  $CNTLG: .ASCIZ  /^G/<15><12>            ;;CONTROL "G"
030551     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
030562     040     040     116  $MNEW:  .ASCIZ  /  NEW = /
                                .EVEN
```

```
5917                                    .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
                                        ;

030574  011646                 $RDDEC: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR
030576  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE INPUT NUMBER
030604  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
030606  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
030610  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
030612  104411                 1$:     RDLIN                   ;;READ AN ASCIZ LINE
030614  012600                         MOV     (SP)+,R0        ;;ADDRESS OF 1ST CHAR.
030616  010037  030742                 MOV     R0,6$           ;;SAVE INCASE OF BAD INPUT
030622  005046                         CLR     -(SP)           ;;CLEAR DATA WORD
030624  005002                         CLR     R2              ;;SIGN SET POSITIVE
030626  122710  000055                 CMPB    #'-,(R0)        ;;SEE IF A MINUS SIGN WAS TYPED
030632  001001                         BNE     2$              ;;BR IF NO MINUS SIGN
030634  112002                         MOVB    (R0)+,R2        ;;SAVE FOR LATER USE
030636  112001                 2$:     MOVB    (R0)+,R1        ;;PICKUP THIS CHARACTER
030640  001424                         BEQ     3$              ;;GET OUT IF ZERO
030642  122701  000060                 CMPB    #'0,R1          ;;MAKE SURE THIS CHARACTER
030646  003032                         BGT     5$              ;;IS A DIGIT BETWEEN 0 & 9
03065C  122701  000071                 CMPB    #'9,R1
030654  002427                         BLT     5$
030656  032716  170000                 BIT     #^C7777,(SP)    ;;DON'T LET NUMBER GET TO BIG
030662  001024                         BNE     5$              ;;BR IF NUMBER WOULD OVERFLOW
030664  006316                         ASL     (SP)            ;;*2
030666  011646                         MOV     (SP),-(SP)      ;;SAVE FOR LATER
030670  006316                         ASL     (SP)            ;;*4
030672  006316                         ASL     (SP)            ;;*8.
030674  062616                         ADD     (SP)+,(SP)      ;;*10.
030676  102416                         BVS     5$              ;;OVERFLOW ISN'T ALLOWED
030700  162701  000060                 SUB     #'0,R1          ;;STRIP AWAY THE ASCII JUNK
030704  060116                         ADD     R1,(SP)         ;;ADD IN THIS DIGIT
030706  102412                         BVS     5$              ;;OVERFLOW ISN'T ALLOWED
030710  000752                         BR      2$              ;;LOOP
030712  005702                 3$:     TST     R2              ;;CHECK IF NUMBER IS NEG
030714  001401                         BEQ     4$              ;;BR IF NO
030716  005416                         NEG     (SP)            ;;YES--NEGATE THE NUMBER
030720  012666  000012         4$:     MOV     (SP)+,12(SP)    ;;SAVE THE RESULT
030724  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
030726  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
030730  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
030732  000002                         RTI                     ;;RETURN

030734  005726                 5$:     TST     (SP)+           ;;CLEAN PARTIAL NUMBER FROM STACK
030736  105010                         CLRB    (R0)            ;;SET A TERMINATOR
030740  104401                         TYPE                    ;;TYPE THE INPUT UP TO BAD CHAR.
```

READ A DECIMAL NUMBER FROM THE TTY

```
030742  000000            6$:     .WORD   0               ;;POINTER GOES HERE
030744  104401  001404            TYPE    ,$QUES          ;;"?" "CR" &"LF"
030750  000720                    BR      1$              ;;TRY AGAIN
```

READ AN OCTAL NUMBER FROM THE TTY

```
5919                                    .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

030752  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
030754  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
030762  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
030764  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
030766  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
030770  104411                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
030772  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
030774  010037  031100                  MOV     R0,5$           ;;AND SAVE IT
031000  005001                          CLR     R1              ;;CLEAR DATA WORD
031002  005002                          CLR     R2
031004  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
031006  001420                          BEQ     3$              ;;IF ZERO GET OUT
031010  122716  000060                  CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
031014  003026                          BGT     4$              ;;IS AN OCTAL DIGIT
031016  122716  000067                  CMPB    #'7,(SP)
031022  002423                          BLT     4$
031024  006301                          ASL     R1              ;;*2
031026  006102                          ROL     R2
031030  006301                          ASL     R1              ;;*4
031032  006102                          ROL     R2
031034  006301                          ASL     R1              ;;*8
031036  006102                          ROL     R2
031040  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
031044  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
031046  000756                          BR      2$              ;;LOOP
031050  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
031052  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
031056  010237  031110                  MOV     R2,$HIOCT
031062  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
031064  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
031066  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
031070  000002                          RTI                     ;;RETURN
031072  005726                  4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
031074  105010                          CLRB    (R0)            ;;SET A TERMINATOR
031076  104401                          TYPE                    ;;TYPE UP THRU THE BAD CHAR.
031100  000000                  5$:     .WORD   0
031102  104401  001404                  TYPE    ,$QUES          ;;"?" "CR" & "LF"
031106  000730                          BR      1$              ;;TRY AGAIN
031110  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
```

```
                        .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.
031112  010046          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
031114  016600  000002          MOV     2(SP),R0        ;;GET TRAP ADDRESS
031120  005740                  TST     -(R0)           ;;BACKUP BY 2
031122  111000                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
031124  006300                  ASL     R0              ;;POSITION FOR INDEXING
031126  016000  031146          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
031132  000200                  RTS     R0              ;;GO TO ROUTINE
                        ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
031134  011646          $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
031136  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
031144  000002                  RTI                     ;;RESTORE THE PSW
                        .SBTTL  TRAP TABLE
                        ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                        ;*BY THE "TRAP" INSTRUCTION.
                        ;       ROUTINE
                        ;       -------
031146  031134  $TRPAD: .WORD   $TRAP2
031150  026374          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
031152  026766          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
031154  026742          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
031156  027002          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
031160  027200          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
031162  030064          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
031164  027764          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
031166  030302          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
031170  030422          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
031172  030752          $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
031174  030574          $RDDEC  ;;CALL=RDDEC    TRAP+13(104413) READ A DECIMAL NUMBER FROM TTY
5922 031176  027170          $TYPDE  ;;CALL=TYPDE    TRAP+14(104414) TYPE DECIMAL EXTENDED
```

```
5924                                    .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

031200  032777  001000  150132  $SCOPE: BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
031206  001406                          BEQ     1$              ;;BR IF NO
031210  005737  001312                  TST     $ERTTL          ;;SEE IF THERE WERE ANY ERRORS YET
031214  001403                          BEQ     1$              ;;BR IF NOT YET
031216  013716  001310                  MOV     $LPERR,(SP)     ;;FUDGE RETURN
031222  000002                          RTI                     ;;RETURN
031224  032777  040000  150106  1$:     BIT     #BIT14,@SWR     ;;LOOP ON TEST?
031232  001403                          BEQ     2$              ;;BR IF NO
031234  013716  001306                  MOV     $LPADR,(SP)     ;;FUDGE RETURN
031240  000002                          RTI                     ;;RETURN
031242  105737  002706          2$:     TSTB    EOPLOC          ;@@ TEST TO SEE IF EOP'S ARE TO PRINT
031246  001043                          BNE     6$              ;@@ BRANCH IF NOT
031250  105737  032555                  TSTB    CHARCT          ;@@ SEE IF A CHARACTER WAS INPUTED IN A PRINT ROUTINE
031254  001406                          BEQ     3$              ;@@ BRANCH TO CHECK KEYBOARD IF NOT
031256  113737  032555  002706          MOVB    CHARCT,EOPLOC   ;@@ MOVE THE CHARACTER TO EOPLOC
031264  105037  032555                  CLRB    CHARCT          ;@@ CLEAR THE LOCATION
031270  000411                          BR      4$              ;@@ BRANCH TO CHECK THE CHARACTER
031272  105777  150046          3$:     TSTB    @$TKS           ;@@ SEE IF EOP REQUESTED
031276  100055                          BPL     8$              ;@@ BRANCH IF NOT
031300  117737  150042  002706          MOVB    @$TKB,EOPLOC    ;@@ GET CHARACTER
031306  142737  000200  002706          BICB    #200,EOPLOC     ;@@ CLEAR PARITY BIT
031314  122737  000030  002706  4$:     CMPB    #30,EOPLOC      ;@@ SEE IF A (^X)
031322  001406                          BEQ     5$              ;@@ BRANCH IF IT IS
031324  113737  002706  032555          MOVB    EOPLOC,CHARCT   ;@@ MOVE CHARACTER BACK TO CHARCT FOR POSSIBLE FUTURE USE
031332  105037  002706                  CLRB    EOPLOC          ;@@ CLEAR THE LOCATION
031336  000435                          BR      8$              ;@@ BRANCH TO START SCOPE ROUTINE
031340  104401  032174          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

031344  105337  002706                  DECB    EOPLOC          ;@@ MAKE ^X ANOTHER CHARACTER AND
031350  105037  032555                  CLRB    CHARCT          ;@@ CLEAR ANY CHARACTER THAT MAY BE THERE
031354  000426                          BR      8$              ;@@ BRANCH TO START SCOPE ROUTINE
031356  105737  032555          6$:     TSTB    CHARCT          ;@@ SEE IF A CHARACTER WAS INPUTED IN THE TYPE ROUTINE
031362  001011                          BNE     7$              ;@@ GO TEST THE CHARACTER IF SO
031364  105777  147754                  TSTB    @$TKS           ;@@ SEE IF CHARACTER WAITING
031370  100020                          BPL     8$              ;@@ BRANCH IF NOT
031372  117737  147750  032555          MOVB    @$TKB,CHARCT    ;@@ MOVE THE CHARACTER FOR TESTING
031400  142737  000200  032555          BICB    #200,CHARCT     ;@@ CLEAR THE PARITY BIT
031406  122737  000030  032555  7$:     CMPB    #30,CHARCT      ;@@ SEE IF ANOTHER (^X) WAS INPUTED
031414  001006                          BNE     8$              ;@@ BRANCH IF NOT
031416  105037  002706                  CLRB    EOPLOC          ;@@ CLEAR EOPLOC TO RESUME EOP MESSAGES
031422  104401  032504                  TYPE    ,EOPRSM         ;@@ TYPE: 'EOP'S AND EOD'S WILL RESUME PRINTING'
```

```
031426  105037  032555                  CLRB    CHARCT          ;&& MAKE CHARACTER SOMETHING ELSE
031432  105737  032555          8$:     TSTB    CHARCT          ;&& SEE IF A CHARACTER IS WAITING
031436  001477                          BEQ     9$              ;&& BRANCH AROUND (^Y) TEST IF NOT
031440  122737  000031  032555          CMPB    #31,CHARCT      ;&& IS THIS A (^Y) (REQUEST FOR RUN SUMMARY)
031446  001073                          BNE     9$              ;&& BRANCH IF NOT
031450  022737  044652  044650          CMP     #CAPSTK,CAPNTR  ;SEE IF THERE IS ANY DATA TO PRINT
031456  001003                          BNE     11$             ;BRANCH TO PRINT IT IF THERE IS
031460  104401  036311                  TYPE    ,NODATA         ;TYPE:  'NO ERROR TOTALS TO REPORT'
031464  000462                          BR      17$             ;KICK OUT
031466  022737  047132  044650  11$:    CMP     #ENDSTK,CAPNTR  ;SEE IF STACK IS FULL OF BULL
031474  001002                          BNE     12$             ;BRANCH AROUND STACK IS FULL MESSAGE IF NOT
031476  104401  036120                  TYPE    ,STKIFL         ;TYPE:  'STACK IS FULL - DATA MAY HAVE BEEN LOST'
031502  104401  036173          12$:    TYPE    ,ERCHDR         ;TYPE:  'SUMMATION OF ERRORS SINCE BEGINNING OR LAST REPORT'
                                                                ;       'BOARD #   PASS #   ERRTTL'
031506  012700  044652                  MOV     #CAPSTK,R0      ;MOVE ADDRESS OF STACK TO PRINT TO R0
031512  012701  000020                  MOV     #16.,R1         ;MOVE 16 BOARDS TO SEARCH TO R0
031516  005037  001362                  CLR     $TMP1           ;CLEAR $TMP1, DEVICE POINTER
031522  021037  001362          13$:    CMP     (R0),$TMP1      ;SEE IF UNIT NUMBER IS TO PRINT
031526  001403                          BEQ     14$             ;BRANCH TO PRINT DATA IF SO
031530  062700  000010                  ADD     #10,R0          ;GO TO NEXT SET OF DATA AND
031534  000420                          BR      16$             ;GO SEE IF ANY MORE TO CHECK
031536  012046                  14$:    MOV     (R0)+,-(SP)     ;MOVE UNIT NUMBER TO STACK FOR PRINTING
031540  104405                          TYPDS                   ;GO TYPE UNIT NUMBER IN DECIMAL
031542  104401  037352                  TYPE    ,SPACES         ;TYPE 2 SPACES
031546  012046                          MOV     (R0)+,-(SP)     ;MOVE OVERFLOW PASS NUMBER TO STACK FOR PRINTING
031550  001002                          BNE     15$             ;BRANCH IF NON-ZERO
031552  104401  037352                  TYPE    ,SPACES         ;TYPE AN EXTRA 2 SPACES - NUMBER WILL NOT BE EXTENDED
031556  012046                  15$:    MOV     (R0)+,-(SP)     ;MOVE MAJOR PASS NUMBER TO STACK FOR PRINTING
031560  104414                          TYPDE                   ;GO TYPE PASS NUMBER IN EXTENDED DECIMAL
031562  104401  037352                  TYPE    ,SPACES         ;TYPE 2 SPACES
031566  012046                          MOV     (R0)+,-(SP)     ;MOVE ERROR TOTAL TO STACK FOR PRINTING
031570  104405                          TYPDS                   ;GO TYPE ERROR TOTAL IN DECIMAL
031572  104401  001405                  TYPE    ,$CRLF          ;TYPE <CRLF>
031576  020037  044650          16$:    CMP     R0,CAPNTR       ;SEE IF ALL DATA HAS BEEN PRINTED
031602  001347                          BNE     13$             ;BRANCH BACK IF NOT
031604  005237  001362                  INC     $TMP1           ;INCREMENT DEVICE POINTER
031610  012700  044652                  MOV     #CAPSTK,R0      ;INITIALIZE TO BEGINNING FOR ANOTHER POSSIBLE PASS
031614  005301                          DEC     R1              ;DECREMENT THE LOOP COUNTER AND
031616  001341                          BNE     13$             ;BRANCH UNTIL ALL DEVICE DATA PRINTED
031620  012737  044652  044650          MOV     #CAPSTK,CAPNTR  ;REINITIALIZE CAPNTR
031626  104401  001405                  TYPE    ,$CRLF          ;TYPE ANOTHER <CRLF>
031632  105037  032555          17$:    CLRB    CHARCT          ;CLEAR ANY CHARACTER ENTERED DURING PRINTING
031636  104407                  9$:     CKSWR                   ;;TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER
                                ;#####START OF CODE FOR THE XOR TESTER#####
031640  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE ''XOR'' TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A ''NOP'' (NOP=240)
031642  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
031646  012737  031666  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
031654  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
031660  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
031664  000517                          BR      $SVLAD          ;;GO TO THE NEXT TEST
031666  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
031670  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
031674  000505                          BR      7$              ;;LOOP ON THE PRESENT TEST
031676                          6$:;#####END OF CODE FOR THE XOR TESTER#####
031676  032777  000400  147434          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
031704  001505                          BEQ     4$              ;;BR IF NO
```

```
031706  005046                          CLR     -(SP)           ;;CLEAR A TEMP. LOCATION
031710  117716  147424                  MOVB    @SWR,(SP)       ;;PICKUP THE DESIRED TEST NUMBER
031714  042716  000200                  BIC     #$SWRMK,(SP)    ;;MASK OUT UNDESIRED BITS
031720  001416                          BEQ     8$              ;;BRANCH IF BAD TEST NUMBER IN SWR
031722  022716  000037                  CMP     #37,(SP)        ;;CHECK THE NUMBER IN THE SWR
031726  002413                          BLT     8$              ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
031730  011637  001302                  MOV     (SP),$TSTNM     ;;UPDATE THE TEST NUMBER IN $TSTNM
031734  011637  001414                  MOV     (SP),$TESTN     ;&& UPDATE THE TEST NUMBER IN $TESTN
031740  005316                          DEC     (SP)            ;;BACKUP BY ONE
031742  006316                          ASL     (SP)            ;;SCALE THE TEST NUMBER AS AN INDEX
031744  062716  032556                  ADD     #$SW08TBL,(SP)  ;;FORM THE ADDRESS OF TEST POINTER
031750  013637  001306                  MOV     @(SP)+,$LPADR   ;;SET LOOP ADDRESS TO DESIRED TEST
031754  000501                          BR      $OVER           ;;GO LOOP ON THE TEST
031756  005726                  8$:     TST     (SP)+           ;;CLEAN THE BAD TEST NUMBER OFF OF THE STACK
031760  022737  177777  032654  2$:2$:  CMP     #-1,CPSAVE      ;SEE IF TIMEOUT WAS PREVIOUSLY RECORDED ;DPM001
031766  001447                          BEQ     2001$           ;BRANCH AROUND ROUTINE IF SO            ;DPM001
031770  005227  177777                  INC     #-1             ;TEST FOR 1ST TIME THROUGH              ;DPM001
031774  001005                          BNE     900$            ;BRANCH IF NOT                          ;DPM001
031776  013746  000004                  MOV     4,-(SP)         ;SAVE TIMEOUT VECTOR AT 4               ;DPM001
032002  012737  032056  000004          MOV     #1999$,4        ;SET VECTOR TO LOCATION BELOW           ;DPM001
032010  013737  177766  032654  900$:   MOV     177766,CPSAVE   ;MOVE CPU ERR REG VALUE TO LOC FOR TST  ;DPM001
032016  032737  000001  032654          BIT     #BIT00,CPSAVE   ;SEE IF THE POWER MONITOR BIT IS ON     ;DPM001
032024  001423                          BEQ     2000$           ;BRANCH TO CONTINUE ROUTINE IF CLEAR    ;DPM001
032026  042737  000001  177766          BIC     #BIT00,177766   ;CLEAR THE BIT FOUND TO BE SET          ;DPM001
032034  017746  147300                  MOV     @SWR,-(SP)      ;SAVE SWR VALUE                         ;DPM001
032040  042777  001000  147272          BIC     #BIT09,@SWR     ;DON'T ALLOW LOOP ON ERROR ON THIS ERROR;DPM001
032046  104177                          EMT     +177            ;CALL SPECIAL POWER FAIL BIT ERROR CALL ;DPM001
032050  012677  147264                  MOV     (SP)+,@SWR      ;RESTORE SWR TO ORIGINAL VALUE          ;DPM001
032054  000407                          BR      2000$           ;BRANCH OVER TIMEOUT SECTION            ;DPM001
032056  022626                  1999$:  CMP     (SP)+,(SP)+     ;CLEAN UP THE STACK AFTER TIMEOUT       ;DPM001
032060  012737  177777  032654          MOV     #-1,CPSAVE      ;MOVE -1 TO CPSAVE - NO CPU ERR REG     ;DPM001
032066  005237  032076                  INC     2000$+2         ;INCREMENT 1ST TIME THROUGH LOCATION    ;DPM001
032072  000403                          BR      910$            ;BRANCH OVER 1ST TIME THROUGH TEST      ;DPM001
032074  005227  177777          2000$:  INC     #-1             ;TEST FOR 1ST TIME THROUGH              ;DPM001
032100  001002                          BNE     2001$           ;BRANCH IF NOT                          ;DPM001
032102  012637  000004          910$:   MOV     (SP)+,4         ;RESTORE LOCATION 4                     ;DPM001
032106                          2001$:  ;TSTB   $ERFLG          ;;HAS AN ERROR OCCURRED? ;&& ELIMINATED FOR CZDRLD
032106  001406                          BEQ     $SVLAD          ;;BR IF NO
032110  013737  001310  001306  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
032116  000420                          BR      $OVER
032120  105037  001303          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
032124  105237  001302          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
032130  113737  001302  001414          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
032136  011637  001306                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
032142  011637  001310                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
032146  005037  001376                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
032152  112737  000001  001315          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
032160  013777  001302  147154  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
032166  013716  001306                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
032172  000002                          RTI                     ;;RETURN
032174     136     130     200  HAKTPM: .ASCII  /^X/<CRLF>/HIT ANY KEY TO OBTAIN A PROGRESS REPORT,/<CRLF> ;&&
032250     105     116     124          .ASCII  /ENTER (^X) TO RESUME EOP'S AND EOD'S/<CRLF>                 ;&&
032315     105     116     124          .ASCII  /ENTER THE KEY REPEATEDLY, AS RESETS DONE IN THE DIAGNOSTIC/<CRLF> ;&&
032410     115     101     131          .ASCIZ  /MAY CLEAR THE CHARACTER BEFORE THE TESTS FOR THE CHARACTER/<CRLF> ;&&
032504     136     130     200  EOPRSM: .ASCIZ  /^X/<CRLF>/EOP'S AND EOD'S WILL RESUME PRINTING/<CRLF>      ;&&
032555     000                  CHARCT: .BYTE   0               ;&& LOCATION TO HOLD INPUTED CHARACTER
                                        .EVEN
```

```
032556                          $SW08TBL:
032556  012636                          .WORD   TS11+30         ;STARTING ADDRESS+30 OF TEST 1
032560  013034                          .WORD   TST2+30         ;STARTING ADDRESS+30 OF TEST 2
032562  013164                          .WORD   TST3+30         ;STARTING ADDRESS+30 OF TEST 3
032564  013516                          .WORD   TST4+30         ;STARTING ADDRESS+30 OF TEST 4
032566  014222                          .WORD   TST5+30         ;STARTING ADDRESS+30 OF TEST 5
032570  014416                          .WORD   TST6+30         ;STARTING ADDRESS+30 OF TEST 6
032572  014764                          .WORD   TST7+30         ;STARTING ADDRESS+30 OF TEST 7
032574  015142                          .WORD   TST10+30        ;STARTING ADDRESS+30 OF TEST 10
032576  015376                          .WORD   TST11+30        ;STARTING ADDRESS+30 OF TEST 11
032600  015562                          .WORD   TST12+30        ;STARTING ADDRESS+30 OF TEST 12
032602  015746                          .WORD   TST13+30        ;STARTING ADDRESS+30 OF TEST 13
032604  016132                          .WORD   TST14+30        ;STARTING ADDRESS+30 OF TEST 14
032606  016322                          .WORD   TST15+30        ;STARTING ADDRESS+30 OF TEST 15
032610  016554                          .WORD   TST16+30        ;STARTING ADDRESS+30 OF TEST 16
032612  017012                          .WORD   TST17+30        ;STARTING ADDRESS+30 OF TEST 17
032614  017410                          .WORD   TST20+30        ;STARTING ADDRESS+30 OF TEST 20
032616  020014                          .WORD   TST21+30        ;STARTING ADDRESS+30 OF TEST 21
032620  020330                          .WORD   TST22+30        ;STARTING ADDRESS+30 OF TEST 22
032622  020626                          .WORD   TST23+30        ;STARTING ADDRESS+30 OF TEST 23
032624  021364                          .WORD   TST24+30        ;STARTING ADDRESS+30 OF TEST 24
032626  021666                          .WORD   TST25+30        ;STARTING ADDRESS+30 OF TEST 25
032630  022156                          .WORD   TST26+30        ;STARTING ADDRESS+30 OF TEST 26
032632  022434                          .WORD   TST27+30        ;STARTING ADDRESS+30 OF TEST 27
032634  022652                          .WORD   TST30+30        ;STARTING ADDRESS+30 OF TEST 30
032636  023140                          .WORD   TST31+30        ;STARTING ADDRESS+30 OF TEST 31
032640  023270                          .WORD   TST32+30        ;STARTING ADDRESS+30 OF TEST 32
032642  023650                          .WORD   TST33+30        ;STARTING ADDRESS+30 OF TEST 33
032644  024154                          .WORD   TST34+30        ;STARTING ADDRESS+30 OF TEST 34
032646  024536                          .WORD   TST35+30        ;STARTING ADDRESS+30 OF TEST 35
032650  025034                          .WORD   TST36+30        ;STARTING ADDRESS+30 OF TEST 36
032652  025350                          .WORD   TST37+30        ;STARTING ADDRESS+30 OF TEST 37
032654  000000                  CPSAVE: .WORD   0               ;LOCATION TO SAVE CPU ERR REG CONTENTS  ;DPM001
```

```
5926                                    .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

