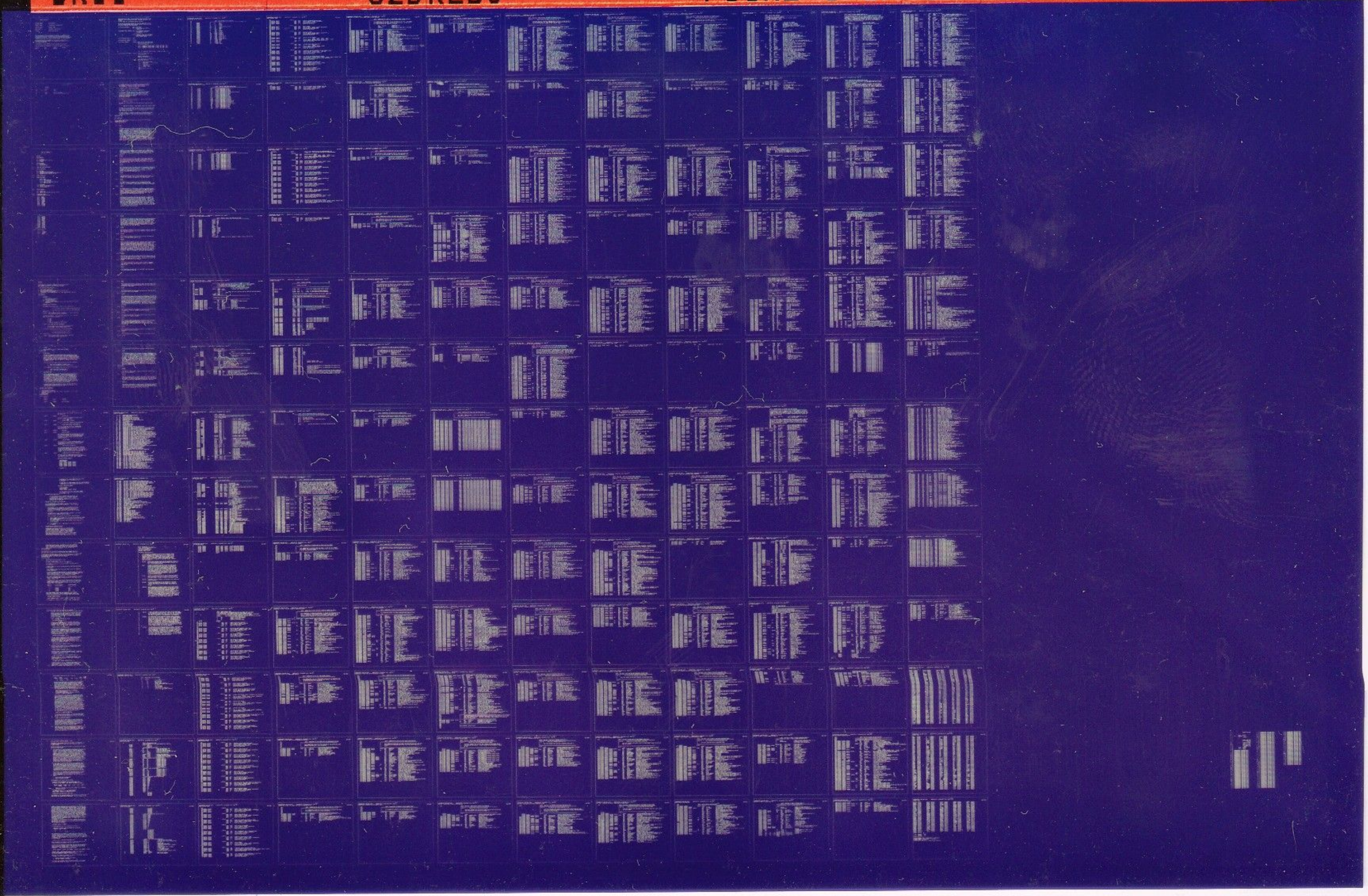
DR11 DR11 GEN NPR INTFC AH-E780B-MC
CZDRLB0 FICHE 1 OF 1



IDENTIFICATION

PRODUCT CODE: AC-E779B-MC

PRODUCT NAME: CZDRLB0 DR11 GEN NPR INTFC

DATE RELEASED: AUGUST, 1980

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: DAN P. MILLEVILLE

HISTORY

REV	DATE	NOTE
A	1977	Initial Release
B	1980	Correction of Coding Errors

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. 52200 words of memory

3.0 TESTING MODES

3.1 DEFINITION

The DR11-W diagnostic accomplishes device register bit tests, internal "LOGIC" wrap-around tests, and with the BC06-R wrap-around cable in J1 and J2, provides external "CABLE" wrap-around tests. In order to FULLY check the module, the diagnostic MUST be run with and without the wraparound cable in place, restarting at address 200 each time, or editing to change the cable mode (See Sect. 7.1.1)

There are only TWO legal modes of operation of this diagnostic:

1. DR11 with no cable(s) in user slots.
2. DR11 with Wrap-Around cable from J1 to J2.

This diagnostic is NOT meant to be run in the following modes:

1. DR11 connected to another DR11.
2. DR11 connected to a user device.

3.2 IMPLEMENTATION

Device register bit tests and internal LOGIC wrap-around tests are executed UNCONDITIONALLY. Cable wrap-around tests are executed ONLY if the BC06-R cable is in place between the J1 and J2 connectors on the DR11-W under test. The presence of this cable is "SIZED" for automatically for each board when the diagnostic is started at address 200. The user *MUST* verify that the "SIZING" occurred correctly by observing the output of the program when starting at 200. (refer to section 5.1 for example) If this summary is not needed, raise bit 12 of the switch register before program execution.

In manual mode (starting address = 204), the user can force uniform testing parameters for ALL modules through use of the edit function (refer to section 7.1.1).

4.0 LOAD AND START PROCEDURE

1. Load program into memory.
2. Load starting address 200, 204 or 210. (See Sects. 7.1, 7.2, 7.3 respectively)
3. Press start.

5.0 SWITCH REGISTER

5.1 Options

SWITCH	OCTAL	FUNCTION
SW15=1	100000	HALT ON ERROR

This will cause the processor to halt at the next error.

SW14=1 040000 LOOP ON TEST

This will cause the processor to loop on the test it is then executing.

SW13=1 020000 INHIBIT ERROR TYPEOUTS

This will cause error typeouts to be inhibited.

SW12=1 010000 DO NOT PRINT BOARD CONFIGURATION

This will cause the list of all boards and their setup data that the autosize routine found to not print.

SW11=1 004000 NOT USED

SW10=1 002000 BELL ON ERROR

This function causes the terminal bell to sound when an error occurs. this can be used in conjunction with LOOP-ON-TEST and INHIBIT-ERROR-TYPEOUTS to see if a loose connection may be causing the error.

SW09=1 001000 LOOP ON ERROR

This function will cause looping on error. It can be used in conjunction with INHIBIT-ERROR-TYPEOUTS when using a scope to find a faulty component.

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

This function causes the CPU to jump to the test in bits <6:0> and execute that test unconditionaly. 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 additional 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 MODES

7.1 MANUAL MODE (STARTING ADDRESS = 204)

Defined as NON-AUTOMATIC use of the diagnostic.

This mode is intended for use in manufacturing when apt is not available.

In manual mode, all DR11-W hardware modules *MUST*BE*CONFIGURED* *AS*FOLLOWS*:

> W/B, PRIORITY LEVEL, 2/N CYCLE and CABLE states set IDENTICAL *IN*ALL*MODULES*.