032656  105037  033224          $ERROR: CLRB    IBSAVE          ;CLEAR THE ITEM BYTE SAVE LOCATION      ;RAN001
032662  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
032664  105237  001303          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
032670  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
032672  013777  001302  146442          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
032700  032777  002000  146432          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
032706  001402                          BEQ     1$              ;;NO - SKIP
032710  104401  001400                  TYPE    ,$BELL          ;;RING BELL
032714  005237  001312          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
032720  005237  002714                  INC     ERRCNT          ;&& INCREMENT THE ERROR COUNT
032724  011637  001316                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
032730  162737  000002  001316          SUB     #2,$ERRPC
032736  117737  146354  001314          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
032744  122737  000177  001314          CMPB    #177,$ITEMB     ;SEE IF THIS IS THE POWER FAIL CALL      ;DPM001
032752  001445                          BEQ     1001$           ;BRANCH AROUND ROUTINE IF IT IS         ;DPM001
032754  105737  033224                  TSTB    IBSAVE          ;SEE IF THIS IS THE 2ND ERROR CALL      ;DPM001
032760  001040                          BNE     1000$           ;BRANCH IF SO                           ;DPM001
032762  022737  177777  032654          CMP     #-1,CPSAVE      ;SEE IF CPU ERR REG EXISTS              ;DPM001
032770  001436                          BEQ     1001$           ;BRANCH IF NOT                          ;DPM001
032772  013746  000004                  MOV     4,-(SP)         ;SAVE TIMEOUT VECTOR AT 4               ;DPM001
032776  012737  033014  000004          MOV     #5000$,4        ;TIMEOUTS TO 5000$                      ;DPM001
033004  013737  177766  032654          MOV     177766,CPSAVE   ;MOVE CPU ERR REG TO CPSAVE FOR TEST    ;DPM001
033012  000405                          BR      4999$           ;BRANCH OVER TIMEOUT HANDLER            ;DPM001
033014  022626                  5000$:  CMP     (SP)+,(SP)+     ;CLEAN STACK AFTER TIMEOUT              ;DPM001
033016  012737  177777  032654          MOV     #-1,CPSAVE      ;SET CPSAVE TO -1 TO NEGATE FURTHER TEST;DPM001
033024  000420                          BR      1001$           ;BRANCH TO CONTINUE ERROR CALL          ;DPM001
033026  032737  000001  032654  4999$:  BIT     #BIT00,CPSAVE   ;SEE IF POWER MONITOR BIT IS SET        ;DPM001
033034  001414                          BEQ     1001$           ;BRANCH IF OK                           ;DPM001
033036  042737  000001  177766          BIC     #BIT00,177766   ;CLEAR THE BIT FOUND SET                 ;DPM001
033044  113737  001314  033224          MOVB    $ITEMB,IBSAVE   ;MAKE IBSAVE NON-ZERO FOR DUAL CALL     ;DPM001
033052  112737  000177  001314          MOVB    #177,$ITEMB     ;SET $ITEMB TO SPECIAL POWER FAIL PNTR  ;DPM001
033060  000402                          BR      1001$           ;BRANCH OVER IBSAVE CLEARING            ;DPM001
033062  105037  033224          1000$:  CLRB    IBSAVE          ;CLEAR IBSAVE SO AFTER 2ND ERROR, EXIT  ;DPM001
033066  012637  000004          1001$:  MOV     (SP)+,4         ;RESTORE STACK TO PRETEST STATE         ;DPM001
033072  032777  020000  146240          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
033100  001002                          BNE     20$             ;;SKIP TYPEOUTS
033102  004737  033226                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
033106                          20$:
033106  122737  000001  001430          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
033114  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
033116  113737  001314  033130          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
033124  004737  033644                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
033130     000                  21$:    .BYTE   0
```

ERROR HANDLER ROUTINE

```
033131     000                          .BYTE   0
033132  000777              22$:        BR      22$             ;;APT ERROR LOOP
033134  105737  033224      2$:         TSTB    IBSAVE          ;SEE IF POWER FAIL ERROR CALL           ;RAN001
033140  001005                          BNE     3$              ;BRANCH IF NOT - HALT NOT ALLOWED       ;RAN001
033142  005777  146172                  TST     @SWR            ;;HALT ON ERROR
033146  100002                          BPL     3$              ;;SKIP IF CONTINUE
033150  000000                          HALT                    ;;HALT ON ERROR!
033152  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
033154  005737  001376      3$:         TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
033160  001402                          BEQ     4$              ;;BR IF NONE
033162  013716  001376                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
033166  032777  001000  146144  4$:     BIT     #BIT9,@SWR      ;&& SEE IF WE ARE TO LOOP ON ERROR
033174  001402                          BEQ     5$              ;&& BRANCH OUT IF NOT
033176  013716  001310                  MOV     $LPERR,(SP)     ;&& FUDGE RETURN
033202                      5$:
033202  022737  026324  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
033210  001001                          BNE     6$              ;;BRANCH IF NO
033212  000000                          HALT                    ;;YES
033214                      6$:
033214  105737  033224                  TSTB    IBSAVE          ;SEE IF THIS IS THE PWR FAIL ERROR CALL ;RAN001
033220  001221                          BNE     7$              ;BRANCH BACK TO CALL ORIGINAL ERR IF SO ;RAN001
033222  000002                          RTI                     ;;RETURN
033224  000000              IBSAVE:     .WORD   0               ;LOC'N TO HOLD $ITEMB DURING DUAL ERR   ;RAN001
```

```
5928                                    .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.

033226                                  $ERRTYP:
033226  010046                                MOV     R0,-(SP)        ;;SAVE R0
033230  113700  001314                        MOVB    @#$ITEMB,R0     ;;PICKUP THE ITEM INDEX
033234  042700  177400                        BIC     #177400,R0      ;CLEAR UPPER BYTE                               ;DPM001
033240  001004                                BNE     1$              ;;IF ITEM NUMBER IS ZERO, TYPE THE PC OF THE ERROR
033242  013746  001316                        MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                        ;;ERROR ADDRESS
033246  104402                                TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
033250  000517                                BR      14$             ;;GET OUT
033252  022700  000177          1$:1$:        CMP     #177,R0         ;SEE IF THIS CALL IS THE PWR FAIL CALL  ;DPM001
033256  001003                                BNE     100$            ;BRANCH IF NOT                          ;DPM001
033260  012700  033516                        MOV     #PFECH,R0       ;MOVE ADDRESS OF PWR FAIL CALL TO R0    ;DPM001
033264  000460                                BR      110$            ;BRANCH TO CALL ERROR                   ;DPM001
033266  010037  001360          100$:         MOV     R0,$TMP0        ;&& MOVE R0 TO $TMP0 FOR 200 TEST
033272  042700  000200                        BIC     #200,R0         ;&& CLEAR BIT 7 IF PRESENT
033276  005300                                DEC     R0              ;&& MAKE POINTER AN INDEX
033300  006300                                ASL     R0              ;&& SHIFT TO MULTIPLY BY 10 (OCTAL)
033302  006300                                ASL     R0              ;&&
033304  006300                                ASL     R0              ;&&
033306  105737  001360                        TSTB    $TMP0           ;&& SEE IF ITEM NUMBER IS OVER 200
033312  100037                                BPL     4$              ;&& BRANCH IF ITEM NUMBER IS LESS THAN 200
033314  023727  002714  000020              CMP     ERRCNT,#20      ;&& SEE IF 20 (OCTAL) ERRORS HAVE PRINTED
033322  002404                                BLT     2$              ;&& BRANCH TO PRINT THE ERROR IF LESS
033324  003071                                BGT     14$             ;&& BRANCH TO RETURN IF GREATER - NO MORE DATA IS TO PRINT
033326  104401  040775                        TYPE    ,NOMORE         ;&& TYPE MESSAGE ANNOUNCING NO MORE PRINTING OF ERRORS
033332  000466                                BR      14$             ;&& BRANCH TO RETURN
033334  022737  000001  002714  2$:           CMP     #1,ERRCNT       ;&& SEE IF THIS IS THE FIRST ERROR
033342  001415                                BEQ     3$              ;&& BRANCH IF IT WAS AND GO TYPE ERROR MESSAGE
033344  123737  001360  033514              CMPB    $TMP0,MEPITM    ;&& SEE IF ITEM MATCHES LAST MULTIPLE ERROR
033352  001011                                BNE     3$              ;&& BRANCH IF NOT - NEW HEADER NEEDED
033354  032777  000200  145756              BIT     #BIT7,@SWR      ;&& SEE IF SWITCH REGISTER BIT 7 IS SET
033362  001052                                BNE     14$             ;&& BRANCH TO RETURN IF SWITCH SET
033364  042700  177400                        BIC     #177400,R0      ;&& CLEAR UPPER BYTE OF R0 EXPOSING ITEM BYTE
033370  062700  002300                        ADD     #ER200+4,R0     ;&& POINT TO DATA TABLE ENTRY
033374  000432                                BR      9$              ;&& BRANCH TO PRINT DATA
033376  113737  001360  033514  3$:           MOVB    $TMP0,MEPITM    ;&& MOVE ITEM NUMBER TO MEPITM FOR POSSIBLE FUTURE USE
033404  062700  000540                        ADD     #ER200-$ERRTB,R0;&& ADD 200 BASE POINTER TO R0 AND
033410  000402                                BR      5$              ;&& BRANCH AROUND ERRCNT CLEAR
033412  005037  002714          4$:           CLR     ERRCNT          ;&& CLEAR ERRCNT SO MULTIPLE ERRORS GET NEW HEADER
033416  104401  001405          5$:           TYPE    ,$CRLF          ;&& TYPE  <CRLF>
033422  062700  001534                        ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
033426  012037  033436          110$:         MOV     (R0)+,6$        ;;PICKUP "ERROR MESSAGE" POINTER
033432  001404                                BEQ     7$              ;;SKIP TYPEOUT IF NO POINTER
033434  104401                                TYPE                    ;;TYPE THE "ERROR MESSAGE"
033436  000000          6$:                   .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
033440  104401  001405                        TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
033444  012037  033454          7$:           MOV     (R0)+,8$        ;;PICKUP "DATA HEADER" POINTER
033450  001404                                BEQ     9$              ;;SKIP TYPEOUT IF 0
033452  104401                                TYPE                    ;;TYPE THE "DATA HEADER"
033454  000000          8$:                   .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
```

ERROR MESSAGE TYPEOUT ROUTINE

```
033456  104401  001405                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
033462  011000                  9$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
033464  001407                          BEQ     13$             ;;GO AROUND ROUTINE TO TYPE THE DATA IF NONE
033466  013046                  10$:    MOV     @(R0)+,-(SP)    ;;PUT OCTAL DATA ON STACK FOR TYPING
033470  104402                          TYPOC                   ;;TYPE AN OCTAL NUMBER
033472  005710                          TST     (R0)            ;;IS THERE ANOTHER NUMBER?
033474  001403                          BEQ     13$             ;;BR IF NO
033476  104401  037352                  TYPE    ,SPACES         ;;TYPE TWO(2) SPACES
033502  000771                          BR      10$             ;;GO BACK TO PRINT THE OCTAL NUMBER
033504  104401  001405          13$:    TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
033510  012600                  14$:    MOV     (SP)+,R0        ;;RESTORE R0
033512  000207                          RTS     PC              ;;RETURN
033514  000000                  MEPITM: .WORD   0               ;&& LOCATION TO STORE 200+ ERROR ITEM NUMBER

033516  033526  033562  033612  PFECH:  .WORD   PFECH1,PFECH2,PFECH3,PFECH4 ;ADDRESSES OF DATA/ASCII BELOW
033526     120     117     127  PFECH1: .ASCIZ  ?POWER MONITOR BIT FOUND SET?
033562     124     105     123  PFECH2: .ASCIZ  ?TES NO  ERR PC  CPUERR?
                                .EVEN
033612  001414  001316  032654  PFECH3: .WORD   $TESTN,$ERRPC,CPSAVE,0
033622     000     000     000  PFECH4: .BYTE   0,0,0,0
```

```
5930                                    .SBTTL  APT COMMUNICATIONS ROUTINE
                                        ;;*****************************************************************
033626  112737  000001  034072  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
033634  112737  000001  034070  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
033642  000403                          BR      $ATYC
033644  112737  000001  034072  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
033652                          $ATYC:
033652  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
033654  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
033656  105737  034070                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
033662  001450                          BEQ     5$              ;;IF NOT:  BR
033664  122737  000001  001430          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
033672  001031                          BNE     3$              ;;IF NOT:  BR
033674  132737  000100  001431          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
033702  001425                          BEQ     3$              ;;IF NOT:  BR
033704  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
033710  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
033716  005737  001410          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
033722  001375                          BNE     1$              ;;IF NOT:  WAIT
033724  010037  001424                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
033730  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
033732  001376                          BNE     2$
033734  163700  001424                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
033740  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
033742  010037  001426                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
033746  012737  000004  001410          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
033754  000413                          BR      5$
033756  017637  000004  034002  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
033764  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
033772  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
033776  004737  026374                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
034002  000000                  4$:     .WORD   0
034004                          5$:
034004  105737  034072          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
034010  001416                          BEQ     12$             ;;IF NOT:  BR
034012  005737  001430                  TST     $ENV            ;;RUNNING UNDER APT?
034016  001413                          BEQ     12$             ;;IF NOT:  BR
034020  005737  001410          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
034024  001375                          BNE     11$             ;;IF NOT:  WAIT
034026  017637  000004  001412          MOV     @4(SP),$FATAL   ;;GET ERROR #
034034  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
034042  005237  001410                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
034046  105037  034072          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
034052  105037  034071                  CLRB    $LFLG           ;;CLEAR LOG FLAG
034056  105037  034070                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
034062  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
034064  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
034066  000207                          RTS     PC              ;;RETURN
034070     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
034071     000          $LFLG:  .BYTE   0               ;;LOG FLAG
034072     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
                                .EVEN
            000200      APTSIZE=200
            000001      APTENV=001
            000100      APTSPOOL=100
            000040      APTCSUP=040
```

```
5932                                    .SBTTL  POWER DOWN AND UP ROUTINE

                                        ;;*****************************************************************
                                        ;POWER DOWN AND UP ROUTINE
034074  012737  034112  000024  $PWRDN: MOV     #$PWRUP,PWRVEC  ;@@ SET UP VECTOR TO RETURN TO THE HALT BELOW
034102  012737  000340  000026          MOV     #LEVEL7,PWRVEC+2;@@ RETURN PRIORITY TO 7
034110  000000                          HALT                    ;;HALT PROCESSOR
034112  012737  034074  000024  $PWRUP: MOV     #$PWRDN,PWRVEC  ;@@ RESET PWRVEC TO PWRDN ROUTINE AND
034120  012737  000340  000026          MOV     #LEVEL7,PWRVEC+2;@@ PRIORITY TO 7
034126  012706  001300                  MOV     #STACK,SP       ;@@ REINITIALIZE THE STACK,
034132  012746  000340                  MOV     #LEVEL7,-(SP)   ;@@ SET UP RETURN PRIORITY TO 7 AND
034136  012746  011420                  MOV     #START1,-(SP)   ;@@ MOVE START1 ADDRESS TO STACK AND
034142  005037  001360                  CLR     $TMP0           ;@@ CLEAR WAIT LOOP COUNTER
034146  005237  001360          1$:     INC     $TMP0           ;@@ GIVE TTY TIME TO RECOVER FROM POWER FAILURE
034152  001375                          BNE     1$              ;@@ BRANCH BACK UNTIL ZERO AGAIN
034154  104401  034162                  TYPE    ,$POWER         ;@@ TYPE THE POWER FAILURE MESSAGE ASCIZED BELOW
034160  000002                          RTI                     ;@@ RETURN TO PROGRAM
034162     200     120     117  $POWER: .ASCIZ  <CRLF>/POWER FAILURE - RESTARTING PROGRAM/<CRLF>
                                        .EVEN
```

```
                                        .SBTTL  MULTIPLE BOARD DIALOGUE ROUTINE
                                ;;*****************************************************************
                                ;>>>>>>>MULTIPLE BOARD DIALOGUE ROUTINE<<<<<<<
                                ;;*****************************************************************
034230  012706  001300          MBD:    MOV     #STACK,SP       ;INITIALIZE THE STACK
034234  004737  006610                  JSR     PC,SETTUP       ;GO INITIALIZE THE COMMON TAGS
034240  104401  040643                  TYPE    ,MBDIAL         ;TYPE:  'MULTIPLE BOARD DIALOGUE'
034244  105037  032555          PROMPT: CLRB    CHARCT          ;CLEAR LOCATION FOR POSSIBLE INPUT DURING PRINT
034250  104401  040676                  TYPE    ,ECELR          ;TYPE:  'ENTER COMMAND ([E]DIT, [L]IST, [B]URST CALIBRATION,
034254  012703  000001                  MOV     #1,R3           ;EXPECT 1 CHARACTER
034260  004737  002722                  JSR     PC,READ         ;GO READ 1 CHARACTER
034264  022737  000114  002664          CMP     #'L,ANSWER      ;LIST PRESENT TABLE?
034272  001073                          BNE     1$              ;BRANCH IF NO
034274  104401  040452                  TYPE    ,HEADER         ;TYPE:  '# OF    START                                  2-N
                                                                ;       'BOARDS  REGADR  VECADR   W-B     P-LEV   CYCLE    T
034300  013737  001474  002720          MOV     $DDW0,DDW       ;SET UP THE DEVICE DESCRIPTOR WORD FOR PRINTING
034306  013746  002414                  MOV     QTYBRD,-(SP)    ;MOVE NUMBER OF DEVICES TO STACK FOR TYPING
034312  104405                          TYPDS                   ;TYPE THE NUMBER OF DEVICES
034314  104401  037355                  TYPE    ,SPACE3         ;TYPE 3 SPACE CHARACTERS
034320  013746  002416                  MOV     REGADR,-(SP)    ;MOVE THE DEVICE REGISTER ADDRESS TO THE STACK
034324  104402                          TYPOC                   ;TYPE THE DEVICE REGISTER ADDRESS
034326  104401  037361                  TYPE    ,SPACE6         ;TYPE 6 SPACE CHARACTERS
034332  013746  002456                  MOV     VECADR,-(SP)    ;MOVE THE DEVICE VECTOR ADDRESS TO THE STACK
034336  104403                          TYPOS                   ;TYPE THE DEVICE VECTOR ADDRESS
034340     003     000                  .BYTE   3,0             ;TYPE 3 CHARACTERS, LEADING ZEROS SUPPRESSED
034342  104401  037370                  TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
034346  032737  000001  002720          BIT     #BIT0,DDW       ;SEE WHICH W/B STATE FOR BOARDS
034354  001403                          BEQ     10$             ;GO PRINT W STATE IF W
034356  104401  037405                  TYPE    ,B              ;TYPE A 'B'
034362  000402                          BR      11$             ;GO TO NEXT CHECK
034364  104401  037407          10$:    TYPE    ,W              ;TYPE A 'W'
034370  104401  037370          11$:    TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
034374  004737  006560                  JSR     PC,PNTPRI       ;PRINT DEVICE PRIORITY
034400  104401  037370                  TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
034404  032737  000002  002720          BIT     #BIT1,DDW       ;SEE WHICH 2/N STATE FOR BOARDS
034412  001003                          BNE     12$             ;GO PRINT N STATE IF N
034414  104401  037420                  TYPE    ,TWO            ;TYPE A '2'
034420  000402                          BR      13$             ;GO TO NEXT CHECK
034422  104401  037422          12$:    TYPE    ,N              ;TYPE AN 'N'
034426  104401  037370          13$:    TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
034432  032737  000004  002720          BIT     #BIT2,DDW       ;SEE WHICH CABLE STATE FOR BOARDS
034440  001403                          BEQ     14$             ;GO PRINT NO CABLE IF NONE
034442  104401  037414                  TYPE    ,YES            ;TYPE 'YES'
034446  000402                          BR      15$             ;BRANCH TO CONTINUE
034450  104401  037411          14$:    TYPE    ,NO             ;TYPE 'NO'
034454  104401  037435          15$:    TYPE    ,CRLF2          ;TYPE 2 <CRLF>'S
034460  000671                          BR      PROMPT          ;BRANCH TO PROMPT ANOTHER COMMAND
034462  022737  000122  002664  1$:     CMP     #'R,ANSWER      ;RUN PROGRAM?
034470  001020                          BNE     4$              ;BRANCH IF NOT
034472  005737  002416                  TST     REGADR          ;SEE IF REGADR HAS BEEN LOADED
034476  001003                          BNE     3$              ;BRANCH TO CHECK VECADR IF SO
034500  104401  037072          2$:     TYPE    ,MUSTED         ;TYPE:  'DEVICE ADDRESS AND/OR VECTOR TABLE NOT SET UP - MUS
034504  000657                          BR      PROMPT          ;BRANCH BACK FOR PROMPT MESSAGE
034506  005737  002456          3$:     TST     VECADR          ;SEE IF VECADR HAS BEEN LOADED
034512  001772                          BEQ     2$              ;BRANCH BACK TO PRINT ERROR MESSAGE IF NOT
034514  004737  003170                  JSR     PC,FIXTBL       ;FILL TABLE
034520  012737  177777  002670          MOV     #-1,MANSIZE     ;MOVE -1 TO MANSIZE TO INDICATE WE HAVE MANUALLY SIZED
```

```
 5991 034526  000137  011424                    JMP     START           ;JUMP TO START
 5992 034532  022737  000105  002664  4$:       CMP     #'E,ANSWER      ;EDIT TABLE?
 5993 034540  001414                            BEQ     EDIT            ;BRANCH TO EDIT IF SO
 5994 034542  022737  000102  002664            CMP     #'B,ANSWER      ;ENTER BURST DATA LATE CALIBRATION?
 5995 034550  001235                            BNE     PROMPT          ;BRANCH TO PROMPT IF COMMAND NOT RECOGNIZED
 5996 034552  005737  002416                    TST     REGADR          ;SEE IF REGADR HAS BEEN LOADED
 5997 034556  001750                            BEQ     2$              ;BRANCH TO ERROR MESSAGE IF NOT
 5998 034560  005737  002456                    TST     VECADR          ;SEE IF VECADR HAS BEEN LOADED
 5999 034564  001745                            BEQ     2$              ;BRANCH TO ERROR MESSAGE IF NOT
 6000 034566  000137  035712                    JMP     BDLCR           ;JUMP TO BURST DATA LATE CALIBRATION ROUTINE
```