> All device addresses MUST be set in a series spaced 10 locations apart, starting with the address inputed to the prompt 'STARTING DEVICE ADDRESS XXXXXX :'. (all modules MUST be addressed within the legal address range of 171000 to 177000)

> All vector addresses MUST be set in a series spaced 10 locations apart. (all modules MUST be vectored within the legal vector range of 300 to 770)

> The module with the lowest device address must also have the lowest vector address, the module with the next to the lowest device address must also have the next to the lowest vector address, etc. for example:

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

Only under MANUAL mode does the diagnostic offer 'BURST DATA LATE' calibration. After loading program, depositing SA 204, and pressing START, the program types the following:

```
MULTIPLE BOARD DIALOGUE

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

The program will allow only 1 character input, automatically

printing a <CRLF> when the character is inputed.
7.1.1 when [E] is entered, the program enters the edit function SEQ 0010

NOTE: To exit this routine at any response and return to the MBD prompt, enter CONTROL 'C' (^C). This does nothing but exit the routine, and does not change any values present or changed. To return to the previous prompt, type <ESC>.

'EDIT' responds first by printing:

OF BOARDS UNDER TEST X:

Program accepts a maximum of 2 decimal characters. An appropriate error message is printed if the number inputed is out of range, or an illegal character was inputed. enter <CR> if present value is OK. Next:

STARTING DEVICE ADDRESS XXXXXX :

The user should respond with the lowest device address in the series. Program accepts a maximum of 6 octal digits between 171000 and 177000. An appropriate error message is printed if the number inputed is out of range, or an illegal character was inputed. Enter <CR> if present value is OK. next:

STARTING VECTOR ADDRESS XXX :

The user should respond with the lowest vector address in the series. program accepts a maximum of 3 octal digits between 300 and 777. An appropriate error message is printed if the number inputed is out of range, or an illegal character was inputed. enter <CR> if present value is OK. next:

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

Program accepts either a 0 or 1, repeating the prompt if any other character is inputed. Enter <CR> if present value is OK. next:

DEVICE PRIORITY PRESENT LEVEL = X :

Program accepts 1 character between 0 and 7, repeating the prompt if another character is inputed. Enter <CR> if present value is OK. next:

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

Program accepts a 0 or 1, repeating the prompt if any other character is inputed. Enter <CR> if present value is OK. next:

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

Program accepts a 0 or 1, repeating the prompt if any other character is inputed. Enter <CR> if present value is OK. Then the command prompt is reprinted.

7.1.2 When [L] is entered, the program enters the list function

The diagnostic then prints the following:

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

As previously mentioned, all boards must be spaced 10 address locations apart starting with the 'REGADR' value above, and vectors spaced 10 address locations apart starting with the 'VECADR' value above. The expected W-B, PRIORITY LEVEL, 2-N CYCLE and CABLE test states will be the same for all modules.

7.1.3 When [B] is entered, the program enters the BURST DATA LATE CALIBRATION routine, and the following is typed:

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

This routine will not execute if you have not used edit to deposit a legal starting address and vector address, or the program has already been started at 200. The multiple board dialogue (MBD) prompt will be returned if this is the case. As stated in the DR11 engineering specification, the 'BURST DLT' multivibrator time out must be calibrated so as to be compatible with the user defined transfer rate in burst mode operation. The program software routine sets the cycle bit in the CSR of the DR11, a short delay is executed, and then the cycle bit is cleared. The diagnostic then tests for any character waiting, indicating the user wishes to go on to the next board. If none, it re-executes the setting and clearing of the cycle bit. If a character was inputed, it checks for the next board, and if any, sets up the addresses for that module, then prints the following:

```
DEVICE #  X UNDER CALIBRATION
```

'X' being the device number. It then reaccomplishes the setting and clearing of the cycle bit for that device. If no further modules are found, the message:

```
BURST CALIBRATION COMPLETE
```

is issued, and the MBD prompt is then returned for another command. To accomplish the burst data late calibration, attach a scope probe to E83-7 on the DR11-W (refer to print set M8716-0-1). A positive pulse will be observed. The pulse should be set between 3-30 us. by adjusting pot. R80.

7.1.4 When [R] is entered, the program begins diagnostic test execution. This will be blocked if legal starting device addresses and vector addresses have not been set up. If they are, the register and vector tables are filled, and normal start is executed.

7.2 AUTO-SIZE MODE (STARTING ADDRESS = 200)

This mode is the normal field service mode. It supports standalone operation as well as script operation under ACT11 or XXDP (chain).

The DR11 diagnostic has the following run characteristics when operating in auto mode:

a. The program will test the boards recognized by the autosize routine. The autosize routine will look at addresses between 172414 and 172604 in steps of 10 (20 octal locations). It will initially determine if the location it found to exist is a DR11 by forcing an interrupt. IF THE BOARD FAILS TO INTERRUPT, YOU MUST USE MANUAL MODE TO FORCE TEST EXECUTION OF THAT MODULE. The purpose of this initial test is to eliminate testing a module that is not a DR11, and determine the interrupt priority and vector of that module. The only legal interrupt vectors the DR11 can be set up for are as follows: 40, 50-174, and 254-774, all in steps of 4. Each board can have a vector anywhere in the stated ranges with no restrictions, allowing complete flexibility in the test sequence.

   In the case of multiple DR11-W's on the same CPU, each DR11-W MUST have its own unique DEVICE/VECTOR addresses. There are no constraints that the boards must start with the first device address 172410, or that multiple boards are assigned consecutive device addresses. When operating in auto-size mode, the user should verify the "SIZED" configuration by knowing how the boards are set up and comparing with the autosize output when starting at 200.

   Auto-sizing will determine the Interrupt priority, Interrupt vector, W/B, 2/N CYCLE, and Cable states of each board, independent of the states of other boards.

b. The following will NOT be offered to the user in autosize mode:

   1. BURST DATA LATE CALIBRATION
   2. MULTIPLE BOARD DIALOGUE

The diagnostic will print the following:

```
DIAGNOSTIC HAS DETERMINED THE FOLLOWING ABOUT THE
DR11-W(S) IT HAS FOUND.  USER *MUST* DETERMINE ACCURACY

   BOARD#  REGADR   VECADR   W/B    P-LEV   2-N CY  CABLE
     X     XXXXXX     XXX     X       X       X       X

      Data will continue to print until data for all modules
      has been printed.

(^X) INHIBITS EOP'S, (^Y) FOR ERROR SUMMARY
UNIBUS HANG?  RESTART AT ADDRESS XXXXXX

CZDRLB0 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 less paper. 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 get 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.
The CONTROL Y (^Y) function calls for a summary of device(s) and pass(es) that had errors. If no errors occured since the beginning of the diagnostic, or since the last error report, the following is printed:

```
  NO ERROR TOTALS TO REPORT
```

If there were errors, the following is printed:

```
  SUMMATION OF ERRORS SINCE BEGINNING OR LAST REPORT
    BOARD #  PASS #  ERRTTL
       X        X        X
       X        X        X
       X        X        X (etc.)
```

The information is stored on a stack that will hold up to 150 (decimal) device-pass error data lines above. If the limit is reached, diagnostic will continue, but further data will not be stored. The data accumulated is not written over, but when (^Y) is entered, the following is printed just before the 'SUMMATION...' statement above:

```
  STACK IS FULL - DATA MAY HAVE BEEN LOST
```

When the data is printed, the stack is reinitialized and will start storing up to another 150 error data lines.

In the event the UNIBUS becomes hung, and you have non-volatile memory or battery backup, restart the program at the address specified by the 'UNIBUS HUNG...' prompt at the start of the diagnostic. The printout will be as follows:

```
  DEVICE ADDRESS - XXXXXX, TEST NUMBER - XXXXXX, PASS NUMBER - XXXXXX
```

CPU will halt. Hitting continue will cause the program to restart as though you had started at 200.