```
6001                                            .SBTTL  TABLE EDIT ROUTINE
6002 034572  104401  037535          EDIT:     TYPE    ,NOBUT          ;TYPE:  'NUMBER OF BOARDS UNDER TEST: '
6003 034576  013746  002414                    MOV     QTYBRD,-(SP)    ;PUT QUANTITY OF BOARDS ON STACK FOR PRINTING
6004 034602  104405                            TYPDS                   ;GO PRINT THE QUANTITY OF BOARDS
6005 034604  104401  037400                    TYPE    ,SPACEC         ;TYPE:  '  : '
6006 034610  012703  000002                    MOV     #2,R3           ;EXPECT MAX OF 2 CHARACTERS
6007 034614  112737  000071  002704            MOVB    #'9,LRGSTC      ;MOVE ASCII '9' TO THE LARGEST CHARACTER LOCATION
6008 034622  004737  002722                    JSR     PC,READ         ;GO READ 2 CHARACTERS
6009 034626  022703  000002                    CMP     #2,R3           ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
6010 034632  001017                            BNE     2$              ;BRANCH IF NOT
6011 034634  022737  000033  002664            CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
6012 034642  001453                            BEQ     5$              ;BRANCH TO PROMPT JUMP IF SO
6013 034644  022737  000003  002664            CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
6014 034652  001447                            BEQ     5$              ;BRANCH TO PROMPT JUMP IF EXIT REQUESTED
6015 034654  022737  000015  002664            CMP     #CARETN,ANSWER  ;SEE IF A <CR> WAS INPUTED
6016 034662  001412                            BEQ     4$              ;IF <CR> USE EXISTING NUMBER
6017 034664  104401  036407          1$:       TYPE    ,BDNERR         ;TYPE:  'ILLEGAL NUMBER (# OTHER THAN 1-16) OR CHARACTER IN
6018 034670  000740                            BR      EDIT            ;BRANCH BACK FOR NEW INPUT
6019 034672  005704                  2$:       TST     R4              ;CHECK FOR ZERO MODULES INPUT
6020 034674  001773                            BEQ     1$              ;BRANCH TO PRINT ERROR MESSAGE IF SO
6021 034676  022704  000020                    CMP     #20,R4          ;SFE IF BOARD NUMBER IS ILLEGAL
6022 034702  100770                            BMI     1$              ;BRANCH TO PRINT ERROR MESSAGE IF SO
6023 034704  010437  002414            3$:     MOV     R4,QTYBRD       ;MOVE INPUTED NUMBER TO QTYBRD
6024 034710  104401  040306            4$:     TYPE    ,SDADRS         ;TYPE:  ' STARTING DEVICE ADDRESS: '
6025 034714  013746  002416                    MOV     REGADR,-(SP)    ;MOVE THE PRESENT ADDRESS TO THE STACK FOR PRINTING
6026 034720  104402                            TYPOC                   ;PRINT THE ADDRESS
6027 034722  104401  037400                    TYPE    ,SPACEC         ;TYPE:  '  : '
6028 034726  012703  000006                    MOV     #6,R3           ;EXPECT MAXIMUM 6 CHARACTERS
6029 034732  112737  000067  002704            MOVB    #'7,LRGSTC      ;MOVE ASCII '7' TO THE LARGEST CHARACTER LOCATION
6030 034740  004737  002722                    JSR     PC,READ         ;GO READ 6 CHARACTERS
6031 034744  022703  000006                    CMP     #6,R3           ;SEE IF NON-NUMERIC CHARACTER INPUTED
6032 034750  001022                            BNE     8$              ;BRANCH IF NOT
6033 034752  022737  000033  002664            CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
6034 034760  001704                            BEQ     EDIT            ;BRANCH AROUND ESC PRINT AND PREVIOUS PROMPT BRANC'' IF SO
6035 034762  022737  000003  002664            CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
6036 034770  001005                            BNE     7$              ;BRANCH AROUND PROMPT JUMP IF NOT
6037 034772  000137  034244          5$:       JMP     PROMPT          ;JUMP TO PROMPT A NEW COMMAND
6038 034776  104401  036532          6$:       TYPE    ,ADRERR         ;TYPE:  'ADDRESS INPUTED IS NOT'          ;DPM001
6039                                                                    ;       'IN THE RANGE 160010 TO 163770' ;DPM001
6040 035002  000742                            BR      4$              ;BRANCH BACK FOR REINPUT
6041 035004  022737  000015  002664  7$:       CMP     #CARETN,ANSWER  ;SEE IF <CR> WAS ONLY CHARACTER INPUTED
6042 035012  001417                            BEQ     10$             ;IF <CR> USE EXISTING REG ADDRESS
6043 035014  000735                            BR      4$              ;BRANCH BACK - INPUT NOT LEGAL
6044 035016  020427  160010          8$:       CMP     R4,#160010      ;IS ANSWER BELOW 160010                   ;DPM001
6045 035022  103765                            BLO     6$              ;BRANCH TO PRINT ERROR MESSAGE IF IT IS
6046 035024  020427  163770                    CMP     R4,#163770      ;IS ANSWER ABOVE 163770                   ;DPM001
6047 035030  101362                            BHI     6$              ;BRANCH TO PRINT ERROR MESSAGE IF NOT
6048 035032  032704  000007                    BIT     #7,R4           ;TEST TO MAKE SURE A '0' IS PRESENT IN LOWEST OCTAL DIGIT
6049 035036  001403                            BEQ     9$              ;BRANCH AROUND ERROR MESSAGE TYPE IF SO
6050 035040  104401  036700                    TYPE    ,ADLCHR         ;TYPE:  'ADDRESS INPUTED HAS OTHER THAN 0 FOR LEAST'
6051                                                                    ;       'SIGNIFICANT OCTAL DIGIT'
6052 035044  000721                            BR      4$              ;BRANCH BACK FOR REINPUT
6053 035046  010437  002416          9$:       MOV     R4,REGADR       ;INSTALL NEW # IN TABLE
6054 035052  104401  040252          10$:      TYPE    ,SVADRS         ;TYPE:  'STARTING VECTOR ADDRESS:  '
6055 035056  013746  002456                    MOV     VECADR,-(SP)    ;MOVE PRESENT VECTOR TO STACK FOR PRINTING
6056 035062  104403                            TYPOS                   ;PRINT THE PRESENT VECTOR ADDRESS
6057 035064     003     000                    .BYTE   3,0             ;TYPE 3 CHARACTERS, SUPPRESS LEADING ZEROS
```

```
 6058 035066  104401  037400                TYPE    ,SPACEC         ;TYPE:  '  : '
 6059 035072  012703  000003                MOV     #3,R3           ;EXPECT ONLY 3 CHARACTERS
 6060 035076  004737  002722                JSR     PC,READ         ;GO READ 3 CHARACTERS
 6061 035102  022703  000003                CMP     #3,R3           ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
 6062 035106  001015                        BNE     11$             ;BRANCH IF NOT
 6063 035110  022737  000033  002664        CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
 6064 035116  001674                        BEQ     4$              ;BRANCH TO PREVIOUS PROMPT IF SO
 6065 035120  022737  000003  002664        CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
 6066 035126  001721                        BEQ     5$              ;BRANCH TO PROMPT JUMP IF SO
 6067 035130  022737  000015  002664        CMP     #CARETN,ANSWER  ;SEE IF <CR> WAS INPUTED
 6068 035136  001417                        BEQ     15$             ;BRANCH IF NO CHANGE WANTED
 6069 035140  000744                        BR      10$             ;BRANCH BACK - INPUT WAS ILLEGAL
 6070 035142  022704  000277          11$:  CMP     #277,R4         ;SEE IF ANSWER IS BELOW 300             ;DPM001
 6071 035146  100403                        BMI     13$             ;BRANCH AROUND ERROR MESSAGE IF NOT
 6072 035150  104401  036617                TYPE    ,VECERR         ;TYPE:  'VECTOR INPUTED IS NOT IN THE RANGE OF 300 TO 777';D
 6073 035154  000736                        BR      10$             ;BRANCH BACK FOR REINPUT
 6074 035156  032704  000003          13$:  BIT     #3,R4           ;MAKE SURE LEAST SIGNIFICANT OCTAL DIGIT IS '0' OR '4'
 6075 035162  001403                        BEQ     14$             ;BRANCH OVER ERROR PRINTING IF NOT
 6076 035164  104401  037003                TYPE    ,VCLCHR         ;TYPE:  'VECTOR INPUTED SHOULD HAVE 0 OR 4 AS LEAST DIGIT'
 6077 035170  000730                        BR      10$             ;BRANCH BACK FOR REINPUT
 6078 035172  010437  002456          14$:  MOV     R4,VECADR       ;INSTALL NEW VECTOR ADDRESS IN TABLE
 6079 035176  104401  040200          15$:  TYPE    ,DR1W0B         ;TYPE:  'DR11-W CR B (W=0, B=1) CURRENT STATE = '
 6080 035202  013737  001474  002720        MOV     $DDW0,DDW       ;MOVE DEVICE DESCRIPTOR WORD TO DDW
 6081 035210  012737  000001  002710        MOV     #BIT0,BITTST    ;MOVE BIT STATE TO PRINT TO BITTST
 6082 035216  004737  006536                JSR     PC,PSTATE       ;PRINT CURRENT W/B STATE
 6083 035222  104401  037400                TYPE    ,SPACEC         ;TYPE:  '  : '
 6084 035226  012703  000001                MOV     #1,R3           ;ONLY INPUT 1 CHARACTER
 6085 035232  004737  002722                JSR     PC,READ         ;GO READ 1 CHARACTER
 6086 035236  005703                        TST     R3              ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
 6087 035240  001415                        BEQ     16$             ;BRANCH AROUND NON-NUMERIC TESTS IF SO
 6088 035242  022737  000033  002664        CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
 6089 035250  001700                        BEQ     10$             ;BRANCH TO PREVIOUS PROMPT IF SO
 6090 035252  022737  000015  002664        CMP     #CARETN,ANSWER  ;SEE IF USER WANTS NO CHANGE
 6091 035260  001417                        BEQ     18$             ;BRANCH IF SO
 6092 035262  022737  000003  002664        CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
 6093 035270  001640                        BEQ     5$              ;BRANCH TO PROMPT JUMP IF SO
 6094 035272  000741                        BR      15$             ;BRANCH BACK - INPUT NOT LEGAL
 6095 035274  005704                16$:    TST     R4              ;CHECK FOR LEGAL INPUT
 6096 035276  001403                        BEQ     17$             ;BRANCH IF OK
 6097 035300  022704  000001                CMP     #1,R4           ;CHECK FOR ILLEGAL INPUT
 6098 035304  001334                        BNE     15$             ;BRANCH BACK IF ILLEGAL STATE INPUTED
 6099 035306  042737  000001  001474  17$:  BIC     #BIT0,$DDW0     ;CLEAR THE BIT TO BE ALTERED
 6100 035314  050437  001474                BIS     R4,$DDW0        ;PUT USER INPUT INTO $DDW0
 6101 035320  104401  040137          18$:  TYPE    ,DEVPRI         ;TYPE:  'DEVICE PRIORITY PRESENT LEVEL = '
 6102 035324  013737  001474  002720        MOV     $DDW0,DDW       ;MOVE DEVICE DESCRIPTOR WORD TO DDW
 6103 035332  004737  006560                JSR     PC,PNTPRI       ;PRINT DEVICE PRIORITY
 6104 035336  104401  037400                TYPE    ,SPACEC         ;TYPE:  '  : '
 6105 035342  012703  000001                MOV     #1,R3           ;ONLY INPUT 1 CHARACTER
 6106 035346  004737  002722                JSR     PC,READ         ;GO READ 1 CHARACTER
 6107 035352  005703                        TST     R3              ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
 6108 035354  001415                        BEQ     19$             ;BRANCH AROUND NON-NUMERIC TESTS IF NOT
 6109 035356  022737  000033  002664        CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
 6110 035364  001704                        BEQ     15$             ;BRANCH TO PREVIOUS PROMPT IF SO
 6111 035366  022737  000003  002664        CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
 6112 035374  001544                        BEQ     26$             ;BRANCH IF EXIT WANTED
 6113 035376  022737  000015  002664        CMP     #CARETN,ANSWER  ;SEE IF <CR> INPUTED FOR NO CHANGE WANTED
 6114 035404  001413                        BEQ     20$             ;BRANCH IF NO CHANGE WANTED
```

```
6115 035406  000744                      BR      18$             ;BRANCH BACK - INPUT NOT LEGAL
6116 035410  006304              19$:    ASL     R4              ;PUT PRIORITY IN PROPER POSITION
6117 035412  006304                      ASL     R4              ;BY SHIFTING TO THE LEFT 5 PLACES
6118 035414  006304                      ASL     R4
6119 035416  006304                      ASL     R4
6120 035420  006304                      ASL     R4
6121 035422  042737  000340  001474      BIC     #LEVEL7,$DDW0   ;CLEAR OLD PRIORITY
6122 035430  050437  001474              BIS     R4,$DDW0        ;SET PRIORITY INTO DEVICE DESCRIPTOR WORD
6123 035434  104401  040051      20$:    TYPE    ,TORNCB         ;TYPE:  '2 OR N CYCLE BURST (2 CY=0, N CY=1) PRESENT STATE =
6124 035440  013737  001474  002720      MOV     $DDW0,DDW       ;MOVE DEVICE DESCRIPTOR WORD TO DDW
6125 035446  012737  000002  002710      MOV     #BIT1,BITTST    ;MOVE BIT STATE TO PRINT TO BITTST
6126 035454  004737  006536              JSR     PC,PSTATE       ;PRINT 2/N CYCLE STATE
6127 035460  104401  037400              TYPE    ,SPACEC         ;TYPE:  '  : '
6128 035464  012703  000001              MOV     #1,R3           ;ONLY ONE CHARACTER TO INPUT
6129 035470  004737  002722              JSR     PC,READ         ;READ 1 CHARACTER
6130 035474  005703                      TST     R3              ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
6131 035476  001415                      BEQ     21$             ;BRANCH AROUND NON-NUMERIC TESTS IF NOT
6132 035500  022737  000033  002664      CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
6133 035506  001704                      BEQ     18$             ;BRANCH TO PREVIOUS PROMPT IF SO
6134 035510  022737  000003  002664      CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
6135 035516  001473                      BEQ     26$             ;BRANCH IF USER WANTS TO EXIT
6136 035520  022737  000015  002664      CMP     #CARET,ANSWER   ;SEE IF USER WANTS NO CHANGE
6137 035526  001414                      BEQ     23$             ;BRANCH IF USER WANTS NO CHANGE
6138 035530  000741                      BR      20$             ;BRANCH BACK - USER INPUT NOT LEGAL
6139 035532  005704              21$:    TST     R4              ;CHECK FOR LEGAL INPUT
6140 035534  001403                      BEQ     22$             ;BRANCH IF OK
6141 035536  022704  000001              CMP     #1,R4           ;CHECK FOR ILLEGAL INPUT
6142 035542  001334                      BNE     20$             ;BRANCH BACK IF ILLEGAL STATE INPUTED
6143 035544  006304              22$:    ASL     R4              ;SHIFT BIT OVER 1 PLACE
6144 035546  042737  000002  001474      BIC     #BIT1,$DDW0     ;CLEAR OLD STATE
6145 035554  050437  001474              BIS     R4,$DDW0        ;SET THE USERS INPUTED STATE TO $DDW0
6146 035560  104401  040373      23$:    TYPE    ,DOCTS          ;TYPE:  'DO CABLE TESTS (NO=0, YES=1) PRESENT STATE = '
6147 035564  013737  001474  002720      MOV     $DDW0,DDW       ;MOVE DEVICE DESCRIPTOR WORD TO DDW
6148 035572  012737  000004  002710      MOV     #BIT2,BITTST    ;MOVE BIT STATE TO PRINT TO BITTST
6149 035600  004737  006536              JSR     PC,PSTATE       ;PRINT CABLE STATE
6150 035604  104401  037400              TYPE    ,SPACEC         ;TYPE:  '  : '
6151 035610  012703  000001              MOV     #1,R3           ;INPUT ONLY 1 CHARACTER
6152 035614  004737  002722              JSR     PC,READ         ;GO INPUT 1 CHARACTER
6153 035620  005703                      TST     R3              ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
6154 035622  001415                      BEQ     24$             ;BRANCH AROUND NON-NUMERIC TESTS IF NOT
6155 035624  022737  000033  002664      CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
6156 035632  001700                      BEQ     20$             ;BRANCH TO PREVIOUS PROMPT IF SO
6157 035634  022737  000003  002664      CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
6158 035642  001421                      BEQ     26$             ;BRANCH IF USER WANTS TO EXIT
6159 035644  022737  000015  002664      CMP     #CARETN,ANSWER  ;SEE IF USER WANTS NO CHANGE
6160 035652  001415                      BEQ     26$             ;BRANCH IF USER WANTS NO CHANGE
6161 035654  000741                      BR      23$             ;BRANCH BACK - USER INPUT NOT LEGAL
6162 035656  005704              24$:    TST     R4              ;CHECK FOR LEGAL INPUT
6163 035660  001403                      BEQ     25$             ;BRANCH IF OK
6164 035662  022704  000001              CMP     #1,R4           ;CHECK FOR ILLEGAL INPUT
6165 035666  001334                      BNE     23$             ;BRANCH BACK IF ILLEGAL STATE INPUTED
6166 035670  006304              25$:    ASL     R4              ;SHIFT INPUTED BIT OVER 2 PLACES
6167 035672  006304                      ASL     R4
6168 035674  042737  000004  001474      BIC     #BIT2,$DDW0     ;CLEAR BIT TO BE CHANGED
6169 035702  050437  001474              BIS     R4,$DDW0        ;SET THE USERS INPUTED STATE TO $DDW0
6170 035706  000137  034244      26$:    JMP     PROMPT          ;JUMP TO GET NEW DEVICE NUMBER
```

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

035712  012737  177777  002670  BDLCR:  MOV     #-1,MANSIZE     ;MOVE -1 TO MANSIZE
035720  004737  003170                  JSR     PC,FIXTBL       ;GO FILL TABLE
035724  104401  037212                  TYPE    ,BDLCRM         ;TYPE:  'BURST DATA LATE CALIBRATION'
                                                                ;TYPE:  'ATTACH SCOPE PROBE...'
                                                                ;       'TO CALIBRATE NEXT BOARD, TYPE ANY CHARACTER'
035730  012737  000001  002544          MOV     #BIT0,DEVMSK    ;SET UP BIT MASK TO TEST $DEVM FOR DEVICES
035736  012700  002456                  MOV     #VECADR,R0      ;MOVE VECADR TO R0
035742  012701  002416                  MOV     #REGADR,R1      ;MOVE REGADR TO R1
035746  005037  001422                  CLR     $UNIT           ;CLEAR $UNIT
035752  033737  002544  001466  2$:     BIT     DEVMSK,$DEVM    ;CHECK TO SEE IF DEVICE IS TO BE TESTED
035760  001015                          BNE     5$              ;BRANCH IF SO
035762  005737  002544                  TST     DEVMSK          ;SEE IF BIT 15 IS SET
035766  100004                          BPL     4$              ;BRANCH TO CONTINUE IF NOT SET
035770  104401  036477                  TYPE    ,BCDONE         ;TYPE:  'BURST CALIBRATION COMPLETE'
035774  000137  034244                  JMP     PROMPT          ;JUMP TO PROMPT A NEW COMMAND
036000  022021                  4$:     CMP     (R0)+,(R1)+     ;INCREMENT THE TWO POINTERS
036002  006337  002544                  ASL     DEVMSK          ;UPDATE DEVICE MAP TEST MASK
036006  005237  001422                  INC     $UNIT           ;INCREMENT UNIT NUMBER
036012  000757                          BR      2$              ;GO TEST NEXT BIT OF DEVICE MASK
036014  011137  002522          5$:     MOV     (R1),CSR        ;PUT UUT CSR ADDRESS INTO DEVICE CSR LOCATION
036020  062737  000004  002522          ADD     #4,CSR          ;POINT CSR TO CSR ADDRESS
036026  011037  002526                  MOV     (R0),DRINV      ;PUT UUT VECTOR ADDRESS INTO DEVICE DRINV
036032  104401  037200                  TYPE    ,DEVICE         ;TYPE:  'DEVICE # '
036036  013746  001422                  MOV     $UNIT,-(SP)     ;PUT UNIT NUMBER ON STACK FOR TYPEOUT
036042  104405                          TYPDS                   ;GO TYPE THE UNIT NUMBER IN DECIMAL
036044  104401  037511                  TYPE    ,UCAL           ;TYPE:  ' UNDER CALIBRATION'
036050  004737  003646                  JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
036054  005077  144442          6$:     CLR     @CSR            ;CLR CYCLE BIT
036060  012737  000077  002660          MOV     #77,TIME        ;MOVE WAIT LOOP COUNTER TO TIME
036066  052777  000400  144426          BIS     #CY,@CSR        ;SET CYCLE BIT
036074  005337  002660          7$:     DEC     TIME            ;SUBTRACT 1 FROM TIME UNTIL ZERO
036100  001375                          BNE     7$              ;BRANCH BACK IF NOT ZERO YET
036102  105777  143236                  TSTB    @$TKS           ;IS A CHARACTER WAITING INDICATING USER WANTS TO GO ON?
036106  100362                          BPL     6$              ;BRANCH IF NOT
036110  017737  144522  002660          MOV     @TKB,TIME       ;WASTE THE CHARACTER, CLEARING THE CHARACTER FLAG
036116  000730                          BR      4$              ;GO ON TO NEXT BOARD
```

```
6212                                        .SBTTL  ASCII AND ASCIZ MESSAGES AND LOCATIONS
6213 036120  200  123  124  STKIFL: .ASCIZ  <CRLF>/STACK IS FULL - DATA MAY HAVE BEEN LOST/<CRLF><CRLF>
6214 036173  136  131  200  ERCHDR: .ASCII  /^Y/<CRLF>/SUMMATION OF ERRORS SINCE START OR LAST REPORT/
6215 036254  200  200  102          .ASCIZ  <CRLF><CRLF>/BOARD #   PASS #   ERRTTL/<CRLF>
6216 036311  136  131  200  NODATA: .ASCIZ  /^Y/<CRLF>/NO ERROR TOTALS TO REPORT/<CRLF><CRLF>
6217 036350  040  055  040  ETDEV:  .ASCIZ  / - TOTAL ERRORS THIS DEVICE = /
6218 036407  111  114  114  BDNERR: .ASCIZ  /ILLEGAL NUMBER (# OTHER THAN 1-16) OR CHARACTER INPUTED/
6219 036477  102  125  122  BCDONE: .ASCIZ  /BURST CALIBRATION COMPLETE/
6220 036532  101  104  104  ADRERR: .ASCIZ  /ADDRESS INPUTED IS NOT IN THE RANGE 160010 TO 163770/
6221 036617  126  105  103  VECERR: .ASCIZ  /VECTOR INPUTED IS NOT IN THE RANGE OF 300 TO 777/
6222 036700  101  104  104  ADLCHR: .ASCIZ  /ADDRESS INPUTED HAS OTHER THAN 0 FOR LEAST SIGNIFICANT OCTAL DIGIT/
6223 037003  126  105  103  VCLCHR: .ASCIZ  /VECTOR INPUTED SHOULD HAVE 0 OR 4 AS LEAST DIGIT/
6224 037064  074  105  123  ESCAPE: .ASCIZ  /<ESC>/
6225 037072  200  104  105  MUSTED: .ASCII  <CRLF>/DEVICE ADDRESS AND-OR VECTOR TABLE NOT SET UP - /
6226 037153  115  125  123          .ASCIZ  /MUST EDIT FIRST/
6227 037173  040  000       LETNCR: .ASCIZ  / /
6228 037175  136  103  000  CNTRLC: .ASCIZ  /^C/
6229 037200  104  105  126  DEVICE: .ASCIZ  /DEVICE # /
6230 037212  200  102  125  BDLCRM: .ASCII  <CRLF>/BURST DATA LATE CALIBRATION/
6231 037246  200  101  124          .ASCII  <CRLF>/ATTACH SCOPE PROBE.../
6232 037274  200  124  117          .ASCIZ  <CRLF>/TO CALIBRATE NEXT BOARD, TYPE ANY CHARACTER/<CRLF>
6233 037352  040  040  000  SPACES: .ASCIZ  /  /
6234 037355  040  040  040  SPACE3: .ASCIZ  /   /
6235 037361  040  040  040  SPACE6: .ASCIZ  /      /
6236 037370  040  040  040  SPACE7: .ASCIZ  /       /
6237 037400  040  040  072  SPACEC: .ASCIZ  /  : /
6238 037405  102  000       B:      .ASCIZ  /B/
6239 037407  127  000       W:      .ASCIZ  /W/
6240 037411  116  117  000  NO:     .ASCIZ  /NO/
6241 037414  131  105  123  YES:    .ASCIZ  /YES/
6242 037420  062  000       TWO:    .ASCIZ  /2/
6243 037422  116  000       N:      .ASCIZ  /N/
6244 037424  102  117  101  BOARD:  .ASCIZ  /BOARD # /
6245 037435  200  200  000  CRLF2:  .ASCIZ  <CRLF><CRLF>
6246 037440  200  101  114  BCFIN:  .ASCIZ  <CRLF>/ALL BOARDS CALIBRATED - BEGINNING TEST/<CRLF>
6247 037511  040  125  116  UCAL:   .ASCIZ  / UNDER CALIBRATION/<CRLF>
6248 037535  200  116  125  NOBUT:  .ASCIZ  <CRLF>/NUMBER OF BOARDS UNDER TEST: /
6249 037574  200  200  104  BRVWPC: .ASCII  <CRLF><CRLF>/DIAGNOSTIC HAS DETERMINED THE FOLLOWING ABOUT THE/<CRLF>
6250 037660  104  122  061          .ASCII  /DR11-W(S) IT HAS FOUND.  USER *MUST* DETERMINE ACCURACY/<CRLF><CRLF>
6251 037751  040  040  040          .ASCIZ  /   BOARD#   REGADR    VECADR     W-B    P-LEV   2-N CY   CABLE/<CRLF>
6252 040051  200  062  040  TORNCB: .ASCIZ  <CRLF>/2 OR N CYCLE BURST (2 CY=0, N CY=1) PRESENT STATE = /
6253 040137  200  104  105  DEVPRI: .ASCIZ  <CRLF>/DEVICE PRIORITY PRESENT LEVEL = /
6254 040201  200  104  122  DR1WOB: .ASCIZ  <CRLF>/DR11-W OR B (W=0, B=1) CURRENT STATE = /
6255 040252  200  123  124  SVADRS: .ASCIZ  <CRLF>/STARTING VECTOR ADDRESS:  /
6256 040306  200  123  124  SDADRS: .ASCIZ  <CRLF>/STARTING DEVICE ADDRESS: /
6257 040341  040  124  105  TSTCOM: .ASCIZ  / TESTING COMPLETE, PASS #/
6258 040373  200  104  117  DOCTS:  .ASCIZ  <CRLF>/DO CABLE TESTS (NO=0, YES=1) PRESENT STATE = /
6259 040452  200  200  043  HEADER: .ASCII  <CRLF><CRLF>/# OF     START                                    2-N    CABLE/
6260 040546  200  102  117          .ASCIZ  <CRLF>/BOARDS   REGADR   VECADR      W-B    P-LEV   CYCLE   TESTS/<CRLF>
6261 040643  200  200  115  MBDIAL: .ASCIZ  <CRLF><CRLF>/MULTIPLE BOARD DIALOGUE/<CRLF>
6262 040676  200  105  116  ECELR:  .ASCIZ  <CRLF>/ENTER COMMAND ([E]DIT, [L]IST, [B]URST CALIBRATION, [R]UN):  /
6263 040775  124  110  105  NOMORE: .ASCII  /THERE ARE STILL MORE ERRORS, BUT WILL NOT BE PRINTED./<CRLF>
6264 041063  105  122  122          .ASCIZ  /ERRORS WILL STILL BE COUNTED AND PRINTED AT THE EOP./<CRLF>
6265 041151  200  116  117  NODVPR: .ASCII  <CRLF>/NO DEVICES RECOGNIZED - DIAGNOSTIC CANNOT BE RUN/<CRLF>
6266 041233  122  105  123          .ASCIZ  /RESTART AT 204 IF A DEVICE IS PRESENT/<CRLF>
6267 041302  200  050  136  M1:     .ASCII  <CRLF>/(^X) INHIBITS EOP'S, (^Y) FOR ERROR SUMMARY/<CRLF>
6268 041357  125  116  111          .ASCIZ  /UNIBUS HANG?  RESTART AT ADDRESS /
```