7.3 RESTARTING PROGRAM IN MEMORY (STARTING ADDRESS = 210)

Whenever the program is halted, all history of previous testing is saved. It will remain intact until:

1. Another program is loaded into memory
2. The user re-edits the table

To restart the program, enter SA 210 and start. This start precludes any setup and negates the start message obtained

when starting at 200. Do not start at this location if the diagnostic has not been previously "STARTED" at either 200 or 204.

7.4 TESTING UNDER APT (AUTOMATED PRODUCT TESTING)

To set up for multiple boards for testing under APT control, the APT system manager should answer the APT queries to the following items as indicated below:

SOFTWARE ENVIRONMENT: 000 - Dump mode
001 - Script mode (apt monitors diagnostic)

ENVIRONMENT MODE ($ENVM): 000 - Let diagnostic auto-size configurator and test accordingly

200 - Diagnostic must use configuration specified by APT ($VECT1, $VASE, $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 for each device:

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

Each descriptor word MUST be set up as follows:

BIT 0 - DR11-W or -B Mode (W=0, B=1)
BIT 1 - 2/N CYCLE (0=2 CY, 1=N CY)
BIT 2 - Cable Tests (0=NO, 1=YES)
BIT 5 \
BIT 6 > Device Priority
BIT 7 /

8.0 MISCELLANEOUS

8.1 POWER FAIL

If a power failure occurs and battery backup maintains the program in memory, or a non-volatile memory exists, the program will restart printing the following:

POWER FAILURE - RESTARTING PROGRAM

The diagnosic will then restart at address 210.

If CPU is turned off while running, the above procedure is followed. If the processor is halted first, then turned off, the processor will come back up halted. To restart the program, hit CONTINUE, and the remaining procedure is the same as above.

8.2 END-OF-PASS MESSAGE

--- -- ---- -------

The EOP will print as follows with no errors on that pass:

END PASS # XXXXXX

The EOP will print as follows with some errors when testing 1 device:

END PASS # XXXXXX TOTAL ERRORS SINCE LAST REPORT XXXXXX

The EOP will print the same as with no errors on any particular pass when testing more than one device and one or more devices has failed, since 'total errors' is meaningless and will more than likely be incorrect.

The pass number is capable of going up to 99,999,999 decimal, or about 3 months running with EOP disabled and no errors. In other words, 32767 is not the limit as with other diagnostics.

9.0 EXECUTION TIME

On a PDP11/44:

In all modes: Approximately 8 passes per second with EOP messages disabled and no errors.

10.0 SUBROUTINE ABSTRACTS

10.1 READ

The READ subroutine is used in the EDIT routine to input up to 6 digits in octal, 2 digits in decimal, or a single non-numeric character. R4 is used as the location to hold the number in octal, and is cleared for that purpose at the start of the subroutine. R3 is to be preloaded with the number of digits expected, since a <CRLF> is printed when the limit is reached. Entering a <CRLF> before the limit is acceptable, as it will be interpreted as a non-numeric character and exit. In any case, the last inputed ASCII character is left in location 'ANSWER'. If a numeric character is inputed, it will clear all but the 1st 4 bits in location 'ANSWER', exposing the value of the digit inputed, rotate R4 to the left 3 places to make room for the inputed digit, and add it to R4. Location

'LRGSTC' is to be loaded with the largest ASCII number digit acceptable for this number, I.E. 7 or 9 (for octal or decimal input respectively). ANY character outside ASCII '0' or '7/9' is treated as a non-numeric, triggering an automatic <CRLF> and exit.

10.2 ERCAPT

This subroutine saves the unit number, pass number and total errors for that device/pass whenever it encountered errors. This routine saves data for 150 (decimal) passes. If the stack should become full, data starting with the 151st pass containing errors is lost.

10.3 PTCAPT

This subroutine prints the data stored by the ERCAPT subroutine and resets the special stack pointer. If no data was stored, a message stating no data was stored is printed. If the stack is found full, a message announcing this finding is printed, warning that data may have been lost.

10.4 FIXTBL

This subroutine fills the 17 octal locations starting at 'REGADR' and 'VECADR' from the starting values already loaded in the first locations in steps of 10 for each table.