ASCII AND ASCIZ MESSAGES AND LOCATIONS

```
 6269 041421  200  200  103  M1A:     .ASCII  <CRLF><CRLF>/CZDRLDO DR11 GEN NPR INTFC LOGIC TEST/<CRLF>
 6270 041471  105  117  120           .ASCIZ  /EOP MSG WILL PRINT EVERY 20 SECONDS APPROXIMATELY/<CRLF>
 6271 041554  104  105  126  BARADR:  .ASCIZ  /DEVICE ADDRESS - /
 6272 041576  054  040  124  TSNUMB:  .ASCIZ  /, TEST NUMBER - /
 6273 041617  054  040  120  PASNUM:  .ASCIZ  /, PASS NUMBER - /
 6274                                  .EVEN
 6275 041640  000C00                .SAV:    .WORD   0
 6276 041642                                  .BLKW   400
 6277 042642  000000                BUFF:    .WORD   0
 6278 042644  042644                XINBUF:  .
 6279 042646                                  .BLKW   400
 6280 043646  043646                XCHKBU:  .
 6281 043650                                  .BLKW   400
 6282 044650  044652                CAPNTR:  .WORD   CAPSTK          ;LOCATION TO HOLD POINTER FOR CAPTURE STACK
 6283 044652                        CAPSTK:  .BLKW   600.            ;LOCATIONS TO STORE UP TO 150 DECIMAL PASSES AND THEIR ERROR
 6284 047132  000000                ENDSTK:  .WORD   0               ;FLAG SIGNALING END OF THE STACK
```

ERROR MESSAGES

```
6285                                            .SBTTL  ERROR MESSAGES
6286 047134      124     105     123  EM1:      .ASCIZ  /TEST SEQUENCING ERROR/
6287 047162      103     101     116  EM2:      .ASCIZ  /CANNOT ACCESS DR11 REGISTER/
6288 047216      104     122     061  EM3:      .ASCIZ  /DR11-B OR W MODE INCORRECT (0=B, 1=W)/
6289 047264      111     116     111  EM4:      .ASCIZ  /INIT FAILED TO CLEAR WCR/
6290 047315      111     116     111  EM5:      .ASCIZ  /INIT FAILED TO CLEAR BAR/
6291 047346      111     116     111  EM6:      .ASCIZ  /INIT FAILED TO CLEAR BDR/
6292 047377      111     116     111  EM7:      .ASCIZ  /INIT FAILED TO CLEAR ALL CSR R-W BITS/
6293 047445      122     105     123  EM10:     .ASCIZ  /RESET FAILED TO CLEAR WCR/
6294 047477      101     124     124  EM11:     .ASCIZ  /ATTEMPT TO SET ALL WCR BITS FAILED/
6295 047542      122     105     123  EM12:     .ASCIZ  /RESET FAILED TO CLEAR BAR/
6296 047574      101     124     124  EM13:     .ASCIZ  /ATTEMPT TO SET ALL BAR BITS TO 1 FAILED/
6297 047644      103     123     122  EM14:     .ASCIZ  /CSR BIT TEST FAILED (FATAL - DIAGNOSTIC NOT CONTINUED)/
6298 047733      103     123     122  EM15:     .ASCIZ  /CSR BIT TEST FAILED/
6299 047757      105     111     122  EM16:     .ASCIZ  /EIR BIT TEST FAILED/
6300 050003      122     105     101  EM17:     .ASCIZ  /READY AND MAINTENANCE ARE NOT THE ONLY BITS SET IN CSR/
6301 050072      101     124     124  EM20:     .ASCIZ  /ATTN AND ERROR FAILED TO SET PROPERLY/
6302 050140      101     124     124  EM21:     .ASCIZ  /ATTN AND ERROR FAILED TO CLEAR PROPERLY/
6303 050210      105     122     122  EM22:     .ASCIZ  /ERROR BIT SHOULD HAVE BEEN CLEAR/
6304 050251      102     111     124  EM23:     .ASCIZ  /BIT PATTERN NOT LOADED PROPERLY IN WCR/
6305 050320      122     105     101  EM24:     .ASCIZ  /READY OF CSR WAS NOT SET/
6306 050351      102     111     124  EM25:     .ASCIZ  /BIT 0 OF THE BAR WAS SET/
6307 050402      102     111     124  EM26:     .ASCIZ  /BIT PATTERN TEST FAILED IN BAR/
6308 050441      127     103     122  EM27:     .ASCIZ  /WCR DATA PATTERN NOT CORRECT/
6309 050476      106     125     116  EM30:     .ASCIZ  /FUNCTION BIT(S) ARE NOT CLEAR/
6310 050534      104     123     124  EM31:     .ASCIZ  /DSTAT A, B OR C ARE NOT AS EXPECTED/
6311 050600      102     104     122  EM32:     .ASCIZ  /BDR IS NOT CLEAR/
6312 050621      101     114     114  EM33:     .ASCIZ  /ALL BDR BITS ARE NOT SET/
6313 050652      102     104     122  EM34:     .ASCIZ  /BDR FAILS TO HOLD A BIT PATTERN/
6314 050712      102     104     122  EM35:     .ASCIZ  /BDR SHOULD NOT HAVE BEEN LOADED WITH NEW PATTERN/
6315 050773      102     104     122  EM36:     .ASCIZ  /BDR PATTERN NOT CORRECT/
6316 051023      122     105     101  EM37:     .ASCIZ  /READY IS NOT THE ONLY BIT SET/
6317 051061      122     105     101  EM40:     .ASCIZ  /READY SHOULD NOT BE SET/
6318 051111      122     105     101  EM41:     .ASCIZ  /READY WAS CLEARED BUT NEVER SET AGAIN/
6319 051157      122     105     101  EM42:     .ASCIZ  /READY CANNOT BE SET BY INIT/
6320 051213      104     122     061  EM43:     .ASCIZ  /DR11 FAILED TO INTERRUPT/
6321 051244      104     122     061  EM44:     .ASCIZ  /DR11 INTERRUPTED, BUT IT SHOULDN'T HAVE/
6322 051314      105     122     122  EM45:     .ASCIZ  /ERROR BIT SHOULD NOT BE CLEAR/
6323 051352      106     125     116  EM46:     .ASCIZ  /FUNCTION BITS DIDN'T INCREMENT IN MAINT MODE/
6324 051427      103     123     122  EM47:     .ASCIZ  /CSR IS WRONG/
6325 051444      124     122     101  EM50:     .ASCIZ  /TRANSFERS SHOULD HAVE BEEN INHIBITED/
6326 051511      104     122     061  EM51:     .ASCIZ  /DR11 SHOULD NOT HAVE INTERRUPTED A SECOND TIME/
6327 051570      105     130     120  EM52:     .ASCIZ  /EXPECTED INTERRUPT DID NOT OCCUR/
6328 051631      127     103     122  EM53:     .ASCIZ  /WCR NOT EQUAL TO ZERO/
6329 051657      102     101     122  EM54:     .ASCIZ  /BAR IS WRONG/
6330 051674      102     101     104  EM55:     .ASCIZ  /BAD DATA IN BDR/
6331 051714      104     101     124  EM56:     .ASCIZ  /DATA NOT TRANSFERED CORRECTLY/
6332 051752      102     125     106  EM57:     .ASCIZ  /BUFFER DATA NOT CORRECT/
6333 052002      124     117     117  EM60:     .ASCIZ  /TOO MANY WORDS WERE TRANSFERED/
6334 052041      125     116     105  EM61:     .ASCIZ  /UNEXPECTED TRAP OR INTERRUPT TO TRAP ADDRESS BELOW/
6335 052124      103     123     122  EM62:     .ASCIZ  /CSR AND-OR WCR AND-OR BAR ARE INCORECT/
6336 052173      104     122     061  EM63:     .ASCIZ  /DR11 INTERRUPTED AT WRONG LEVEL/
6337 052233      062     055     116  EM65:     .ASCIZ  /2-N CYCLE BURST SWITCH IN WRONG POSITION/
6338 052304      115     125     114  EM66:     .ASCIZ  /MULTICYCLE BIT IN THE EIR IS WRONG/
6339 052347      103     123     122  EM202:    .ASCIZ  /CSR PATTERN NOT CORRECT/
6340 052377      102     104     122  EM211:    .ASCIZ  /BDR AND-OR WCR AND-OR BAR ARE INCORECT/
```

DATA HEADERS

```
6341                                                .SBTTL  DATA HEADERS
6342 052446  105  130  120  DH1:    .ASCII  /EXPCTD  RECVD/<CRLF>
6343 052464  124  105  123          .ASCIZ  /TEST #  TEST #/
6344 052503  124  105  123  DH2:    .ASCIZ  /TEST #  ERR PC  ABRTPC  REGISTER/
6345 052544  124  105  123  DH3:    .ASCIZ  /TEST #  ERR PC  EXPMOD  ACTMOD  CSRADR/
6346 052613  124  105  123  DH4:    .ASCIZ  /TEST #  ERR PC  WCRADR  WCRCONTENTS/
6347 052657  124  105  123  DH5:    .ASCIZ  /TEST #  ERR PC  BARADR  BAREXP  BARRCV/
6348 052726  124  105  123  DH6:    .ASCIZ  /TEST #  ERR PC  BDRADR  BDRCONTENTS/
6349 052772  124  105  123  DH7:    .ASCIZ  /TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS/
6350 053046  040  040  040  DH14:   .ASCII  /                BIT(S)/<CRLF>
6351 053075  124  105  123          .ASCIZ  /TEST #  ERR PC  TESTED  CSRADR  CSREXP  CSRCONTENTS/
6352 053161  040  040  040  DH16:   .ASCII  /                BIT(S)/<CRLF>
6353 053210  124  105  123          .ASCIZ  /TEST #  ERR PC  TESTED  EIRADR  EIREXP  EIRCONTENTS/
6354 053274  124  105  123  DH17:   .ASCIZ  /TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS/
6355 053350  124  105  123  DH23:   .ASCIZ  /TEST #  ERR PC  WCRADR  WCREXP  WCRCONTENTS/
6356 053424  124  105  123  DH26:   .ASCIZ  /TEST #  ERR PC  BARADR  BAREXP  BARCONTENTS/
6357 053500  124  105  123  DH34:   .ASCIZ  /TEST #  ERR PC  BDRADR  BDREXP  BDRCONTENTS/
6358 053554  124  105  123  DH43:   .ASCIZ  /TEST #  ERR PC  CSRADR  CSRCONTENTS/
6359 053620  124  105  123  DH50:   .ASCIZ  /TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  BARADR  BAREXP  BARRCV/
6360 053717  124  105  123  DH56:   .ASCIZ  /TEST #  ERR PC  NPR1AD  NPR1EX  NPR1RC  CSRADR/
6361 053776  040  040  040  DH57:   .ASCII  /                CHECK   CHECK   INPUT   INPUT/<CRLF>
6362 054054  124  105  123          .ASCIZ  /TEST #  ERR PC  BUFADR  BUFDAT  BUFADR  BUFDAT  CSRADR/
6363 054143  040  040  040  DH60:   .ASCII  /                DIDNOT/<CRLF>
6364 054172  124  105  123          .ASCIZ  /TEST #  ERR PC  EXPECT  ADRESS  CSRADR/
6365 054241  124  105  123  DH61:   .ASCIZ  /TEST #  ERR PC  WCRADR  OLDPC   TRAP ADR/
6366 054312  124  105  123  DH62:   .ASCII  /TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  CSREXP  /
6367 054372  103  123  122          .ASCIZ  /CSRRCV  BAREXP  BARRCV/
6368 054421  124  105  123  DH63:   .ASCIZ  /TEST #  ERR PC  EXPLVL  RCVLVL  CSRADR/
6369 054470  124  105  123  DH64:   .ASCIZ  /TEST #  ERR PC  EXPLVL  CSRADR/
6370 054527  124  105  123  DH65:   .ASCIZ  /TEST #  ERR PC  CSRADR  EIREXP  EIRRCV/
6371 054576  124  105  123  DH66:   .ASCIZ  /TEST #  ERR PC  PATERN  CSRADR  CSRRCV/
6372 054645  124  105  123  DH202:  .ASCIZ  /TEST #  ERR PC  CSRADR  PATLDD  CSREXP  CSRRCV/
6373 054724  124  105  123  DH203:  .ASCIZ  /TEST #  ERR PC  CSRADR  PATERN  CSREXP  CSRCONTENTS/
6374 055010  124  105  123  DH207:  .ASCIZ  /TEST #  ERR PC  PATERN  CSRADR  CSRCONTENTS/
6375 055064  124  105  123  DH210:  .ASCIZ  /TEST #  ERR PC  WCRADR  WCRCONTENTS/
6376 055130  124  105  123  DH211:  .ASCIZ  /TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  BDREXP  BDRRCV  BAREXP  BARRCV/
6377                                        .EVEN
```

DATA TABLES

```
6378                                                .SBTTL  DATA TABLES
6379 055240  001362  001414  000000  DT1:    .WORD   $TMP1,$TESTN,0
6380 055246  001414  001316  002672  DT2:    .WORD   $TESTN,$ERRPC,OLDPC1,DREG,0
6381 055260  001414  001316  001362  DT3:    .WORD   $TESTN,$ERRPC,$TMP1,BORW,CSR,0
6382 055274  001414  001316  002516  DT4:    .WORD   $TESTN,$ERRPC,WCR,RWCR,0
6383 055306  001414  001316  002520  DT5:    .WORD   $TESTN,$ERRPC,BAR,EBAR,RBAR,0
6384 055322  001414  001316  002524  DT6:    .WORD   $TESTN,$ERRPC,BDR,RBDR,0
6385 055334  001414  001316  002522  DT7:    .WORD   $TESTN,$ERRPC,CSR,ECSR,RCSR,0
6386 055350  001414  001316  002536  DT14:   .WORD   $TESTN,$ERRPC,BUT,CSR,ECSR,RCSR,0
6387 055366  001414  001316  002536  DT16:   .WORD   $TESTN,$ERRPC,BUT,CSR,EEIR,REIR,0
6388 055404  001414  001316  002522  DT17:   .WORD   $TESTN,$ERRPC,CSR,ECSR,RCSR,0
6389 055420  001414  001316  002516  DT23:   .WORD   $TESTN,$ERRPC,WCR,EWCR,RWCR,0
6390 055434  001414  001316  002520  DT26:   .WORD   $TESTN,$ERRPC,BAR,EBAR,RBAR,
6391 055450  001414  001316  002524  DT34:   .WORD   $TESTN,$ERRPC,BDR,EBDR,RBDR,0
6392 055464  001414  001316  002522  DT43:   .WORD   $TESTN,$ERRPC,CSR,RCSR,0
6393 055476  001414  001316  002516  DT50:   .WORD   $TESTN,$ERRPC,WCR,EWCR,RWCR,BAR,EBAR,RBAR,0
6394 055520  001414  001316  002606  DT56:   .WORD   $TESTN,$ERRPC,ANPR1,ENPR1,NPR1,CSR,0
6395 055536  001414  001316  001370  DT57:   .WORD   $TESTN,$ERRPC,$TMP4,$TMP2,$TMP5,$TMP3,CSR,0
6396 055556  001414  001316  001364  DT60:   .WORD   $TESTN,$ERRPC,$TMP2,$TMP3,CSR,0
6397 055572  001414  001316  002516  DT61:   .WORD   $TESTN,$ERRPC,WCR,OLDPC2,BDVECT,0
6398 055606  001414  001316  002516  DT62:   .WORD   $TESTN,$ERRPC,WCR,EWCR,RWCR,ECSR,RCSR,EBAR,RBAR,0
6399 055632  001414  001316  002552  DT63:   .WORD   $TESTN,$ERRPC,DRLEV,LEVEL,CSR,0
6400 055646  001414  001316  001362  DT64:   .WORD   $TESTN,$ERRPC,$TMP1,CSR,0
6401 055660  001414  001316  002522  DT65:   .WORD   $TESTN,$ERRPC,CSR,EEIR,REIR,0
6402 055674  001414  001316  002604  DT66:   .WORD   $TESTN,$ERRPC,ENPR1,CSR,RCSR,0
6403 055710  001414  001316  002522  DT202:  .WORD   $TESTN,$ERRPC,CSR,BUT,ECSR,RCSR,0
6404 055726  001414  001316  002522  DT203:  .WORD   $TESTN,$ERRPC,CSR,$TMP0,ECSR,RCSR,0
6405 055744  001414  001316  001362  DT207:  .WORD   $TESTN,$ERRPC,$TMP1,CSR,RCSR,0
6406 055760  001414  001316  002516  DT210:  .WORD   $TESTN,$ERRPC,WCR,RWCR,0
6407 055772  001414  001316  002516  DT211:  .WORD   $TESTN,$ERRPC,WCR,EWCR,RWCR,EBDR,RBDR,EBAR,RBAR,0
```

BUS HANG ROUTINE

```
 6408                                           .SBTTL  BUS HANG ROUTINE
 6409 056016  104401  041554          UBHANG:   TYPE    ,BARADR           ;TYPE:  'DEVICE ADDRESS - '
 6410 056022  013746  002516                    MOV     WCR,-(SP)         ;PUT DEVICE ADDRESS ON STACK
 6411 056026  104402                            TYPOC                     ;GO TYPE IT IN OCTAL
 6412 056030  104401  041576                    TYPE    ,TSNUMB           ;TYPE:  ', TEST NUMBER - '
 6413 056034  013746  001414                    MOV     $TESTN,-(SP)      ;PUT TEST NUMBER ON STACK
 6414 056040  104403                            TYPOS                     ;GO TYPE IT IN OCTAL
 6415 056042     002     000                    .BYTE   2,0               ;TYPE 2 DIGITS, LEADING ZEROS SUPRESSED
 6416 056044  104401  041617                    TYPE    ,PASNUM           ;TYPE:  ', PASS NUMBER - '
 6417 056050  013746  002716                    MOV     $PASS2,-(SP)      ;MOVE OVERFLOW NUMBER TO THE STACK
 6418 056054  013746  001416                    MOV     $PASS,-(SP)       ;PUT PASS NUMBER ON STACK
 6419 056060  104414                            TYPDE                     ;GO TYPE IT IN EXTENDED DECIMAL
 6420 056062  104401  001405                    TYPE    ,$CRLF            ;TYPE A <CRLF>
 6421 056066  000000                            HALT                      ;WHOA - YOU GOTTA SERIOUSA PROBLEMA, BUDDY!
 6422 056070  000137  011420                    JMP     START1            ;JUMP TO RESTART PROGRAM
 6423 056074  177777  177777  177777          .WORD   -1,-1,-1,-1       ;TAKE UP SOME SPACE (APT HACK)
 6424 056104  000000                  NOCARE:   .WORD   0         ;LOCATION FOR USE WHENEVER CYCLE BIT OF CSR IS USED. THIS
 6425                                                             ;SHOULD *ALWAYS* BE THE LAST WORD LOCATION IN THIS DIAGNOSTIC
 6426          000001                           .END
```

SYMBOL TABLE

```
ABASE = 172410          AUNIT = 000000          BUT     002536          CY    = 000400          DSA   = 004000
ACDW1 = 000000          AUSWR = 000000          CAPNTR  044650          DAB   = 006000          DSB   = 002000
ACDW2 = 000000          AVECT1= 000300          CAPSTK  044652          DAC   = 005000          DSC   = 001000
ACPUOP= 000000          AVECT2= 000000          CARETN= 000015          DATCHK  003526          DST   = 007000
ADDR    002646          B       037405          CAT   = 157777          DATCHX  003644          DSWR  = 177570
ADDW0 = 000000          BAR     002520          CATCH   006466          DATCH1  003546          DT1     055240
ADDW1 = 000000          BARADR  041554          CBIT0 = 177776          DATCH2  003624          DT14    055350
ADDW10= 000000          BCDONE  036477          CBIT1 = 177775          DATOCK  003716          DT16    055366
ADDW11= 000000          BCFIN   037440          CBIT10= 175777          DATOCX  004062          DT17    055404
ADDW12= 000000          BDFAIL  002666          CBIT11= 173777          DATOC1  003736          DT2     055246
ADDW13= 000000          BDLCR   035712          CBIT12= 167777          DATOC2  004000          DT202   055710
ADDW14= 000000          BDLCRM  037212          CBIT13= 157777          DBC   = 003000          DT203   055726
ADDW15= 000000          BDNERR  036407          CBIT14= 137777          DDISP = 177570          DT207   055744
ADDW2 = 000000          BDNUMB  006122          CBIT15= 077777          DDW     002720          DT210   055760
ADDW3 = 000000          BDR     002524          CBIT2 = 177773          DEVICE  037200          DT211   055772
ADDW4 = 000000          BDVECT  002542          CBIT3 = 177767          DEVMSK  002544          DT23    055420
ADDW5 = 000000          BEGIN   011674          CBIT4 = 177757          DEVPRI  040137          DT26    055434
ADDW6 = 000000          BEGIN1  012356          CBIT5 = 177737          DH1     052446          DT3     055260
ADDW7 = 000000          BITTST  002710          CBIT6 = 177677          DH14    053046          DT34    055450
ADDW8 = 000000          BIT0  = 000001          CBIT7 = 177577          DH16    053161          DT4     055274
ADDW9 = 000000          BIT00 = 000001          CBIT8 = 177377          DH17    053274          DT43    055464
ADEVCT= 000000          BIT01 = 000002          CBIT9 = 176777          DH2     052503          DT5     055306
ADEVM = 000000          BIT02 = 000004          CB1513= 057777          DH202   054645          DT50    055476
ADLCHR  036700          BIT03 = 000010          CCY   = 177377          DH203   054724          DT56    055520
ADRERR  036532          BIT04 = 000020          CDAB  = 171777          DH207   055010          DT57    055536
AENV  = 000000          BIT05 = 000040          CDAC  = 172777          DH210   055064          DT6     055322
AENVM = 000000          BIT06 = 000100          CDBC  = 174777          DH211   055130          DT60    055556
AFATAL= 000000          BIT07 = 000200          CDSA  = 173777          DH23    053350          DT61    055572
AMADR1= 000000          BIT08 = 000400          CDSB  = 175777          DH26    053424          DT62    055606
AMADR2= 000000          BIT09 = 001000          CDSC  = 176777          DH3     052544          DT63    055632
AMADR3= 000000          BIT1  = 000002          CDST  = 170777          DH34    053500          DT64    055646
AMADR4= 00000C          BIT10 = 002000          CEIR  = 077777          DH4     052613          DT65    055660
AMAMS1= 000000          BIT11 = 004000          CER   = 077777          DH43    053554          DT66    055674
AMAMS2= 000000          BIT12 = 010000          CFNC  = 177761          DH5     052657          DT7     055334
AMAMS3= 000000          BIT13 = 020000          CF1   = 177775          DH50    053620          DYWTLT  006401
AMAMS4= 000000          BIT14 = 040000          CF2   = 177773          DH56    053717          EBAR    002600
AMSGAD= 000000          BIT15 = 100000          CF3   = 177767          DH57    053776          EBDR    002576
AMSGLG= 000000          BIT2  = 000004          CGO   = 177776          DH6     052726          ECELR   040676
AMSGTY= 000000          BIT3  = 000010          CHARCT  032555          DH60    054143          ECSR    002572
AMTYP1= 000000          BIT4  = 000020          CHKBFF  003330          DH61    054241          EDIT    034572
AMTYP2= 000000          BIT5  = 000040          CHKBUF  002620          DH62    054312          EEIR    002574
AMTYP3= 000000          BIT6  = 000100          CHKCAB  003670          DH63    054421          EIR   = 100000
AMTYP4= 000000          BIT7  = 000200          CIE   = 177677          DH64    054470          EMTVEC= 000030
ANPR1   002606          BIT8  = 000400          CKSWR = 104407          DH65    054527          EM1     047134
ANSWER  002664          BIT9  = 001000          CLENUP  003646          DH66    054576          EM10    047445
APASS = 000000          BOARD   037424          CMA   = 167777          DH7     052772          EM11    047477
APRIOR= 000000          BORW    002610          CNTLC = 000003          DIOMEM  002614          EM12    047542
APTCSU= 000040          BPINIT  004154          CNTRLC  037175          DISPLA  001342          EM13    047574
APTENV= 000001          BPT   = 000003          CNX   = 137777          DISPRE  000174          EM14    047644
APTSIZ= 000200          BPTINT  004216          CPSAVE  032654          DOCTS   040373          EM15    047733
APTSPO= 000100          BPTVCT  000014          CR    = 000015          DOWEPR  026012          EM16    047757
ASIZE   005416          BPTVEC= 000014          CRLF  = 000200          DREG    002550          EM17    050003
ASK4PR  005056          BRVWPC  037574          CRLF2   037435          DRGET   004232          EM2     047162
ASK4VC  004742          BRWAIT  002626          CRY   = 177577          DRINV   002526          EM20    050072
ASWREG= 000000          BUFF    042642          CSR     002522          DRLEV   002552          EM202   052347
AT    = 020000          BUFLEN  002622          CX6   = 177757          DRVS    002530          EM21    050140
ATESTN= 000000          BUSERR= 000004          CX7   = 177737          DR1WOB  040201          EM211   052377
```

SYMBOL TABLE

```
EM22    050210      EWCR    002602      M1A     041421      RDOCT = 104412      SW9   = 001000
EM23    050251      EXPAND  027516      N       037422      RDYCHK  002632      TABINX  002546
EM24    050320      FIXTBL  003170      NO      037411      READ    002722      TBITVE= 000014
EM25    050351      FLAG    002652      NOBD    006207      REGADR  002416      TIME    002660
EM26    050402      FNC   = 000016      NOBUT   037535      REINIT  012556      TKB     002636
EM27    050441      FNCCNT  002654      NOCARE  056104      REIR    002562      TKS     002634
EM3     047216      F1    = 000002      NODATA  036311      RESVEC= 000010      TKVEC = 000060
EM30    050476      F2    = 000004      NODVPR  041151      RSTRT   026364      TMOPSW= 000006
EM31    050534      F3    = 000010      NOMORE  040775      RWCR    002570      TORNCB  040051
EM32    050600      GO    = 000001      NOTST   006335      RY    = 000200      TOVECT= 000004
EM33    050621      GOAGIN  026344      NOTVEC  005017      R6    =%000006      TPB     002642
EM34    050652      GTSWR = 104406      NPR1    002612      R7    =%000007      TPS     002640
EM35    050712      HAKTPM  032174      NX    = 040000      SCOPE = 000004      TPVEC = 000064
EM36    050773      HEADER  040452      NXTTST  002554      SDADRS  040306      TRAPVE= 000034
EM37    051023      HT    = 000011      N2    = 000400      SDRINV  002532      TRTVEC= 000014
EM4     047264      IBSAVE  033224      OFL     002702      SDRVS   002534      TSNUMB  041576
EM40    051061      IE    = 000100      OLDPC1  002672      SETTUP  006610      TSTCOM  040341
EM41    051111      INBUF   002616      OLDPC2  002676      SPACEC  037400      TSTDEV  012370
EM42    051157      INBUF1  002656      OLDPS1  002674      SPACES  037352      TSTMM   007066
EM43    051213      INDEV   006124      OLDPS2  002700      SPACE3  037355      TST1    012606
EM44    051244      INOOUT  022606      OUTORG  004762      SPACE6  037361      TST10   015112
EM45    051314      INTA    003356      OUTRAN  006255      SPACE7  037370      TST11   015346
EM46    051352      IOTVEC= 000020      PASCNT  002556      STACK = 001300      TST12   015532
EM47    051427      KDPAR2= 172364      PASNUM  041617      STAD    006142      TST13   015716
EM5     047315      KDPDR2= 172324      PATCHS  011220      STAGIN  000210      TST14   016102
EM50    051444      KIPAR0= 172340      PATRNS  007202      START   011424      TST15   016272
EM51    051511      KIPAR2= 172344      PFECH   033516      START1  011420      TST16   016524
EM52    051570      KIPDR0= 172300      PFECH1  033526      STKIFL  036120      TST17   016762
EM53    051631      KIPDR2= 172304      PFECH2  033562      STKLMT= 177774      TST2    013004
EM54    051657      LENCHK  002624      PFECH3  033612      SVADRS  040252      TST20   017360
EM55    051674      LETNCR  037173      PFECH4  033622      SWR     001340      TST21   017764
EM56    051714      LEVEL   002540      PIRQ  = 177772      SWREG   000176      TST22   020300
EM57    051752      LEVELS  004732      PIRQVE= 000240      SW0   = 000001      TST23   020576
EM6     047346      LEVEL3= 000140      PNTPRI  006560      SW00  = 000001      TST24   021334
EM60    052002      LEVEL4= 000200      PREOP   026010      SW01  = 000002      TST25   021636
EM61    052041      LEVEL5= 000240      PROMPT  034244      SW02  = 000004      TST26   022126
EM62    052124      LEVEL6= 000300      PROUT   005107      SW03  = 000010      TST27   022404
EM63    052173      LEVEL7= 000340      PR0   = 000000      SW04  = 000020      TST3    013134
EM65    052233      LF    = 000012      PR1   = 000040      SW05  = 000040      TST30   022622
EM66    052304      LODBUF  003302      PR2   = 000100      SW06  = 000100      TST31   023110
EM7     047377      LOOP    002662      PR3   = 000140      SW07  = 000200      TST32   023240
ENDEV   025622      LRGSTC  002704      PR4   = 000200      SW08  = 000400      TST33   023620
ENDSTK  047132      MA    = 010000      PR5   = 000240      SW09  = 001000      TST34   024124
ENPR1   002604      MAICLR  007220      PR6   = 000300      SW1   = 000002      TST35   024506
EOPLOC  002706      MAISET  010220      PR7   = 000340      SW10  = 002000      TST36   025004
EOPRSM  032504      MANSIZ  002670      PS    = 177776      SW11  = 004000      TST37   025320
ER    = 100000      MBD     034230      PSTATE  006536      SW12  = 010000      TST4    013466
ERCAPT  003126      MBDIAL  040643      PSW   = 177776      SW13  = 020000      TST40 = 025622
ERCHDR  036173      MEMGMT  002712      PWRVEC= 000024      SW14  = 040000      TST5    014172
ERRCHK  004064      MEPITM  033514      QTYBRD  002414      SW15  = 100000      TST6    014366
ERRCNT  002714      MESSAG  002650      RA    = 002416      SW2   = 000004      TST7    014734
ERROR = 104000      MMPS  = 000252      RBAR    002566      SW3   = 000010      TWO     037420
ERRVEC= 000004      MMR0  = 177572      RBDR    002564      SW4   = 000020      TYPCNF  005142
ER200   002274      MMVECT= 000250      RCSR    002560      SW5   = 000040      TYPDE = 104414
ESC   = 000033      MSG     002644      RDCHR = 104410      SW6   = 000100      TYPDS = 104405
ESCAPE  037064      MUSTED  037072      RDDEC = 104413      SW7   = 000200      TYPE  = 104401
ETDEV   036350      M1      041302      RDLIN = 104411      SW8   = 000400      TYPOC = 104402
```

SYMBOL TABLE

```
TYPON = 104404          $DBLK   027504          $ETEND  001534          $MTYP2  001445          $TMP2   001364
TYPOS = 104403          $DDW0   001474          $FATAL  001412          $MTYP3  001451          $TMP3   001366
UBHANG  056016          $DDW1   001476          $FFLG   034072          $MTYP4  001455          $TMP4   001370
UCAL    037511          $DDW10  001520          $FILLC  001356          $NULL   001354          $TMP5   001372
VA    = 002456          $DDW11  001522          $FILLS  001355          $NUMS   027752          $TMP6   001374
VCLCHR  037003          $DDW12  001524          $GDADR  001320          $NWTST= 000001          $TN   = 000040
VECADR  002456          $DDW13  001526          $GDDAT  001324          $OCNT   027164          $TPB    001352
VECERR  036617          $DDW14  001530          $GET    026310          $OMODE  027166          $TPFLG  001357
W       037407          $DDW15  001532          $GET42  026314          $OVER   032160          $TPS    001350
WCLEN   002630          $DDW2   001500          $GTSWR  030064          $PASS   001416          $TRAP   031112
WCR     002516          $DDW3   001502          $HD   = 000000          $PASS2  002716          $TRAP2  031134
XCHKBU  043646          $DDW4   001504          $HIBTS  001000          $PASTM  001006          $TRP  = 000015
XINBUF  042644          $DDW5   001506          $HIOCT  031110          $POWER  034162          $TRPAD  031146
X6    = 000020          $DDW6   001510          $ICNT   001304          $PWRDN  034074          $TSTM   001004
X7    = 000040          $DDW7   001512          $INTAG  001335          $PWRUP  034112          $TSTNM  001302
YES     037414          $DDW8   001514          $ITEMB  001314          $QUES   001404          $TTYIN  030530
$APTHD  001000          $DDW9   001516          $LF     001406          $RDCHR  030302          $TYPD   027204
$ATYC   033652          $DEVCT  001420          $LFLG   034071          $RDDEC  030574          $TYPDE  027170
$ATY1   033626          $DEVM   001466          $LPADR  001306          $RDLIN  030422          $TYPDS  027200
$ATY3   033634          $DOAGN  026334          $LPERR  001310          $RDOCT  030752          $TYPE   026374
$ATY4   033644          $DTBL   027474          $MADR1  001442          $RDSZ = 000007          $TYPEC  026606
$AUTOB  001334          $ENDAD  026324          $MADR2  001446          $RTNAD  026336          $TYPEX  026740
$BASE   001464          $ENDCT  026102          $MADR3  001452          $SCOPE  031200          $TYPOC  026766
$BDADR  001322          $ENULL  026340          $MADR4  001456          $SETUP= 000137          $TYPON  027002
$BDDAT  001326          $ENV    001430          $MAIL   001410          $STUP = 177777          $TYPOS  026742
$BELL   001400          $ENVM   001431          $MAMS1  001440          $SVLAD  032124          $UNIT   001422
$CDW1   001470          $EOP    026046          $MAMS2  001444          $SVPC = 001000          $UNITM  001010
$CDW2   001472          $EOPCT  026074          $MAMS3  001450          $SWR  = 163400          $USWR   001434
$CHARC  026736          $ERFLG  001303          $MAMS4  001454          $SWREG  001432          $VECT1  001460
$CKSWR  027764          $ERMAX  001315          $MBADR  001002          $SWRMK= 000200          $VECT2  001462
$CMTAG  001300          $ERROR  032656          $MFLG   034070          $SW08T  032556          $XTSTR  031640
$CM3  = 000000          $ERRPC  001316          $MNEW   030562          $TESTN  001414          $$GET4= 000000
$CM4  = 000007          $ERRTB  001534          $MSGAD  001424          $TKB    001346          $$SW08= 000040
$CNTLG  030544          $ERRTY  033226          $MSGLG  001426          $TKS    001344          $OFILL  027165
$CNTLU  030537          $ERTTL  001312          $MSGTY  001410          $TMP0   001360          .SAV    041640
$CPUOP  001436          $ESCAP  001376          $MSWR   030551          $TMP1   001362          .$X   = 001000
$CRLF   001405          $ETABL  001430          $MTYP1  001441

. ABS.  056106      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  57064 WORDS  ( 223 PAGES)
DYNAMIC MEMORY:  20034 WORDS  ( 77 PAGES)
ELAPSED TIME:  00:11:33
CZDRLD.BIN,CZDRLD/CR/-SP/NL:TOC=CZDRLD.MLB/ML,CZDRLD.P11
```

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES							
ABASE	= 172410	#43-3051	52-3178	52-3178					
ACDW1	= 000000	52-3178	52-3178						
ACDW2	= 000000	52-3178	52-3178						
ACPUOP	= 000000	52-3178	52-3178						
ADDR	002646	#61-3569							
ADDW0	= 000000	52-3178	52-3178						
ADDW1	= 000000	52-3178	52-3178						
ADDW10	= 000000	52-3178	52-3178						
ADDW11	= 000000	52-3178	52-3178						
ADDW12	= 000000	52-3178	52-3178						
ADDW13	= 000000	52-3178	52-3178						
ADDW14	= 000000	52-3178	52-3178						
ADDW15	= 000000	52-3178	52-3178						
ADDW2	= 000000	52-3178	52-3178						
ADDW3	= 000000	52-3178	52-3178						
ADDW4	= 000000	52-3178	52-3178						
ADDW5	= 000000	52-3178	52-3178						
ADDW6	= 000000	52-3178	52-3178						
ADDW7	= 000000	52-3178	52-3178						
ADDW8	= 000000	52-3178	52-3178						
ADDW9	= 000000	52-3178	52-3178						
ADEVCT	= 000000	52-3178	52-3178						
ADEVM	= 000000	52-3178	52-3178						
ADLCHR	036700	140-6050	#142-6222						
ADRERR	036532	140-6038	#142-6220						
AENV	= 000000	52-3178	52-3178						
AENVM	= 000000	52-3178	52-3178						
AFATAL	= 000000	52-3178	52-3178						
AMADR1	= 000000	52-3178	52-3178						
AMADR2	= 000000	52-3178	52-3178						
AMADR3	= 000000	52-3178	52-3178						
AMADR4	= 000000	52-3178	52-3178						
AMAMS1	= 000000	52-3178	52-3178						
AMAMS2	= 000000	52-3178	52-3178						
AMAMS3	= 000000	52-3178	52-3178						
AMAMS4	= 000000	52-3178	52-3178						
AMSGAD	= 000000	52-3178	52-3178						
AMSGLG	= 000000	52-3178	52-3178						
AMSGTY	= 000000	52-3178	52-3178						
AMTYP1	= 000000	52-3178	52-3178						
AMTYP2	= 000000	52-3178	52-3178						
AMTYP3	= 000000	52-3178	52-3178						
AMTYP4	= 000000	52-3178	52-3178						
ANPR1	002606	#61-3553	145-6394						
ANSWER	002664	#61-3576	*63-3623	*63-3627	63-3628	63-3632	63-3636	63-3638	63-3640	63-3642
		63-3654	77-4127	139-5945	139-5981	139-5992	139-5994	140-6011	140-6013	140-6015
		140-6033	140-6035	140-6041	140-6063	140-6065	140-6067	140-6088	140-6090	140-6092
		140-6109	140-6111	140-6113	140-6132	140-6134	140-6136	140-6155	140-6157	140-6159
APASS	= 000000	52-3178	52-3178							
APRIOR	= 000000	52-3178								
APTCSU	= 000040	123-5909	#137-5930							
APTENV	= 000001	123-5909	135-5926	137-5930	#137-5930					

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
APTSIZ	= 000200	81-4264	#137-5930							
APTSPO	= 000100	123-5909	137-5930	#137-5930						
ASIZE	005416	#77-4105	87-4428							
ASK4PR	005056	75-3975	#75-4035							
ASK4VC	004742	75-3961	#75-4032							
ASWREG	= 000000	52-3178	52-3178							
AT	= 020000	#44-3074	75-4019	96-4902	96-4906	96-4908	96-4910	96-4913	104-5157	
ATESTN	= 000000	52-3178	52-3178							
AUNIT	= 000000	52-3178	52-3178							
AUSWR	= 000000	52-3178	52-3178							
AVECT1	= 000300	#43-3052	52-3178	52-3178						
AVECT2	= 000000	52-3178	52-3178							
B	037405	76-4069	139-5962	#142-6238						
BAR	002520	#61-3521	68-3760	89-4598	89-4599	91-4662	92-4718	92-4719	92-4726	92-4727
		92-4752	93-4775	93-4781	94-4839	94-4840	94-4847	94-4848	104-5177	105-5207
		106-5253	107-5296	107-5324	107-5339	108-5369	109-5406	110-5444	113-5537	114-5570
		114-5576	115-5600	115-5620	116-5647	117-5694	117-5719	118-5745	119-5786	120-5828
		145-6383	145-6390	145-6393						
BARADR	041554	#142-6271	146-6409							
BCDONE	036477	141-6189	#142-6219							
BCFIN	037440	#142-6246								
BDFAIL	002666	#61-3577								
BDLCR	035712	139-6000	#141-6176							
BDLCRM	037212	141-6178	#142-6230							
BDNERR	036407	140-6017	#142-6218							
BDNUMB	006122	75-4022	*77-4116	*77-4136	77-4141	77-4186	*77-4188	#77-4201		
BDR	002524	#61-3523	89-4602	89-4603	92-4733	92-4734	92-4739	92-4740	92-4763	93-4774
		93-4792	94-4810	94-4811	94-4817	94-4818	100-5004	100-5005	100-5012	100-5013
		106-5254	107-5297	108-5370	115-5601	115-5622	116-5648	116-5671	117-5696	118-5746
		119-5787	119-5810	120-5829	145-6384	145-6391				
BDVECT	002542	#61-3533	*78-4216	*78-4220	145-6397					
BEGIN	011674	87-4427	87-4429	#88-4454						
BEGIN1	012356	48-3173	88-4458	#88-4532	88-4539	88-4541	122-5908			
BITTST	002710	#61-3586	79-4233	*140-6081	*140-6125	*140-6148				
BIT0	= 000001	#43-3049	75-3957	75-4005	75-4007	76-4054	76-4067	77-4110	77-4164	77-4182
		77-4189	82-4284	87-4422	88-4497	88-4532	94-4842	95-4865	95-4868	95-4870
		95-4873	95-4877	95-4880	96-4889	100-5000	111-5479	114-5577	114-5580	139-5960
		140-6081	140-6099	141-6181						
BIT00	= 000001	#43-3049	43-3049	133-5924	133-5924	135-5926	135-5926			
BIT01	= 000002	#43-3049	43-3049							
BIT02	= 000004	#43-3049	43-3049							
BIT03	= 000010	#43-3049	43-3049							
BIT04	= 000020	#43-3049	43-3049							
BIT05	= 000040	#43-3049	43-3049							
BIT06	= 000100	#43-3049	43-3049							
BIT07	= 000200	#43-3049	43-3049							
BIT08	= 000400	#43-3049	43-3049	133-5924						
BIT09	= 001000	#43-3049	43-3049	133-5924	133-5924					
BIT1	= 000002	#43-3049	75-4011	76-4075	82-4289	88-4501	111-5494	139-5968	140-6125	140-6144
BIT10	= 002000	#43-3049	135-5926							
BIT11	= 004000	#43-3049	89-4593	90-4622	91-4643	92-4699	93-4771	94-4804	95-4864	96-4888
		97-4922	98-4947	99-4972	100-4999	101-5025	102-5058	103-5092	104-5148	105-5203

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
		106-5245	107-5287	108-5360	109-5402	110-5441	111-5478	113-5522	114-5568	115-5588
		116-5641	117-5682	118-5738	119-5778	120-5820				
BIT12	= 010000	#43-3049	75-3959	77-4112	77-4166	77-4184	77-4191			
BIT13	= 020000	#43-3049	135-5926							
BIT14	= 040000	#43-3049	111-5504	133-5924						
BIT15	= 100000	#43-3049								
BIT2	= 000004	#43-3049	68-3757	71-3829	75-4021	76-4081	82-4292	88-4505	91-4646	112-5512
		139-5974	140-6148	140-6168						
BIT3	= 000010	#43-3049								
BIT4	= 000020	#43-3049								
BIT5	= 000040	#43-3049								
BIT6	= 000100	#43-3049								
BIT7	= 000200	#43-3049	87-4424	136-5928						
BIT8	= 000400	#43-3049	111-5498	111-5500	111-5502	111-5504				
BIT9	= 001000	#43-3049	92-4714	92-4731	92-4744	94-4822	94-4837	94-4852	135-5926	
BOARD	037424	121-5873	#142-6244							
BORW	002610	#61-3554	*90-4625	*90-4626	90-4633	95-4865	100-5000	111-5479	145-6381	
BPINIT	004154	#74-3905	77-4105	88-4569						
BPT	= 000003	#47-3142	74-3912	87-4415						
BPTINT	004216	74-3905	#74-3920							
BPTVCT	000014	#48-3150	75-3938	*75-3939	*75-3946	*75-3996				
BPTVEC	= 000014	#43-3049								
BRVWPC	037574	76-4052	#142-6249							
BRWAIT	002626	#61-3561								
BUFF	042642	#142-6277								
BUFLEN	002622	#61-3559	66-3714	67-3730	68-3751	68-3752	69-3805	72-3861	*106-5247	106-5248
		*107-5290	107-5291	*108-5363	108-5364	*116-5643	*118-5740	118-5742	*119-5781	119-5783
		*120-5823	120-5825							
BUSERR	= 000004	#43-3050	*89-4595	*89-4615						
BUT	002536	#61-3531	*71-3832	*91-4677	*95-4868	*113-5554	145-6386	145-6387	145-6403	
CAPNTR	044650	*48-3171	64-3667	64-3669	*64-3674	133-5924	133-5924	133-5924	*133-5924	#142-6282
CAPSTK	044652	48-3171	133-5924	133-5924	133-5924	133-5924	142-6282	#142-6283		
CARETN	= 000015	#47-3130	63-3654	140-6015	140-6041	140-6067	140-6090	140-6113	140-6136	140-6159
CAT	= 15777?	#45-3098								
CATCH	006466	48-3150	#78-4215							
CBIT0	= 177776	#46-3104	90-4626	90-4628						
CBIT1	= 177775	#46-3105	111-5488	111-5490						
CBIT10	= 175777	#46-3114								
CBIT11	= 173777	#46-3115								
CBIT12	= 167777	#46-3116								
CBIT13	= 157777	#46-3117								
CBIT14	= 137777	#46-3118								
CBIT15	= 077777	#46-3119								
CBIT2	= 177773	#46-3106								
CBIT3	= 177767	#46-3107								
CBIT4	= 177757	#46-3108								
CBIT5	= 177737	#46-3109								
CBIT6	= 177677	#46-3110								
CBIT7	= 177577	#46-3111								
CBIT8	= 177377	#46-3112								
CBIT9	= 176777	#46-3113								
CB1513	= 057777	#46-3120	96-4901							

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES
CCY	= 177377	#45-3089
CDAB	= 171777	#45-3093
CDAC	= 172777	#45-3094
CDBC	= 174777	#45-3095
CDSA	= 173777	#45-3092
CDSB	= 175777	#45-3091
CDSC	= 176777	#45-3090
CDST	= 170777	#45-3096 97-4936 98-4961 99-4986
CEIR	= 077777	#45-3100
CER	= 077777	#45-3101
CFNC	= 177761	#45-3084 97-4928 98-4953 99-4978 105-5230
CF1	= 177775	#45-3081
CF2	= 177773	#45-3082
CF3	= 177767	#45-3083
CGO	= 177776	#45-3080
CHARCT	032555	63-3621 63-3623 *63-3624 *88-4568 121-5895 *121-5900 *123-5909 *123-5909 123-5909 *123-5909 *123-5909 123-5909 127-5915 127-5915 *127-5915 *127-5915 *127-5915 133-5924 133-5924 *133-5924 *133-5924 *133-5924 133-5924 *133-5924 *133-5924 133-5924 133-5924 *133-5924 #133-5924 *139-5941
CHKBFF	003330	#67-3729 106-5251 107-5294 108-5367
CHKBUF	002620	#61-3558 67-3729 69-3791
CHKCAB	003670	#71-3829 92-4758 93-4787 96-4905 96-4914 101-5033 102-5064 103-5108
CIE	= 177677	#45-3087
CKSWR	= 104407	#132-5921 133-5924 135-5926 135-5926
CLENUP	003646	#70-3817 88-4570 91-4644 94-4805 94-4857 100-5018 102-5059 103-5093 104-5149 105-5206 107-5288 108-5361 109-5403 110-5442 111-5481 113-5523 114-5569 116-5642 119-5779 120-5821 141-6202
CMA	= 167777	#45-3097
CNTLC	= 000003	#47-3129 63-3628 140-6013 140-6035 140-6065 140-6092 140-6111 140-6134 140-6157
CNTRLC	037175	63-3630 #142-6228
CNX	= 137777	#45-3099
CPSAVE	032654	133-5924 *133-5924 133-5924 *133-5924 #134-5924 135-5926 *135-5926 *135-5926 135-5926 136-5928
CR	= 000015	#43-3049 123-5909 123-5909
CRLF	= 000200	#43-3049 75-4033 75-4034 75-4036 77-4204 77-4205 77-4206 89-4593 90-4622 91-4643 92-4699 93-4771 94-4804 95-4864 96-4888 97-4922 98-4947 99-4972 100-4999 101-5025 102-5058 103-5092 104-5148 105-5203 106-5245 107-5287 108-5360 109-5402 110-5441 111-5478 113-5522 114-5568 115-5588 116-5641 117-5682 118-5738 119-5778 120-5820 123-5909 123-5909 133-5924 133-5924 133-5924 133-5924 133-5924 133-5924 133-5924 138-5932 138-5932 142-6213 142-6213 142-6213 142-6214 142-6215 142-6215 142-6215 142-6216 142-6216 142-6216 142-6225 142-6230 142-6231 142-6232 142-6232 142-6245 142-6245 142-6246 142-6246 142-6247 142-6248 142-6249 142-6249 142-6249 142-6250 142-6250 142-6251 142-6252 142-6253 142-6254 142-6255 142-6256 142-6258 142-6259 142-6259 142-6260 142-6260 142-6261 142-6261 142-6261 142-6262 142-6263 142-6264 142-6265 142-6265 142-6266 142-6267 142-6267 142-6269 142-6269 142-6269 142-6270 144-6342 144-6350 144-6352 144-6361 144-6363
CRLF2	037435	139-5979 #142-6245
CRY	= 177577	#45-3088
CSR	002522	#61-3522 68-3750 68-3754 70-3818 70-3819 73-3886 73-3887 89-4600 89-4601 90-4623 90-4624 90-4625 91-4654 91-4655 91-4656 91-4663 91-4664 92-4700 92-4701 92-4702 92-4716 92-4717 92-4725 92-4746 92-4756 92-4762 93-4772 93-4776 93-4777 93-4785 93-4791 94-4809 94-4824 95-4867 95-4869 95-4875