10.5 LODBUF

The INBUF buffer is loaded with an incrementing pattern (0,1,2,3,...) beginning at the starting address of INBUF. The number of words loaded is determined by the contents of BUFLEN.

10.6 CHKBFF

The CHKBUF buffer is loaded with a modified incrementing pattern (0,0,2,2,4,4,6,6,...) beginning at the starting address of CHKBUF. The number of words loaded is determined by the contents of BUFLEN. This buffer is loaded only for tests which use the maintenance mode of the DR11-W which has a special alternating DATI-DATO sequence of operation.

10.7 INTA

The IE bit is cleared in the CSR then the CSR is checked for the absence of the error bit and the presence of READY. The WCR is checked to see that it is equal to zero. The correct contents of the BAR are calculated and checked. The program will fail to update the PC return address by 2 if ERROR is set, READY is clear, READY and ERROR are clear of the CSR, WCR is not zero or the BAR contents is not zero. This will call the error that is just after the jsr call in the test. If all data is acceptable, the PC is updated, and the return from the subroutine is after the error call.

10.8 DATCHK

This routine is entered to check inbuf after a maintenance mode operation. The contents of INBUF and the contents of CHKBUF are checked to see that they are the same. The number of comparisons made is determined by the contents of BUFLEN. Any errors result in an RTS to the test to call the error there. A JSR back to the subroutine is executed to resume its checking. When returning, SP return address is updated by 6 to return after the error call and JSR return.

10.9 CLENUP

The routine is entered at the end of several tests to clear any data that may have been left in any registers, and to restore the interrupt vectors.

10.10 CHKCAB

This routine is used in various tests to alter the expected data if the WRAP-AROUND cable is out.

10.11 DATOCK

After a string of DATO'S has been completed this routine checks that the correct data pattern was transferred to INBUF. The number of comparisons made is determined by the contents of BUFLEN. An error in the check results in an RTS to the test to call the first error after the JSR call, where a JSR returns control back to the subroutine for further checking. An additional check is made on BUFLEN+2 to insure that not too many words were transferred. If they were, the PC return address is altered so that the second error after the JSR is called. If no errors, return is altered to just after the second error call.

10.12 ERRCHK

This routine clears IE and updates the PC for return after the error in the test if error is clear. If set, return is executed without updating the PC return so the error call after the JSR call in the test will be called.

10.13 DEVADS

This routine generates an address table located at REGADR starting with the base device address (contents of $BASE) in steps of 10.

10.14 BPINIT

This subroutine reloads the ".+2" and "BPT" into the unused locations between 4 and 776.

10.15 DRGET

This subroutine extracts information about the DR11 that interrupted and loads the accumulated data into the device descriptor word for that board.

10.16 TYPCNF

This subroutine prints the board configurations that the ASIZE subroutine found on the UNIBUS.

10.17 CHK4DR

This subroutine checks for a location as belonging to a DR11 by trying to force an interrupt a total of 4 times with the CPU at priorities 6 through 3. If the attempt fails, routine corrects the stack to return after the DR11 extraction routine. If the location does belong to a DR11, the return address of the subroutine on the stack is moved down one location, and the address+4 of the interrupt vector of the DR11 is put in the return addresses previous location. The stack is then popped 3 times by adjusting the pointer up 6, and a normal return is executed.

10.18 ASIZE

This routine autosizes the board configuration and prints the configuration if bit 12 (10000) is set in the SWR.

10.19 VCTADS

This routine generates the vector address table starting with the address in location 'VECADR'.

10.20 CATCH

This routine reports unexpected or erroneous traps or interrupts through the BREAK-POINT-TRAP loaded in locations 4-776. The stack is cleaned 4 times before the error call, and restored twice after the error call for returning to the source of the trap.

10.21 PSTATE

This routine prints the state of the bit in the DDW that was preloaded in location 'BITTST'.