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
		95-4876	96-4891	96-4892	96-4893	96-4898	96-4899	96-4908	96-4909	96-4916
		97-4923	97-4924	97-4925	97-4932	97-4933	98-4948	98-4949	98-4950	98-4957
		98-4958	99-4973	99-4974	99-4975	99-4982	99-4983	100-5002	100-5003	100-5010
		101-5026	101-5027	101-5029	101-5038	101-5040	101-5043	101-5047	102-5060	102-5072
		102-5073	102-5076	102-5081	102-5082	103-5103	103-5104	103-5106	103-5119	103-5121
		103-5124	103-5135	104-5156	104-5157	104-5162	104-5164	104-5169	104-5180	104-5185
		104-5187	104-5192	105-5214	105-5216	105-5219	105-5223	105-5228	106-5246	106-5260
		106-5261	106-5267	106-5269	107-5302	107-5303	107-5318	107-5321	107-5328	107-5329
		107-5334	107-5348	107-5353	108-5362	108-5376	108-5378	108-5381	108-5385	109-5404
		109-5412	109-5413	109-5414	109-5420	109-5422	109-5427	110-5451	110-5452	110-5457
		110-5463	110-5471	111-5482	111-5483	111-5484	113-5529	113-5530	113-5531	113-5538
		113-5539	114-5571	114-5572	115-5597	115-5598	115-5603	115-5604	115-5610	115-5613
		116-5654	116-5655	116-5661	116-5663	116-5664	117-5691	117-5692	117-5697	117-5698
		117-5704	117-5707	117-5708	118-5739	118-5752	118-5753	118-5759	118-5761	118-5762
		119-5780	119-5793	119-5794	119-5800	119-5802	119-5803	120-5822	120-5835	120-5836
		120-5842	120-5844	120-5845	*141-6195	*141-6196	141-6203	141-6205	145-6381	145-6385
		145-6386	145-6387	145-6388	145-6392	145-6394	145-6395	145-6396	145-6399	145-6400
		145-6401	145-6402	145-6403	145-6404	145-6405				
CX6	= 177757	#45-3085								
CX7	= 177737	#45-3086								
CY	= 000400	#44-3064	82-4276	82-4278	91-4668	91-4670	91-4680	91-4682	105-5216	106-5261
		107-5303	107-5329	108-5378	109-5414	113-5543	113-5545	115-5604	116-5655	117-5698
		118-5753	119-5794	120-5836	141-6205					
DAB	= 006000	#44-3069	97-4939							
DAC	= 005000	#44-3070	98-4964							
DATCHK	003526	#69-3790	106-5277	107-5311	108-5393					
DATCHX	003644	69-3804	#69-3809							
DATCH1	003546	#69-3794	69-3806							
DATCH2	003624	69-3796	#69-3805	106-5279	107-5313	108-5395				
DATOCK	003716	#72-3848	118-5769	120-5852						
DATOCX	004062	72-3860	72-3871	#72-3874						
DATOC1	003736	#72-3852	72-3862							
DATOC2	004000	72-3854	#72-3861	118-5771	120-5854					
DBC	= 003000	#44-3071	99-4989							
DDISP	= 177570	#43-3049	52-3178	81-4264						
DDW	002720	#62-3601	68-3757	71-3829	*76-4066	76-4067	76-4075	76-4081	79-4233	80-4245
		*88-4565	88-4566	90-4627	91-4646	96-4889	103-5094	111-5489	112-5512	*139-5949
		139-5960	139-5968	139-5974	*140-6080	*140-6102	*140-6124	*140-6147		
DEVICE	037200	141-6198	#142-6229							
DEVMSK	002544	#61-3534	*88-4532	88-4534	88-4536	*88-4546	*88-4551	*141-6181	141-6185	141-6187
		*141-6192								
DEVPRI	040137	140-6101	#142-6253							
DH1	052446	#144-6342								
DH14	053046	54-3240	54-3247	#144-6350						
DH16	053161	54-3254	#144-6352							
DH17	053274	54-3261	54-3267	54-3273	55-3278	55-3284	55-3296	55-3302	56-3331	56-3337
		56-3343	56-3361	56-3367	60-3500	#144-6354				
DH2	052503	53-3181	#144-6344							
DH202	054645	59-3453	#144-6372							
DH203	054724	#144-6373								
DH207	055010	#144-6374								
DH210	055064	59-3477	#144-6375							

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
DH211	055130	59-3495	#144-6376							
DH23	053350	59-3465	#144-6355							
DH26	053424	57-3390	59-3459	59-3483	#144-6356					
DH3	052544	53-3187	#144-6345							
DH34	053500	55-3320	55-3326	57-3396	59-3471	#144-6357				
DH4	052613	53-3193	53-3217	54-3222	#144-6346					
DH43	053554	56-3349	56-3355	56-3379	57-3384	#144-6358				
DH5	052657	53-3199	54-3228	54-3234	55-3290	#144-6347				
DH50	053620	56-3373	#144-6359							
DH56	053717	59-3489	#144-6360							
DH57	053776	57-3402	59-3446	#144-6361						
DH6	052726	53-3205	55-3308	55-3314	#144-6348					
DH60	054143	57-3409	#144-6363							
DH61	054241	57-3416	#144-6365							
DH62	054312	57-3422	#144-6366							
DH63	054421	57-3428	#144-6368							
DH64	054470	57-3434	#144-6369							
DH65	054527	58-3439	#144-6370							
DH66	054576	#144-6371								
DH7	052772	53-3211	#144-6349							
DIOMEM	002614	#61-3556	109-5406							
DISPLA	001342	#52-3178	*81-4264	*81-4264	133-5924	135-5926				
DISPRE	000174	#48-3156	81-4264							
DOCTS	040373	140-6146	#142-6258							
DOWEPR	026012	121-5865	#121-5893	122-5902						
DREG	002550	#61-3536	*89-4596	*89-4598	*89-4600	*89-4602	*89-4604	145-6380		
DRGET	004232	#75-3932	77-4179							
DRINV	002526	#61-3524	89-4604	89-4605	101-5035	101-5037	101-5044	101-5050	102-5067	102-5069
		102-5077	102-5083	103-5115	103-5117	103-5125	103-5136	104-5150	104-5152	104-5160
		104-5167	104-5174	104-5176	104-5183	104-5190	105-5208	105-5210	105-5220	105-5226
		106-5255	106-5257	106-5265	106-5272	107-5298	107-5300	107-5306	107-5316	107-5325
		107-5327	107-5332	107-5351	108-5371	108-5373	108-5382	108-5388	109-5407	109-5409
		109-5418	109-5425	110-5445	110-5447	110-5455	110-5461	115-5592	115-5594	115-5608
		115-5616	116-5649	116-5651	116-5659	116-5667	117-5686	117-5688	117-5702	117-5711
		118-5747	118-5749	118-5757	118-5765	119-5788	119-5790	119-5798	119-5806	120-5830
		120-5832	120-5840	120-5848	*141-6197					
DRLEV	002552	#61-3537	*88-4566	*88-4567	103-5132	103-5139	145-6399			
DRVS	002530	#61-3525	101-5036	101-5045	101-5051	102-5068	102-5070	102-5078	102-5084	103-5116
		103-5118	103-5126	103-5137	104-5151	104-5153	104-5161	104-5168	104-5175	104-5184
		104-5191	105-5209	105-5211	105-5221	105-5227	106-5256	106-5258	106-526[illegible]	106-5273
		107-5299	107-5301	107-5307	107-5317	107-5326	107-5333	107-5352	108-537[illegible]	108-5374
		108-5383	108-5389	109-5408	109-5410	109-5419	109-5426	110-5446	110-5448	110-5456
		110-5462	115-5593	115-5595	115-5609	115-5617	116-5650	116-5652	116-5660	116-5668
		117-5687	117-5689	117-5703	117-5712	118-5748	118-5750	118-5758	118-5766	119-5789
		119-5791	119-5799	119-5807	120-5831	120-5833	120-5841	120-5849		
DR1WOB	040201	140-6079	#142-6254							
DSA	= 004000	#44-3068	99-4987	99-4990						
DSB	= 002000	#44-3067	98-4962	98-4965						
DSC	= 001000	#44-3066	97-4937	97-4940						
DST	= 007000	#44-3072								
DSWR	= 177570	#43-3049	52-3178	81-4264						
DT1	055240	#145-6379								

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
DT14	055350	54-3242	54-3249	#145-6386						
DT16	055366	54-3256	#145-6387							
DT17	055404	54-3262	54-3268	54-3274	55-3279	55-3285	55-3297	55-3303	56-3332	56-3338
		56-3344	56-3362	56-3368	60-3501	#145-6388				
DT2	055246	53-3182	#145-6380							
DT202	055710	59-3454	#145-6403							
DT203	055726	#145-6404								
DT207	055744	#145-6405								
DT210	055760	59-3478	#145-6406							
DT211	055772	59-3496	#145-6407							
DT23	055420	59-3466	#145-6389							
DT26	055434	57-3391	59-3460	59-3484	#145-6390					
DT3	055260	53-3188	#145-6381							
DT34	055450	55-3321	55-3327	57-3397	59-3472	#145-6391				
DT4	055274	53-3194	53-3218	54-3223	#145-6382					
DT43	055464	56-3350	56-3356	56-3380	57-3385	#145-6392				
DT5	055306	53-3200	54-3229	54-3235	55-3291	#145-6383				
DT50	055476	56-3374	#145-6393							
DT56	055520	59-3490	#145-6394							
DT57	055536	57-3404	59-3448	#145-6395						
DT6	055322	53-3206	55-3309	55-3315	#145-6384					
DT60	055556	57-3411	#145-6396							
DT61	055572	57-3417	#145-6397							
DT62	055606	57-3423	#145-6398							
DT63	055632	57-3429	#145-6399							
DT64	055646	57-3435	#145-6400							
DT65	055660	58-3440	#145-6401							
DT66	055674	#145-6402								
DT7	055334	53-3212	#145-6385							
DYWTLT	006401	77-4123	#77-4207							
EBAR	002600	#61-3550	*68-3762	68-3770	*92-4723	*92-4754	*93-4783	*94-4841	*94-4842	94-4843
		*107-5346	*114-5579	*114-5580	*115-5621	115-5626	*117-5723	145-6383	145-6390	145-6393
		145-6398	145-6407							
EBDR	002576	#61-3549	*94-4815	*100-5008	*100-5011	*100-5016	*115-5623	115-5628	*116-5674	*119-5813
		145-6391	145-6407							
ECELR	040676	139-5942	#142-6262							
ECSR	002572	#61-3547	*68-3763	*68-3764	*68-3765	68-3768	*71-3831	*73-3888	*73-3889	*73-3890
		82-4280	82-4282	82-4287	*91-4659	*91-4665	91-4666	91-4672	*91-4674	91-4675
		*92-4757	92-4759	*93-4786	93-4788	*95-4872	*95-4873	*96-4896	*96-4904	*96-4906
		*96-4912	*96-4913	*97-4926	97-4927	*97-4930	*97-4934	97-4935	*97-4939	*97-4940
		*98-4951	98-4952	*98-4955	*98-4959	98-4960	*98-4964	*98-4965	*99-4976	99-4977
		*99-4980	*99-4984	99-4985	*99-4989	*99-4990	*101-5032	*102-5063	*102-5065	*103-5107
		103-5109	*103-5112	*104-5171	*104-5172	*104-5194	*104-5195	*105-5233	*105-5234	*105-5235
		*107-5337	*109-5432	*109-5433	*110-5468	*110-5469	*113-5534	*113-5540	113-5541	113-5549
		*113-5551	113-5552	*114-5574	145-6385	145-6386	145-6388	145-6398	145-640.	145-6404
EDIT	034572	139-5993	#140-6002	140-6018	140-6034					
EEIR	002574	#61-3548	*95-4879	*95-4880	*111-5497	111-5498	*111-5500	*111-5502	145-6387	145-6401
EIR	= 100000	#44-3076	75-4000	90-4624	95-4875	100-5010	111-5482			
EMTVEC	= 000030	#43-3049	*81-4264	*81-4264						
EM1	047134	#143-6286								
EM10	047445	53-3216	#143-6293							
EM11	047477	54-3221	#143-6294							

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES		
EM12	047542	54-3227	#143-6295	
EM13	047574	54-3233	#143-6296	
EM14	047644	54-3239	#143-6297	
EM15	047733	54-3246	#143-6298	
EM16	047757	54-3253	#143-6299	
EM17	050003	54-3260	#143-6300	
EM2	047162	53-3180	#143-6287	
EM20	050072	54-3266	#143-6301	
EM202	052347	59-3452	#143-6339	
EM21	050140	54-3272	#143-6302	
EM211	052377	59-3494	#143-6340	
EM22	050210	55-3277	#143-6303	
EM23	050251	#143-6304		
EM24	050320	55-3283	#143-6305	
EM25	050351	55-3289	#143-6306	
EM26	050402	59-3458	#143-6307	
EM27	050441	59-3464	#143-6308	
EM3	047216	53-3186	#143-6288	
EM30	050476	55-3295	#143-6309	
EM31	050534	55-3301	#143-6310	
EM32	050600	55-3307	#143-6311	
EM33	050621	55-3313	#143-6312	
EM34	050652	#143-6313		
EM35	050712	55-3319	#143-6314	
EM36	050773	55-3325	59-3470	#143-6315
EM37	051023	56-3330	#143-6316	
EM4	047264	53-3192	#143-6289	
EM40	051061	56-3336	#143-6317	
EM41	051111	56-3342	#143-6318	
EM42	051157	#143-6319		
EM43	051213	56-3348	57-3433	#143-6320
EM44	051244	56-3354	#143-6321	
EM45	051314	56-3360	#143-6322	
EM46	051352	60-3499	#143-6323	
EM47	051427	56-3366	#143-6324	
EM5	047315	53-3198	#143-6290	
EM50	051444	56-3372	#143-6325	
EM51	051511	56-3378	#143-6326	
EM52	051570	57-3383	#143-6327	
EM53	051631	59-3476	#143-6328	
EM54	051657	57-3389	59-3482	#143-6329
EM55	051674	57-3395	#143-6330	
EM56	051714	59-3488	#143-6331	
EM57	051752	57-3401	59-3445	#143-6332
EM6	047346	53-3204	#143-6291	
EM60	052002	57-3408	#143-6333	
EM61	052041	57-3415	#143-6334	
EM62	052124	57-3421	#143-6335	
EM63	052173	57-3427	#143-6336	
EM65	052233	58-3438	#143-6337	
EM66	052304	#143-6338		
EM7	047377	53-3210	#143-6292	

SYMBOL CROSS REFERENCE
CREF V01

SYMBOL		VALUE	REFERENCES								
ENDEV		025622	47-3143	89-4608	112-5514	#121-5858					
ENDSTK		047132	64-3667	133-5924	#142-6284						
ENPR1		002604	#61-3552	*115-5611	*117-5728	145-6394	145-6402				
EOPLOC		002706	*48-3170	#61-3585	*87-4421	88-4571	121-5893	133-5924	*133-5924	*133-5924	*133-5924
			133-5924	133-5924	*133-5924	*133-5924	*133-5924				
EOPRSM		032504	133-5924	#133-5924							
ER	=	100000	#44-3077	68-3764	73-3889	91-4674	96-4902	96-4906	96-4910	96-4913	104-5172
			104-5195	109-5430	109-5433	110-5466	110-5469	113-5551			
ERCAPT		003126	#64-3667	121-5884	122-5902						
ERCHDR		036173	133-5924	#142-6214							
ERRCHK		004064	#73-3886	115-5618	117-5713						
ERRCNT		002714	#61-3588	*91-4651	*94-4806	*106-5276	*107-5310	*108-5392	*113-5524	*115-5590	*117-5684
			*135-5926	136-5928	136-5928	*136-5928					
ERROR	=	104000	#43-3049	78-4221	89-4611	90-4635	91-4660	91-4678	92-4713	92-4730	92-4743
			92-4751	92-4755	92-4761	92-4765	93-4780	93-4784	93-4790	93-4794	94-4821
			94-4836	94-4851	95-4874	95-4881	96-4897	96-4907	96-4915	97-4931	97-4941
			98-4956	98-4966	99-4981	99-4991	100-5009	100-5017	101-5034	101-5046	102-5066
			102-5085	103-5113	103-5131	103-5141	104-5163	104-5173	104-5186	104-5196	105-5222
			105-5236	106-5268	106-5275	106-5278	107-5309	107-5312	107-5319	107-5338	107-5347
			107-5354	108-5384	108-5391	108-5394	109-5421	109-5434	110-5458	110-5470	111-5503
			113-5535	113-5555	114-5575	114-5581	115-5612	115-5619	115-5632	116-5662	116-5670
			116-5675	117-5706	117-5714	117-5718	117-5725	117-5729	118-5760	118-5768	118-5770
			118-5772	119-5801	119-5809	119-5814	120-5843	120-5851	120-5853	120-5855	
ERRVEC	=	000004	#43-3049	43-3050	81-4264	*81-4264	*81-4264	133-5924	*133-5924	*133-5924	*133-5924
ER200		002274	#59-3442	136-5928	136-5928						
ESC	=	000033	#47-3128	63-3632	140-6011	140-6033	140-6063	140-6088	140-6109	140-6132	140-6155
ESCAPE		037064	63-3634	#142-6224							
ETDEV		036350	121-5881	#142-6217							
EWCR		002602	#61-3551	*68-3761	*94-4830	*107-5345	*115-5625	115-5630	145-6389	145-6393	145-6398
			145-6407								
EXPAND		027516	125-5913	#126-5913							
FIXTBL		003170	#65-3684	139-5989	141-6177						
FLAG		002652	#61-3571								
FNC	=	000016	#44-3059	97-4930	98-4955	99-4980	105-5234				
FNCCNT		002654	#61-3572								
F1	=	000002	#44-3056	97-4932	109-5413	117-5697	118-5752	120-5835			
F2	=	000004	#44-3057	75-3942	75-4015	96-4898	96-4908	98-4957	101-5040	102-5073	103-5121
			104-5157								
F3	=	000010	#44-3058	99-4982	115-5603	117-5697	119-5793	120-5835			
GNS	=	******	132-5921	132-5921	132-5921	132-5921	132-5921	132-5921	132-5921	132-5921	132-5921
			132-5921	132-5921	132-5921	132-5921	132-5921	132-5921	132-5921	132-5921	132-5921
			132-5921	132-5921	132-5921	132-5921	132-5922	132-5922			
GO	=	000001	#44-3055	75-3942	82-4276	82-4278	91-4668	91-4670	91-4680	91-4682	92-4716
			101-5040	102-5073	103-5121	104-5156	104-5180	105-5216	106-5261	107-5303	107-5329
			108-5378	109-5414	113-5543	113-5545	114-5571	115-5604	116-5655	117-5698	118-5753
			119-5794	120-5836							
GOAGIN		026344	122-5902	#122-5903							
GTSWR	=	104406	#132-5921								
HAKTPM		032174	133-5924	#133-5924							
HEADER		040452	139-5947	#142-6259							
HT	=	000011	#43-3049	123-5909	123-5909						
IBSAVE		033224	*135-5926	135-5926	*135-5926	*135-5926	135-5926	135-5926	#135-5926		

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES
IE	= 000100	#44-3062 68-3750 73-3886 75-3942 101-5040 102-5073 103-5121 104-5156 105-5216 106-5261 107-5303 107-5329 108-5378 109-5414 115-5603 116-5654 117-5697 118-5752 119-5793 120-5835
INBUF	002616	#61-3557 66-3713 68-3753 69-3792 72-3850 105-5207 106-5253 107-5296 107-5324 107-5342 107-5346 108-5369 116-5647 118-5745 119-5786 120-5828
INBUF1	002656	#61-3573
INDEV	006124	77-4140 #77-4202
INOOUT	022606	111-5480 #112-5512
INTA	003356	#68-3750 106-5274 107-5308 108-5390 116-5669 118-5767 119-5808 120-5850
IOTVEC	= 000020	#43-3049 *81-4264 *81-4264
KDPAR2	= 172364	#47-3137 *88-4495 *88-4499 *88-4503 *88-4515
KDPDR2	= 172324	#47-3134
KIPAR0	= 172340	#47-3135 88-4474
KIPAR2	= 172344	#47-3136 *88-4494 *88-4498 *88-4502 *88-4514
KIPDR0	= 172300	#47-3132 88-4466
KIPDR2	= 172304	#47-3133
LENCHK	002624	#61-3560 *69-3793 *69-3794 69-3805 *72-3851 *72-3852 72-3861
LETNCR	037173	*63-3636 63-3637 #142-6227
LEVEL	002540	#61-3532 *75-3932 *75-3947 *75-3989 75-3995 *103-5127 *103-5132 *103-5138 103-5139 105-5211 106-5258 107-5301 108-5374 109-5410 110-5448 115-5595 116-5652 117-5689 118-5750 119-5791 120-5833 145-6399
LEVELS	004732	75-3933 #75-4030 103-5102
LEVEL3	= 000140	#47-3123 75-4031 101-5028 104-5153
LEVEL4	= 000200	#47-3124 75-4031
LEVEL5	= 000240	#47-3125 75-4030
LEVEL6	= 000300	#47-3126 75-4030
LEVEL7	= 000340	#47-3127 48-3151 70-3817 75-3932 75-3945 75-3991 78-4215 88-4463 88-4492 102-5070 103-5118 138-5932 138-5932 138-5932 140-6121
LF	= 000012	#43-3049 123-5909 123-5909
LODBUF	003302	#66-3713 106-5250 107-5293 108-5366 116-5644 118-5741 119-5782 120-5824
LOOP	002662	#61-3575 *76-4044 76-4056 *76-4087
LRGSTC	002704	#61-3584 63-3640 63-3650 *77-4139 *140-6007 *140-6029
MA	= 010000	#44-3073 68-3755 70-3818 75-3936 75-4003 75-4017 91-4650 91-4654 92-4700 92-4702 92-4762 93-4777 93-4791 94-4809 96-4892 96-4894 96-4896 97-4924 98-4949 99-4974 100-5003 101-5026 101-5038 102-5072 103-5103 103-5119 104-5156 105-5214 106-5260 107-5302 107-5321 107-5328 108-5376 109-5412 110-5451 113-5529 115-5597 116-5663 117-5691 117-5707 118-5761 119-5802 120-5844
MAICLR	007220	#84-4321 113-5526
MAISET	010220	#85-4357 91-4649
MANSIZ	002670	#61-3578 *87-4405 87-4426 *139-5990 *141-6176
MBD	034230	48-3164 #139-5038
MBDIAL	040643	139-5940 #142-6261
MEMGMT	002712	#61-3587 82-4274 82-4284 82-4289 82-4292 *88-4459 *88-4465 *88-4497 *88-4501 *88-4505 113-5547
MEPITM	033514	136-5928 *136-5928 #136-5928
MESSAG	002650	#61-3570
MMPS	= 000252	#47-3139 88-4491 *88-4492 *88-4518
MMR0	= 177572	#47-3131 *88-4464 *88-4493 *88-4512
MMVECT	= 000250	#47-3138 88-4489 *88-4490 *88-4519
MSG	002644	#61-3568
MUSTED	037072	139-5985 #142-6225
M1	041302	88-4524 #142-6267

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
M1A	041421	88-4528	#142-6269							
N	037422	76-4077	139-5972	#142-6243						
NO	037411	76-4085	139-5978	#142-6240						
NOBD	006207	77-4172	#77-4204							
NOBUT	037535	76-4049	140-6002	#142-6248						
NOCARE	056104	88-4496	88-4500	88-4504	91-4662	113-5537	#146-6424			
NODATA	036311	133-5924	#142-6216							
NODVPR	041151	88-4542	#142-6265							
NOMORE	040775	136-5928	#142-6263							
NOTST	006335	77-4160	#77-4206							
NOTVEC	005017	75-3972	#75-4034							
NPR1	002612	61-3553	#61-3555	115-5600	*115-5602	115-5621	117-5694	*117-5695	117-5720	117-5723
		117-5726	145-6394							
NX	= 040000	#44-3075	91-4674	109-5430	109-5433	110-5466	110-5469	113-5551		
NXTTST	002554	#61-3538								
N2	= 000400	#44-3065	75-4009							
OFL	002702	#61-3583								
OLDPC1	002672	#61-3579	*77-4106	77-4197	*88-4460	88-4523	*89-4609	89-4613	145-6380	
OLDPC2	002676	#61-3581	*78-4218	78-4223	*88-4489	88-4519	145-6397			
OLDPS1	002674	#61-3580	*77-4108	77-4196	*78-4217	*88-4462	88-4522	*89-4610	89-4612	
OLDPS2	002700	#61-3582	*78-4219	78-4222	*88-4491	88-4518				
OUTORG	004762	75-3968	#75-4033							
OUTRAN	006255	77-4155	#77-4205							
PASCNT	002556	#61-3539								
PASNUM	041617	#142-6273	146-6416							
PATCHS	011220	#86-4396								
PATRNS	007202	#83-4305	94-4807	115-5591	117-5685					
PFECH	033516	136-5928	#136-5928							
PFECH1	033526	136-5928	#136-5928							
PFECH2	033562	136-5928	#136-5928							
PFECH3	033612	136-5928	#136-5928							
PFECH4	033622	136-5928	#136-5928							
PIRQ	= 177772	#43-3049								
PIRQVE	= 000240	#43-3049								
PNTPRI	006560	76-4073	#80-4245	139-5966	140-6103					
PREOP	026010	*87-4407	*121-5859	121-5868	*121-5870	#121-5891	122-5902	*122-5902		
PROMPT	034244	#139-5941	139-5980	139-5986	139-5995	140-6037	140-6170	141-6190		
PROUT	005107	75-3982	#75-4036							
PR0	= 000000	#43-3049								
PR1	= 000040	#43-3049								
PR2	= 000100	#43-3049								
PR3	= 000140	#43-3049								
PR4	= 000200	#43-3049								
PR5	= 000240	#43-3049								
PR6	= 000300	#43-3049								
PR7	= 000340	#43-3049								
PS	= 177776	#43-3049	43-3049							
PSTATE	006536	#79-4232	140-6082	140-6126	140-6149					
PSW	= 177776	#43-3049	*70-3817	*75-3940	*75-3945	*75-3991	*78-4215	*101-5028	*103-5105	103-5127
		*104-5154	*105-5213	*106-5259	*108-5375	*109-5411	*110-5450	*115-5596	*116-5653	*117-5690
		*118-5751	*119-5792	*120-5834						
PWRVEC	= 000024	#43-3049	*81-4264	*81-4264	*138-5932	*138-5932	*138-5932	*138-5932		

SYMBOL	VALUE	REFERENCES
QTYBRD	002414	#61-3510 65-3687 65-3689 *75-3999 76-4050 *77-4114 77-4121 *77-4129 *87-4430 *87-4434 121-5861 121-5887 122-5902 122-5904 139-5950 140-6003 *140-6023
RA	= 002416	#65-3695 65-3696 *65-3696 *65-3697
RBAR	002566	#61-3544 *68-3760 68-3770 *92-4719 92-4720 *92-4727 92-4728 *92-4752 *93-4781 *94-4840 94-4843 *94-4848 94-4849 *107-5339 107-5342 *114-5576 114-5577 114-5579 *115-5620 115-5626 *117-5719 117-5720 145-6383 145-6390 145-6393 145-6398 145-6407
RBDR	002564	#61-3543 *92-4734 92-4735 *92-4740 92-4741 *92-4763 *93-4792 *94-4811 94-4812 *94-4818 94-4819 *100-5005 100-5006 *100-5013 100-5014 *115-5622 115-5628 *116-5671 116-5672 *119-5810 119-5811 145-6384 145-6391 145-6407
RCSR	002560	#61-3541 *68-3754 68-3755 68-3763 68-3768 *73-3887 73-3888 73-3891 *75-4012 75-4013 *75-4016 75-4019 *91-4656 91-4657 *91-4664 91-4666 91-4675 *92-4717 *92-4725 *92-4756 92-4759 *93-4785 93-4788 *95-4869 95-4870 95-4872 *96-4893 96-4894 *96-4899 96-4900 96-4904 *96-4909 96-4910 96-4912 *97-4925 97-4926 *97-4933 97-4934 *98-4950 98-4951 *98-4958 98-4959 *99-4975 99-4976 *99-4983 99-4984 *101-5029 101-5030 *101-5043 *102-5060 102-5061 *102-5081 *103-5106 103-5109 *104-5162 *104-5169 104-5171 *104-5185 *104-5192 104-5194 *105-5219 *105-5228 105-5229 105-5233 *106-5267 *107-5318 *107-5334 107-5335 *107-5353 *108-5381 *109-5420 *109-5427 109-5428 109-5432 *110-5457 *110-5463 110-5464 110-5468 *113-5531 113-5532 *113-5539 113-5541 113-5552 *114-5572 *115-5610 *116-5661 *117-5704 *118-5759 *119-5800 *120-5842 145-6385 145-6386 145-6388 145-6392 145-6398 145-6402 145-6403 145-6404 145-6405
RDCHR	= 104410	63-3626 129-5915 #132-5921
RDDEC	= 104413	#132-5921
RDLIN	= 104411	130-5917 131-5919 #132-5921
RDOCT	= 104412	#132-5921
RDYCHK	002632	#61-3563
READ	002722	#63-3620 75-3963 75-3977 77-4126 77-4148 139-5944 140-6008 140-6030 140-6060 140-6085 140-6106 140-6129 140-6152
REGADR	002416	#61-3512 65-3695 *75-4025 *75-4026 76-4046 87-4438 88-4554 139-5953 139-5983 139-5996 140-6025 *140-6053 141-6183
RFINIT	012556	#88-4568 122-5906
RFIR	002562	#61-3542 *75-4001 75-4005 75-4009 *95-4876 95-4877 95-4879 *111-5483 111-5485 111-5497 145-6387 145-6401
RESVEC	= 000010	#43-3049
RSTRT	026364	122-5905 #122-5907
RWCR	002570	#61-3545 *68-3766 *92-4704 92-4705 *92-4710 92-4711 *92-4749 *93-4778 *94-4826 94-4827 *94-4833 94-4834 *107-5344 *115-5624 115-5630 *117-5715 145-6382 145-6389 145-6393 145-6398 145-6406 145-6407
RY	= 000200	#44-3063 68-3765 73-3890 73-3892 73-3894 91-4645 92-4757 93-4786 96-4894 96-4896 101-5032 102-5063 102-5065 103-5107 103-5112 109-5430 109-5433 110-5466 110-5469 113-5532 113-5534
R6	=%000006	#43-3049 *81-4264 *81-4264 81-4264
R7	=%000007	#43-3049
SCOPE	= 000004	#43-3049 89-4593 90-4622 91-4643 92-4699 93-4771 94-4804 95-4864 96-4888 97-4922 98-4947 99-4972 100-4999 101-5025 102-5058 103-5092 104-5148 105-5203 106-5245 107-5287 108-5360 109-5402 110-5441 111-5478 113-5522 114-5568 115-5588 116-5641 117-5682 118-5738 119-5778 120-5820 121-5858 122-5902
SDADRS	040306	140-6024 #142-6256
SDRINV	002532	#61-3526 *101-5035 101-5044 101-5050 *102-5067 102-5077 102-5083 *103-5115 103-5125 103-5136 *104-5150 104-5160 104-5167 *104-5174 104-5183 104-5190 *105-5208 105-5220 105-5226 *106-5255 106-5265 106-5272 *107-5298 107-5306 107-5316 *107-5325 107-5332 107-5351 *108-5371 108-5382 108-5388 *109-5407 109-5418 109-5425 *110-5445 110-5455 110-5461 *115-5592 115-5608 115-5616 *116-5649 116-5659 116-5667 *117-5686 117-5702

SYMBOL CROSS REFERENCE

SYMBOL		VALUE	REFERENCES								
			117-5711	*118-5747	118-5757	118-5765	*119-5788	119-5798	119-5806	*120-5830	120-5840
			120-5848								
SDRVS		002534	#61-3527	*101-5036	101-5045	101-5051	*102-5068	102-5078	102-5084	*103-5116	103-5126
			103-5137	*104-5151	104-5161	104-5168	*104-5175	104-5184	104-5191	*105-5209	105-5221
			105-5227	*106-5256	106-5266	106-5273	*107-5299	107-5307	107-5317	*107-5326	107-5333
			107-5352	*108-5372	108-5383	108-5389	*109-5408	109-5419	109-5426	*110-5446	110-5456
			110-5462	*115-5593	115-5609	115-5617	*116-5650	116-5660	116-5668	*117-5687	117-5703
			117-5712	*118-5748	118-5758	118-5766	*119-5789	119-5799	119-5807	*120-5831	120-5841
			120-5849								
SETTUP		006610	#81-4263	87-4417	139-5939						
SPACEC		037400	140-6005	140-6027	140-6058	140-6083	140-6104	140-6127	140-6150	#142-6237	
SPACES		037352	133-5924	133-5924	133-5924	136-5928	#142-6233				
SPACE3		037355	139-5952	#142-6234							
SPACE6		037361	76-4058	76-4061	139-5955	#142-6235					
SPACE7		037370	76-4065	76-4072	76-4074	76-4080	139-5959	139-5965	139-5967	139-5973	#142-6236
STACK	=	001300	#43-3049	81-4264	87-4416	138-5932	139-5938				
STAD		006142	77-4145	#77-4203							
STAGIN		000210	#48-3165								
START		011424	48-3172	#87-4406	139-5991						
START1		011420	48-3163	#87-4405	88-4545	138-5932	146-6422				
STKIFL		036120	133-5924	#142-6213							
STKLMT	=	177774	#43-3049								
SVADRS		040252	140-6054	#142-6255							
SWR		001340	#52-3178	75-3959	77-4112	77-4166	77-4184	77-4191	81-4264	*81-4264	81-4264
			*81-4264	*81-4264	89-4593	90-4622	91-4643	92-4699	92-4714	92-4731	92-4744
			93-4771	94-4804	94-4822	94-4837	94-4852	95-4864	96-4888	97-4922	98-4947
			99-4972	100-4999	101-5025	102-5058	103-5092	104-5148	105-5203	106-5245	107-5287
			108-5360	109-5402	110-5441	111-5478	111-5504	113-5522	114-5568	115-5588	116-5641
			117-5682	118-5738	119-5778	120-5820	127-5915	127-5915	133-5924	133-5924	133-5924
			133-5924	133-5924	133-5924	133-5924	135-5926	135-5926	135-5926	135-5926	136-5928
SWREG		000176	#48-3157	81-4264	127-5915	127-5915					
SW0	=	000001	#43-3049								
SW00	=	000001	#43-3049	43-3049							
SW01	=	000002	#43-3049	43-3049							
SW02	=	000004	#43-3049	43-3049							
SW03	=	000010	#43-3049	43-3049							
SW04	=	000020	#43-3049	43-3049							
SW05	=	000040	#43-3049	43-3049							
SW06	=	000100	#43-3049	43-3049							
SW07	=	000200	#43-3049	43-3049							
SW08	=	000400	#43-3049	43-3049							
SW09	=	001000	#43-3049	43-3049							
SW1	=	000002	#43-3049								
SW10	=	002000	#43-3049								
SW11	=	004000	#43-3049								
SW12	=	010000	#43-3049								
SW13	=	020000	#43-3049								
SW14	=	040000	#43-3049								
SW15	=	100000	#43-3049								
SW2	=	000004	#43-3049								
SW3	=	000010	#43-3049								
SW4	=	000020	#43-3049								

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES							
SW5	= 000040	#43-3049							
SW6	= 000100	#43-3049							
SW7	= 000200	#43-3049							
SW8	= 000400	#43-3049							
SW9	= 001000	#43-3049							
TABINX	002546	#61-3535	*88-4535	*88-4547	88-4552	*88-4553			
TBITVE	= 000014	#43-3049							
TIME	002660	#61-3574	*75-3935	*75-3943	*101-5039	*101-5041	*102-5071	*102-5074	*103-5120	*103-5122
		*104-5155	*104-5158	*104-5179	*104-5181	*105-5215	*105-5217	*106-5262	*106-5263	*107-5289
		*107-5304	*107-5322	*107-5330	*108-5377	*108-5379	*109-5415	*109-5416	*110-5449	*110-5453
		*115-5605	*115-5606	*116-5656	*116-5657	*117-5699	*117-5700	*118-5754	*118-5755	*119-5795
		*119-5796	*120-5837	*120-5838	*141-6204	*141-6206	*141-6210			
TKB	002636	#61-3565	141-6210							
TKS	002634	#61-3564								
TKVEC	= 000060	#43-3049								
TMOPSW	= 000006	#47-3141	77-4108	*77-4109	*77-4196	*87-4415	88-4462	*88-4463	*88-4522	
TORNCB	040051	140-6123	#142-6252							
TOVECT	= 000004	#47-3140	77-4106	*77-4107	*77-4197	87-4414	*87-4414	88-4460	*88-4461	*88-4468
		*88-4488	*88-4513	*88-4523						
TPB	002642	#61-3567								
TPS	002640	#61-3566								
TPVEC	= 000064	#43-3049								
TRAPVE	= 000034	#43-3049	*81-4264	*81-4264						
TRTVEC	= 000014	#43-3049								
TSNUMB	041576	#142-6272	146-6412							
TSTCOM	040341	121-5875	#142-6257							
TSTDEV	012370	#88-4534	88-4549	121-5890						
TSTMM	007066	#82-4274	91-4684							
TST1	012606	88-4572	#89-4593	134-5924						
TST10	015112	95-4866	95-4878	#96-4888	134-5924					
TST11	015346	96-4890	#97-4922	134-5924						
TST12	015532	97-4938	#98-4947	134-5924						
TST13	015716	98-4963	#99-4972	134-5924						
TST14	016102	99-4988	#100-4999	134-5924						
TST15	016272	100-5001	#101-5025	134-5924						
TST16	016524	101-5048	#102-5058	134-5924						
TST17	016762	102-5079	#103-5092	134-5924						
TST2	013004	#90-4622	134-5924							
TST20	017360	103-5133	103-5140	#104-5148	134-5924					
TST21	017764	104-5165	104-5188	104-5193	#105-5203	134-5924				
TST22	020300	105-5224	#106-5245	134-5924						
TST23	020576	#107-5287	134-5924							
TST24	021334	107-5320	107-5349	#108-5360	134-5924					
TST25	021636	#109-5402	134-5924							
TST26	022126	109-5423	109-5431	#110-5441	134-5924					
TST27	022404	110-5459	110-5467	#111-5478	134-5924					
TST3	013134	90-4634	#91-4643	134-5924						
TST30	022622	112-5513	#113-5522	134-5924						
TST31	023110	#114-5568	134-5924							
TST32	023240	114-5578	#115-5588	134-5924						
TST33	023620	#116-5641	134-5924							
TST34	024124	116-5665	116-5673	#117-5682	134-5924					

SYMBOL CROSS REFERENCE

SYMBOL		VALUE	REFERENCES								
TST35		024506	#118-5738	134-5924							
TST36		025004	118-5763	#119-5778	134-5924						
TST37		025320	119-5804	119-5812	#120-5820	134-5924					
TST4		013466	#92-4699	134-5924							
TST40	=	025622	#47-3143	120-5846							
TST5		014172	92-4764	#93-4771	134-5924						
TST6		014366	93-4793	#94-4804	134-5924						
TST7		014734	#95-4864	134-5924							
TWO		037420	76-4079	139-5970	#142-6242						
TYPCNF		005142	#76-4044	77-4198							
TYPDE	=	104414	121-5878	122-5902	#132-5922	133-5924	146-6419				
TYPDS	=	104405	76-4051	76-4057	121-5874	121-5883	122-5902	#132-5921	133-5924	133-5924	139-5951
			140-6004	141-6200							
TYPE	=	104401	63-3630	63-3634	63-3637	63-3657	75-3961	75-3968	75-3972	75-3975	75-3982
			76-4049	76-4052	76-4058	76-4061	76-4065	76-4069	76-4071	76-4072	76-4074
			76-4077	76-4079	76-4080	76-4083	76-4085	76-4086	76-4093	77-4123	77-4140
			77-4145	77-4155	77-4160	77-4172	88-4524	88-4528	88-4542	89-4593	90-4622
			91-4643	92-4699	93-4771	94-4804	95-4864	96-4888	97-4922	98-4947	99-4972
			100-4999	101-5025	102-5058	103-5092	104-5148	105-5203	106-5245	107-5287	108-5360
			109-5402	110-5441	111-5478	113-5522	114-5568	115-5588	116-5641	117-5682	118-5738
			119-5778	120-5820	121-5872	121-5873	121-5875	121-5881	121-5885	122-5902	122-5902
			122-5902	123-5909	124-5911	125-5913	127-5915	127-5915	127-5915	127-5915	127-5915
			127-5915	129-5915	129-5915	129-5915	130-5917	130-5917	131-5919	131-5919	#132-5921
			133-5924	133-5924	133-5924	133-5924	133-5924	133-5924	133-5924	133-5924	133-5924
			133-5924	135-5926	136-5928	136-5928	136-5928	136-5928	136-5928	136-5928	136-5928
			136-5928	138-5932	139-5940	139-5942	139-5947	139-5952	139-5955	139-5959	139-5962
			139-5964	139-5965	139-5967	139-5970	139-5972	139-5973	139-5976	139-5978	139-5979
			139-5985	140-6002	140-6005	140-6017	140-6024	140-6027	140-6038	140-6050	140-6054
			140-6058	140-6072	140-6076	140-6079	140-6083	140-6101	140-6104	140-6123	140-6127
			140-6146	140-6150	141-6178	141-6189	141-6198	141-6201	146-6409	146-6412	146-6416
			146-6420								
TYPOC	=	104402	76-4060	88-4527	127-5915	#132-5921	136-5928	136-5928	139-5954	140-6026	146-6411
TYPON	=	104404	#132-5921								
TYPOS	=	104403	76-4063	77-4142	79-4236	80-4252	#132-5921	139-5957	140-6056	146-6414	
UBHANG		056016	88-4526	#146-6409							
UCAL		037511	141-6201	#142-6247							
VA	=	002456	#65-3698	65-3699	*65-3699	*65-3700					
VCLCHR		037003	140-6076	#142-6223							
VECADR		002456	#61-3513	65-3698	76-4047	77-4119	77-4138	87-4445	88-4561	139-5956	139-5987
			139-5998	140-6055	*140-6078	141-6182					
VECERR		036617	140-6072	#142-6221							
W		037407	76-4071	139-5964	#142-6239						
WCLEN		002630	#61-3562	*106-5248	*106-5249	106-5252	*107-5291	*107-5292	107-5295	107-5323	*108-5364
			*108-5365	108-5368	*116-5645	116-5646	*118-5742	*118-5743	118-5744	*119-5783	*119-5784
			119-5785	*120-5825	*120-5826	120-5827					
WCR		002516	#61-3520	68-3766	88-4555	89-4596	89-4597	91-4661	92-4703	92-4704	92-4709
			92-4710	92-4749	93-4773	93-4778	94-4825	94-4826	94-4832	94-4833	104-5178
			105-5212	106-5252	107-5295	107-5323	107-5340	107-5344	108-5368	109-5405	110-5443
			113-5536	115-5599	115-5624	116-5646	117-5693	117-5715	118-5744	119-5785	120-5827
			145-6382	145-6389	145-6393	145-6397	145-6398	145-6406	145-6407	146-6410	
XCHKBU		043646	61-3558	#142-6280							
XINBUF		042644	61-3557	#142-6278							

SYMBOL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES								
X6	= 000020	#44-3060	82-4280	82-4287	91-4672	109-5413	113-5549			
X7	= 000040	#44-3061	82-4280	82-4282	91-4672	109-5413	113-5549			
YES	037414	76-4083	139-5976	#142-6241						
$APTHD	001000	49-3176	#49-3176							
$ASTAT	= ******	137-5930	137-5930							
$ATYC	033652	137-5930	#137-5930							
$ATY1	033626	#137-5930								
$ATY3	033634	123-5909	#137-5930							
$ATY4	033644	135-5926	#137-5930							
$AUTOB	001334	#52-3178	127-5915	129-5915	129-5915					
$BASE	001464	#52-3178	87-4439							
$BDADR	001322	#52-3178								
$BDDAT	001326	#52-3178								
$BELL	001400	#52-3178	135-5926	135-5926	135-5926					
$CDW1	001470	#52-3178								
$CDW2	001472	#52-3178								
$CHARC	026736	*123-5909	*123-5909	123-5909	*123-5909	#123-5909				
$CKSWR	027764	#127-5915	132-5921	132-5921						
$CMTAG	001300	#52-3178	81-4264	81-4264	81-4264	81-4264	81-4264	81-4264		
$CM3	= 000000	#52-3178	52-3178							
$CM4	000007	#52-3178	52-3178	52-3178	#52-3178	52-3178	52-3178	#52-3178	52-3178	52-3178
		#52-3178	52-3178	52-3178	#52-3178	52-3178	52-3178	#52-3178	52-3178	52-3178
		#52-3178	52-3178	52-3178	#52-3178					
$CNTLG	030544	127-5915	#129-5915							
$CNTLU	030537	127-5915	#129-5915							
$CPUOP	001436	#52-3178								
$CRLF	001405	#52-3178	63-3657	76-4086	76-4093	121-5872	121-5885	122-5902	123-5909	123-5909
		123-5909	127-5915	129-5915	129-5915	130-5917	130-5917	131-5919	131-5919	133-5924
		133-5924	135-5926	135-5926	136-5928	136-5928	136-5928	136-5928	146-6420	
$DBLK	027504	125-5913	125-5913	*125-5913	*125-5913	*125-5913	#125-5913	126-5913	126-5913	126-5913
		126-5913	126-5913							
$DDW0	001474	#52-3178	65-3701	*65-3701	76-4066	77-4118	77-4130	77-4137	88-4565	139-5949
		140-6080	*140-6099	*140-6100	140-6102	*140-6121	*140-6122	140-6124	*140-6144	*140-6145
		140-6147	*140-6168	*140-6169						
$DDW1	001476	#52-3178								
$DDW10	001520	#52-3178								
$DDW11	001522	#52-3178								
$DDW12	001524	#52-3178								
$DDW13	001526	#52-3178								
$DDW14	001530	#52-3178								
$DDW15	001532	#52-3178								
$DDW2	001500	#52-3178								
$DDW3	001502	#52-3178								
$DDW4	001504	#52-3178								
$DDW5	001506	#52-3178								
$DDW6	001510	#52-3178								
$DDW7	001512	#52-3178								
$DDW8	001514	#52-3178								
$DDW9	001516	#52-3178								
$DEVCT	001420	*48-3168	#52-3178	*87-4412	*87-4419	*121-5886	121-5887	*122-5903		
$DEVM	001466	#52-3178	*65-3684	*65-3686	*65-3694	*75-3998	76-4045	*77-4115	*77-4135	87-4431
		88-4534	141-6185							

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
$DOAGN	026534	122-5902	122-5902	#122-5902						
$DTBL	027474	125-5913	#125-5913							
$ENDAD	026324	49-3175	#122-5902	135-5926						
$ENDCT	026102	81-4264	#122-5902							
$ENULL	026340	#122-5902								
$ENV	001430	#52-3178	75-3957	77-4110	77-4164	77-4182	77-4189	87-4422	123-5909	135-5926
		137-5930	137-5930							
$ENVM	001431	#52-3178	81-4264	87-4424	123-5909	123-5909	137-5930			
$EOP	026046	121-5888	#122-5902							
$EOPCT	026074	*81-4264	#122-5902	122-5902						
$ERFLG	001303	#52-3178	133-5924	*133-5924	133-5924	133-5924	*135-5926	135-5926	135-5926	
$ERMAX	001315	#52-3178	*81-4264	*133-5924	133-5924	133-5924				
$ERROR	032656	81-4264	#135-5926							
$ERRPC	001316	#52-3178	*135-5926	*135-5926	135-5926	135-5926	135-5926	136-5928	136-5928	145-6380
		145-6381	145-6382	145-6383	145-6384	145-6385	145-6386	145-6387	145-6388	145-6389
		145-6390	145-6391	145-6392	145-6393	145-6394	145-6395	145-6396	145-6397	145-6398
		145-6399	145-6400	145-6401	145-6402	145-6403	145-6404	145-6405	145-6406	145-6407
$ERRTB	001534	#53-3178	136-5928	136-5928						
$ERRTY	033226	135-5926	#136-5928							
$ERTTL	001312	#52-3178	64-3673	*89-4594	89-4606	121-5863	121-5879	121-5882	122-5902	122-5902
		122-5902	*122-5902	133-5924	*135-5926	135-5926	135-5926			
$ESCAP	001376	#52-3178	*81-4264	*133-5924	135-5926	135-5926	135-5926	135-5926		
$ETABL	001430	#52-3178								
$ETEND	001534	49-3176	#52-3178							
$FATAL	001412	#52-3178	*87-4409	*137-5930						
$FFLG	034072	*137-5930	*137-5930	137-5930	*137-5930	#137-5930				
$FILLC	001356	#52-3178	123-5909	123-5909	123-5909					
$FILLS	001355	#52-3178	123-5909	123-5909						
$GDADR	001320	#52-3178								
$GDDAT	001324	#52-3178								
$GET	026310	122-5902	#122-5902							
$GET42	026314	122-5902	122-5902	#122-5902						
$GTSWR	030064	#127-5915	127-5915	132-5921	132-5921					
$HD =	000000	41-3032	41-3032	41-3032						
$HIBTS	001000	#49-3176								
$HIOCT	031110	*131-5919	#131-5919							
$ICNT	001304	#52-3178								
$INTAG	001335	#52-3178	127-5915	129-5915	129-5915					
$ITEMB	001314	#52-3178	*135-5926	135-5926	135-5926	*135-5926	135-5926	135-5926	135-5926	136-5928
$LF	001406	#52-3178	123-5909	123-5909	129-5915	129-5915	129-5915	130-5917	130-5917	131-5919
		131-5919	135-5926	135-5926						
$LFLG	034071	*137-5930	#137-5930							
$LPADR	001306	#52-3178	*81-4264	133-5924	*133-5924	*133-5924	*133-5924	133-5924	133-5924	133-5924
$LPERR	001310	#52-3178	*81-4264	*89-4593	*90-4622	*91-4643	*92-4699	*92-4707	*92-4722	*92-4737
		*92-4748	*94-4814	*94-4829	*94-4845	*95-4864	*96-4888	*97-4922	*98-4947	*99-4972
		*100-4999	*101-5025	*102-5058	*103-5092	*103-5111	*103-5114	*104-5148	*105-5203	*106-5245
		*107-5287	*108-5360	*109-5402	*110-5441	*111-5478	*113-5522	*114-5568	*115-5588	*116-5641
		*117-5682	*118-5738	*119-5778	*120-5820	133-5924	133-5924	*133-5924	133-5924	133-5924
		135-5926								
$MADR1	001442	#52-3178								
$MADR2	001446	#52-3178								
$MADR3	001452	#52-3178								

SYMBOL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES
$MADR4	001456	#52-3178
$MAIL	001410	49-3176 49-3176 #52-3178 81-4264 123-5909 133-5924 135-5926
$MAMS1	001440	#52-3178
$MAMS2	001444	#52-3178
$MAMS3	001450	#52-3178
$MAMS4	001454	#52-3178
$MBADR	001002	#49-3176
$MFLG	034070	*137-5930 137-5930 *137-5930 #137-5930
$MNEW	030562	127-5915 #129-5915
$MSGAD	001424	*52-3178 *137-5930 137-5930
$MSGLG	001426	#52-3178 *137-5930
$MSGTY	001410	#52-3178 *87-4410 137-5930 *137-5930 137-5930 *137-5930
$MSWR	030551	127-5915 #129-5915
$MTYP1	001441	#52-3178
$MTYP2	001445	#52-3178
$MTYP3	001451	#52-3178
$MTYP4	001455	#52-3178
$NULL	001354	#52-3178 123-5909 123-5909 123-5909
$NUMS	027752	126-5913 #126-5913
$NWTST	= 000001	#88-4593 88-4593 #88-4593 89-4593 #89-4622 89-4622 #89-4622 90-4622 #90-4643 90-4643 #90-4643 91-4643 #91-4699 91-4699 #91-4699 92-4699 #92-4771 92-4771 #92-4771 93-4771 #93-4804 93-4804 #93-4804 94-4804 #94-4864 94-4864 #94-4864 95-4864 #95-4888 95-4888 #95-4888 96-4888 #96-4922 96-4922 #96-4922 97-4922 #97-4947 97-4947 #97-4947 98-4947 #98-4972 98-4972 #98-4972 99-4972 #99-4999 99-4999 #99-4999 100-4999 #100-5025 100-5025 #100-5025 101-5025 #101-5058 101-5058 #101-5058 102-5058 #102-5092 102-5092 #102-5092 103-5092 #103-5148 103-5148 #103-5148 104-5148 #104-5203 104-5203 #104-5203 105-5203 #105-5245 105-5245 #105-5245 106-5245 #106-5287 106-5287 #106-5287 107-5287 #107-5360 107-5360 #107-5360 108-5360 #108-5402 108-5402 #108-5402 109-5402 #109-5441 109-5441 #109-5441 110-5441 #110-5478 110-5478 #110-5478 111-5478 #112-5522 112-5522 #112-5522 113-5522 #113-5568 113-5568 #113-5568 114-5568 #114-5588 114-5588 #114-5588 115-5588 #115-5641 115-5641 #115-5641 116-5641 #116-5682 116-5682 #116-5682 117-5682 #117-5738 117-5738 #117-5738 118-5738 #118-5778 118-5778 #118-5778 119-5778 #119-5820 119-5820 #119-5820 120-5820
$OCNT	027164	*124-5911 *124-5911 #124-5911
$OMODE	027166	*124-5911 *124-5911 124-5911 *124-5911 *124-5911 #124-5911
$OVER	032160	133-5924 133-5924 #133-5924
$PASS	001416	*48-3165 #52-3178 64-3672 *81-4264 *87-4406 88-4454 88-4538 121-5877 *122-5902 *122-5902 122-5902 122-5902 122-5902 122-5902 146-6418
$PASS2	002716	*48-3166 #61-3589 64-3671 *87-4408 88-4456 88-4540 121-5876 *122-5902 122-5902 146-6417
$PASTM	001006	#49-3176
$POWER	034162	138-5932 #138-5932
$PWRDN	034074	81-4264 #138-5932 138-5932
$PWRUP	034112	138-5932 #138-5932
$QUES	001404	#52-3178 123-5909 123-5909 127-5915 129-5915 129-5915 129-5915 130-5917 130-5917 130-5917 131-5919 131-5919 131-5919 135-5926 135-5926
$RDCHR	030302	#128-5915 132-5921 132-5921
$RDDEC	030574	#130-5917 132-5921 132-5921
$RDLIN	030422	#129-5915 132-5921 132-5921
$RDOCT	030752	#131-5919 132-5921 132-5921
$RDSZ	= 000007	#128-5915 128-5915
$RTNAD	026336	#122-5902

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES
$R2A	= ******	132-5921
$SAVRE	= ******	132-5921
$SCOPE	031200	81-4264 #133-5924
$SETUP	= 000137	#48-3152 48-3152 #48-3152 48-3152 #48-3152 48-3152 #48-3152 48-3152 #48-3152
		48-3152 #48-3152 48-3152 #48-3152 81-4264 81-4264 81-4264 81-4264 81-4264
		81-4264 81-4264 81-4264 81-4264 81-4264 81-4264 81-4264 122-5902 122-5902
		127-5915 129-5915 133-5924 135-5926 135-5926 135-5926 135-5926
$STUP	= 177777	#48-3152 #48-3152 48-3152 #48-3152 #48-3152 48-3152 #48-3152 #48-3152 48-3152
		#48-3152 #48-3152 48-3152 #48-3152 #48-3152 48-3152 #48-3152 #48-3152 48-3152
$SVLAD	032124	133-5924 133-5924 #133-5924
$SVPC	= 001000	#49-3175 49-3175
$SWR	= 163400	#41-3021 41-3032 42-3034 42-3034 42-3034 42-3034 42-3034 42-3034 42-3034
		42-3034 52-3178 52-3178 52-3178 81-4264 81-4264 81-4264 81-4264 81-4264
		89-4593 90-4622 91-4643 92-4699 93-4771 94-4804 95-4864 96-4888 97-4922
		98-4947 99-4972 100-4999 101-5025 102-5058 103-5092 104-5148 105-5203 106-5245
		107-5287 108-5360 109-5402 110-5441 111-5478 113-5522 114-5568 115-5588 116-5641
		117-5682 118-5738 119-5778 120-5820 122-5902 122-5902 122-5902 122-5902 122-5902
		133-5924 133-5924 133-5924 133-5924 133-5924 133-5924 133-5924 133-5924 133-5924
		133-5924 133-5924 133-5924 133-5924 133-5924 133-5924 133-5924 133-5924 133-5924
		133-5924 133-5924 135-5926 135-5926 135-5926 135-5926 135-5926 135-5926 135-5926
		135-5926 135-5926 135-5926 135-5926
$SWREG	001432	#52-3178 81-4264
$SWRMK	= 000200	#41-3030 42-3034 42-3034 42-3034 42-3034 42-3034 42-3034 42-3034 42-3034
		42-3034 133-5924 133-5924 133-5924 133-5924 133-5924 133-5924 133-5924 133-5924
		133-5924 133-5924 133-5924
$SW08T	032556	133-5924 #134-5924
$TESTN	001414	*48-3169 #52-3178 *87-4411 *87-4420 *133-5924 *133-5924 136-5928 145-6379 145-6380
		145-6381 145-6382 145-6383 145-6384 145-6385 145-6386 145-6387 145-6388 145-6389
		145-6390 145-6391 145-6392 145-6393 145-6394 145-6395 145-6396 145-6397 145-6398
		145-6399 145-6400 145-6401 145-6402 145-6403 145-6404 145-6405 145-6406 145-6407
		146-6413
$TKB	001346	#52-3178 123-5909 123-5909 127-5915 127-5915 127-5915 127-5915 128-5915 128-5915
		133-5924 133-5924
$TKS	001344	#52-3178 123-5909 123-5909 127-5915 127-5915 127-5915 127-5915 127-5915 128-5915
		128-5915 133-5924 133-5924 141-6208
$TMP0	001360	#52-3178 *73-3892 *81-4263 81-4265 *105-5205 105-5212 *105-5237 114-5570 *117-5705
		126-5913 *126-5913 *126-5913 *126-5913 *126-5913 126-5913 *126-5913 126-5913 *126-5913
		*136-5928 136-5928 136-5928 136-5928 *138-5932 *138-5932 145-6404
$TMP1	001362	#52-3178 *69-3790 69-3797 69-3807 *72-3848 72-3855 72-3869 72-3872 *75-3938
		75-3946 75-3996 *90-4627 *90-4628 *90-4630 *90-4632 90-4633 *91-4645 *91-4648
		91-4657 91-4659 *103-5094 *103-5095 *103-5096 *103-5097 *103-5098 *103-5099 *103-5100
		*117-5717 *117-5724 126-5913 *126-5913 *126-5913 *126-5913 126-5913 *126-5913 *133-5924
		133-5924 *133-5924 145-6379 145-6381 145-6400 145-6405
$TMP2	001364	#52-3178 *69-3798 *72-3856 *72-3865 *105-5204 105-5231 105-5235 *105-5238 *125-5913
		*125-5913 125-5913 125-5913 145-6395 145-6396
$TMP3	001366	#52-3178 *69-3801 *72-3857 *72-3866 145-6395 145-6396
$TMP4	001370	#52-3178 *69-3799 *69-3800 *72-3858 *72-3859 *72-3867 *72-3868 145-6395
$TMP5	001372	#52-3178 *69-3802 *69-3803 145-6395
$TMP6	001374	#52-3178
$TN	= 000040	#41-3022 41-3032 88-4593 89-4593 #89-4593 89-4622 90-4622 #90-4622 *90-4634
		90-4643 91-4643 #91-4643 91-4699 92-4699 #92-4699 *92-4764 92-4771 93-4771
		#93-4771 *93-4793 93-4804 94-4804 #94-4804 94-4864 95-4864 #95-4864 *95-4866

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
		*95-4878	95-4888	96-4888	#96-4888	*96-4890	96-4922	97-4922	#97-4922	*97-4938
		97-4947	98-4947	#98-4947	*98-4963	98-4972	99-4972	#99-4972	*99-4988	99-4999
		100-4999	#100-4999	*100-5001	100-5025	101-5025	#101-5025	101-5048	101-5058	102-5058
		#102-5058	*102-5079	102-5092	103-5092	#103-5092	*103-5133	*103-5140	103-5148	104-5148
		#104-5148	104-5165	104-5188	*104-5193	104-5203	105-5203	#105-5203	105-5224	105-5245
		106-5245	#106-5245	106-5287	107-5287	#107-5287	107-5320	107-5349	107-5360	108-5360
		#108-5360	108-5402	109-5402	#109-5402	109-5423	*109-5431	109-5441	110-5441	#110-5441
		110-5459	*110-5467	110-5478	111-5478	#111-5478	*112-5513	112-5522	113-5522	#113-5522
		113-5568	114-5568	#114-5568	*114-5578	114-5588	115-5588	#115-5588	115-5641	116-5641
		#116-5641	116-5665	*116-5673	116-5682	117-5682	#117-5682	117-5738	118-5738	#118-5738
		118-5763	118-5778	119-5778	#119-5778	119-5804	*119-5812	119-5820	120-5820	#120-5820
		120-5846	133-5924	134-5924						
$TPB	001352	#52-3178	123-5909	123-5909	123-5909					
$TPFLG	001357	#52-3178	123-5909	123-5909	123-5909					
$TPS	001350	#52-3178	123-5909	123-5909	123-5909					
$TRAP	031112	81-4264	#132-5921							
$TRAP2	031134	#132-5921	132-5921							
$TRP	= 000015	#132-5921	132-5921	132-5921	132-5921	132-5921	#132-5921	132-5921	132-5921	132-5921
		132-5921	#132-5921	132-5921	132-5921	132-5921	132-5921	#132-5921	132-5921	132-5921
		132-5921	132-5921	#132-5921	132-5921	132-5921	132-5921	132-5921	#132-5921	132-5921
		132-5921	132-5921	132-5921	#132-5921	132-5921	132-5921	132-5921	132-5921	#132-5921
		132-5921	132-5921	132-5921	132-5921	#132-5921	132-5921	132-5921	132-5921	132-5921
		#132-5921	132-5921	132-5921	132-5921	132-5921	#132-5921	132-5921	132-5921	132-5921
		132-5921	#132-5921	132-5922	132-5922	132-5922	132-5922	#132-5922		
$TRPAD	031146	132-5921	#132-5921							
$TSTM	001004	#49-3176								
$TSTNM	001302	#52-3178	*121-5860	*122-5902	133-5924	*133-5924	*133-5924	133-5924	133-5924	133-5924
		133-5924	135-5926	135-5926	135-5926					
$TTYIN	030530	129-5915	129-5915	129-5915	#129-5915					
$TYPBN	= ******	132-5921								
$TYPD	027204	125-5913	#125-5913							
$TYPDE	027170	#125-5913	132-5922							
$TYPDS	027200	#125-5913	132-5921	132-5921						
$TYPE	026374	#123-5909	132-5921	132-5921	137-5930					
$TYPEC	026606	123-5909	123-5909	123-5909	#123-5909	123-5909	127-5915			
$TYPEX	026740	123-5909	123-5909	#123-5909						
$TYPOC	026766	#124-5911	132-5921	132-5921						
$TYPON	027002	124-5911	#124-5911	132-5921						
$TYPOS	026742	#124-5911	132-5921							
$UNIT	001422	*48-3167	#52-3178	64-3670	*87-4413	*87-4418	*88-4548	121-5874	*121-5889	*122-5907
		*141-6184	*141-6193	141-6199						
$UNITM	001010	#49-3176								
$USWR	001434	#52-3178								
$VECT1	001460	#52-3178	87-4446							
$VECT2	001462	#52-3178								
$XTSTR	031640	#133-5924								
$$GET4	= 000000	#122-5902	122-5902							
$$SW08	= 000040	#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924
		#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924
		#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924
		#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924
		#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
		#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924
		#134-5924	134-5924	134-5924	#134-5924	134-5924	'34-5924	#134-5924	134-5924	134-5924
		#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924
		#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924
		#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924	#134-5924	134-5924	134-5924
		#134-5924	134-5924	134-5924	#134-5924					
$OFILL	027165	*124-5911	*124-5911	124-5911	#124-5911					
$40CAT	= ******	133-5924	135-5926							
.SAV	041640	#142-6275								
.$ASTA	= ******	137-5930	137-5930							
.$X	= 001000	#49-3176	49-3176							

MACRO CROSS REFERENCE

MACRO NAME	REFERENCES									
COMMEN	#43-3049									
ENDCOM	#43-3049									
ESCAPE	#43-3049									
GETPRI	#43-3049									
GETSWR	#43-3049									
MSG1	#88-4586	89-4593								
MSG10	#91-4693	92-4699								
MSG25	#94-4858	#95-4864								
MSG26	#95-4882	#96-4888								
MSG3	#89-4616	90-4622								
MSG32	#93-4795	#94-4804								
MSG340	#96-4917	#97-4922								
MSG341	#97-4942	#98-4947								
MSG342	#98-4967	99-4972								
MSG4	#92-4766	#93-4771								
MSG40	#99-4992	#100-4999								
MSG43	#100-5019	#101-5025								
MSG44	#101-5052	#102-5058								
MSG450	#102-5086	103-5092								
MSG46	#103-5142	104-5148								
MSG47	#104-5197	105-5203								
MSG50	#105-5240	#106-5245								
MSG51	#106-5281	107-5287								
MSG52	#107-5355	#108-5360								
MSG53	#108-5396	109-5402								
MSG54	#109-5435	110-5441								
MSG55	#114-5582	#115-5588								
MSG550	#113-5563	114-5568								
MSG56	#116-5676	117-5682								
MSG57	#115-5636	#116-5641								
MSG60	#117-5733	118-5738								
MSG61	#118-5773	#119-5778								
MSG62	#119-5815	120-5820								
MSG70	#110-5472	111-5478								
MSG71	#112-5515	#113-5522								
MSG710	#90-4636	91-4643								
MULT	#43-3049									
NEWTST	#32-1239	#43-3049	#88-4593	#89-4622	#90-4643	#91-4699	#92-4771	#93-4804	#94-4864	#95-4888
	#96-4922	#97-4947	#98-4972	#99-4999	#100-5025	#101-5058	#102-5092	#103-5148	#104-5203	#105-5245
	#106-5287	#107-5360	#108-5402	#109-5441	#110-5478	#112-5522	#113-5568	#114-5588	#115-5641	#116-5682
	#117-5738	#118-5778	#119-5820							
POP	#41-3027	#43-3049	#125-5913	#130-5917	#131-5919	#137-5930	#137-5930			
PUSH	#41-3027	#43-3049	#125-5913	#130-5917	#131-5919	#137-5930	#137-5930	#137-5930		
REPORT	#43-3049									
SETPRI	#43-3049									
SETTRA	#132-5921	132-5921	132-5921	132-5921	132-5921	132-5921	132-5921	132-5921	132-5921	132-5921
	132-5921	132-5921	132-5922							
SETUP	#41-3027	#43-3049	#81-4264							
SKIP	#41-3014	#43-3049	90-4634	92-4764	93-4793	95-4866	95-4878	96-4890	97-4938	98-4963
	99-4988	100-5001	101-5048	102-5079	103-5133	103-5140	104-5165	104-5188	104-5193	105-5224
	107-5320	107-5349	109-5423	109-5431	110-5459	110-5467	112-5513	114-5578	116-5665	116-5673
	118-5763	119-5804	119-5812	120-5846						

MACRO CROSS REFERENCE

MACRO NAME	REFERENCES									
SLASH	#43-3049									
SPACE	#41-3007	#43-3049								
STARS	#41-3027	#42-3037	#42-3047	#43-3049	#44-3054	#45-3079	#46-3103	#47-3122	#48-3144	#48-3148
	#48-3158	#48-3162	#49-3175	#49-3176	#49-3176	#49-3176	#52-3178	#52-3178	#52-3178	#61-3504
	#61-3508	#61-3517	#61-3519	#61-3528	#61-3530	#62-3591	#63-3603	#63-3619	#64-3660	#64-3666
	#65-3677	#65-3683	#66-3707	#66-3712	#67-3723	#67-3728	#68-3739	#68-3749	#69-3775	#69-3789
	#70-3811	#70-3816	#71-3822	#71-3828	#72-3835	#72-3847	#73-3876	#73-3885	#74-3899	#74-3904
	#75-3923	#75-3931	#75-3950	#75-3956	#76-4039	#76-4043	#77-4096	#77-4104	#78-4210	#78-4214
	#79-4226	#79-4231	#80-4240	#80-4244	#81-4256	#81-4262	#82-4268	#82-4273	#83-4297	#83-4302
	#84-4314	#84-4319	#86-4390	#86-4395	#87-4399	#87-4403	#88-4573	#88-4578	#88-4581	#88-4585
	#89-4593	#89-4593	#90-4622	#90-4622	#91-4643	#91-4643	#92-4699	#92-4699	#93-4771	#93-4771
	#94-4804	#94-4804	#95-4864	#95-4864	#96-4888	#96-4888	#97-4922	#97-4922	#98-4947	#98-4947
	#99-4972	#99-4972	#100-4999	#100-4999	#101-5025	#101-5025	#102-5058	#102-5058	#103-5092	#103-5092
	#104-5148	#104-5148	#105-5203	#105-5203	#106-5245	#106-5245	#107-5287	#107-5287	#108-5360	#108-5360
	#109-5402	#109-5402	#110-5441	#110-5441	#111-5478	#111-5478	#113-5522	#113-5522	#114-5568	#114-5568
	#115-5588	#115-5588	#116-5641	#116-5641	#117-5682	#117-5682	#118-5738	#118-5738	#119-5778	#119-5778
	#120-5820	#120-5820	#121-5857	#122-5902	#123-5909	#124-5911	#125-5913	#126-5913	#127-5915	#127-5915
	#127-5915	#129-5915	#130-5917	#131-5919	#132-5921	#133-5924	#135-5926	#136-5928	#137-5930	#138-5932
	#139-5935	#139-5937	#141-6172	#141-6174						
SWRSU	#43-3049	#81-4264	#81-4264							
TRACK	#41-3010	#89-4593	#90-4622	#91-4643	#92-4699	#95-4864	#96-4888	#97-4922	#98-4947	#99-4972
	#100-4999	#101-5025	#102-5058	#103-5092	#104-5148	#105-5203	#106-5245	#107-5287	#108-5360	#109-5402
	#110-5441	#111-5478	#113-5522	#114-5568	#115-5588	#116-5641	#117-5682	#118-5738	#119-5778	#120-5820
TRMTRP	#132-5921									
TYPBIN	#43-3049									
TYPDEC	#43-3049	121-5874	122-5902							
TYPNAM	#43-3049									
TYPNUM	#43-3049									
TYPOCS	#43-3049									
TYPOCT	#43-3049	127-5915	136-5928							
TYPTXT	#32-1009	#43-3049	122-5902	122-5902						
$$CMRE	#51-3178									
$$CMTM	#51-3178	52-3178	52-3178	52-3178	52-3178	52-3178	52-3178	52-3178		
$$ESCA	#43-3049									
$$NEWT	#32-1271	#43-3049	#88-4593	#89-4622	#90-4643	#91-4699	#92-4771	#93-4804	#94-4864	#95-4888
	#96-4922	#97-4947	#98-4972	#99-4999	#100-5025	#101-5058	#102-5092	#103-5148	#104-5203	#105-5245
	#106-5287	#107-5360	#108-5402	#109-5441	#110-5478	#112-5522	#113-5568	#114-5588	#115-5641	#116-5682
	#117-5738	#118-5778	#119-5820							
$$SET	#132-5921	#132-5921	#132-5921	#132-5921	#132-5921	#132-5921	#132-5921	#132-5921	#132-5921	#132-5921
	#132-5921	#132-5921	#132-5922							
$$SETM	#81-4264	#81-4264								
$$SKIP	#41-3017	#43-3049	90-4634	92-4764	93-4793	95-4866	95-4878	96-4890	97-4938	98-4963
	99-4988	100-5001	101-5048	102-5079	103-5133	103-5140	104-5165	104-5188	104-5193	105-5224
	107-5320	107-5349	109-5423	109-5431	110-5459	110-5467	112-5513	114-5578	116-5665	116-5673
	118-5763	119-5804	119-5812	120-5846						
.EQUAT	#41-3027	43-3049								
.EQUIV	#41-3004	#41-3029	#43-3049	#43-3049	#43-3049	#43-3049	#43-3049	#43-3049	#43-3049	#43-3049
	#43-3049	#43-3049	#43-3049	#43-3049	#43-3049	#43-3049	#43-3049	#43-3049	#43-3049	#43-3049
	#43-3049	#43-3049	#43-3049	#43-3049	#43-3049					
.HEADE	#41-3027	#41-3032								
.SETUP	#41-3027	48-3152								
.SWRHI	#41-3028	#42-3034								

MACRO CROSS REFERENCE

MACRO NAME	REFERENCES		
.SWRLO	#41-3028	#42-3034	42-3035
.$ACT1	#41-3029	#48-3175	
.$APTB	#41-3026	#52-3178	#52-3178
.$APTH	#41-3028	49-3176	
.$APTY	#41-3028	137-5930	
.$CMTA	#41-3028	#50-3178	
.$EOP	#32-1016	122-5902	
.$ERRO	#32-1679	135-5926	
.$ERRT	#33-1880	136-5928	
.$POWE	#41-2983	#138-5932	
.$RDDE	#40-2910	130-5917	
.$RDOC	#39-2830	131-5919	
.$READ	#38-2432	#127-5915	
.$SCOP	#32-1334	#133-5924	
.$TRAP	#41-3027	#132-5921	
.$TYPB	#37-2345		
.$TYPD	#36-2202	125-5913	
.$TYPE	#34-1986	#123-5909	
.$TYPO	#35-2113	#124-5911	