10.22 PNTPRI

This routine prints the device priority in the DDW location.

10.23 SETTUP

This subroutine initializes the trap and interrupt vectors.

10.24 TSTMM

This subroutine checks for existence of Memory Management and if it exists, checks for the error condition of no memory location, but no ERROR and NEX bit sets. If memory management is not there, an exit updating the return address by 2 is done. If there, the XBA16 and XBA17 bits of the expected data are checked. If both zero, an exit updating the return address by 2 is done. If either or both are set, the upper byte of the Memory Management location is checked for the existence of upper memory, initialized at the beginning of the diagnostic. If not there (bits 0, 1 or 2 of upper byte clear), a normal exit is executed so the branch immediately following the JSR call will cause a check for the error bits in the expected to be set for another check.

11.0 DATA STACKS

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 possible 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.

TABLE OF CONTENTS

```
     1                  .NLIST  MC,MD,CND
    39                  .TITLE  CZDRLB0-DR11 GEN NPR INTFC
                        ;*COPYRIGHT (C) 1980
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY DAN MILLEVILLE
                        ;*
                        ;*      THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*      PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977, MODIFIED FOR
                        ;*      THE CZDRLB DIAGNOSTIC.  CHANGES ARE NOTED BY [;&& ] IN THE
                        ;*      COMMENT FIELDS OF THE $SCOPE, $EOP, $TYPE, $ERROR, $CKSWR
                        ;*      $PWRDN, $PWRUP (ELIMINATED), $ERRTYP AND $TYPDS ROUTINES.
                        ;*      NEWTST WAS MODIFIED IN FUNCTION AS DESCRIBED BELOW.  CHANGES
                        ;*      WERE AS FOLLOWS:
                        ;*
                        ;*      $SCOPE:   END-OF-PASS MESSAGE DISABLING AND REENABLING CAPABILITY,
                        ;*                PLUS UPDATING $TESTN JUST AFTER $TSTNM IS UPDATED, AS
                        ;*                WELL AS ANY NEEDED MESSAGES.
                        ;*
                        ;*      $EOP:     CHANGED TO RECOGNIZE WHETHER OR NOT THE USER WISHES
                        ;*                THE EOP MESSAGES PRINTED, AND PRINTS THE NUMBER OF
                        ;*                ERRORS IN THAT PASS IF THERE WERE ANY, AND TYPES AN
                        ;*                EXTRA <CRLF> IF THERE WERE ERRORS TO SPACE EOP FROM
                        ;*                THE ERROR.  CHANGED ALSO TO RECOGNIZE WHEN THE PASS
                        ;*                COUNT GOES NEGATIVE, AND IF SO, CLEARS THE PASS COUNT,
                        ;*                AND INCREMENTS THE NEXT LOCATION AFTER $PASS TO COUNT
                        ;*                A BLOCK OF 32768 PASSES HAS OCCURED.  THIS LOCATION
                        ;*                IS USED IN CONJUNCTION WITH $PASS TO PRINT UP TO
                        ;*                PASS # 99,999,999 DECIMAL.
                        ;*
                        ;*      $TYPE:    INSTEAD OF USING THE STACK TO LOAD A CHARACTER INPUTED
                        ;*                WHILE PRINTING, LOOSING THE CHARACTER IN THE PROCESS,
                        ;*                LOCATION 'CHARCT' IS USED TO SAVE IT FOR $SCOPE TO USE.
                        ;*
                        ;*      $ERROR:   INCREMENT LOCATION 'ERRCNT' FOR POSSIBLE USE IN MULTIPLE
                        ;*                ERROR PRINTOUTS.
                        ;*
                        ;*      $CKSWR:   TESTING AND PROCESSING OF THE CONTENTS OF LOCATION
                        ;*                'CHARCT' WAS ADDED TO INCREASE CHANCES OF DIAGNOSTIC
                        ;*                CATCHING USER REQUEST FOR A SOFTWARE SWITCH REGISTER
                        ;*                CHANGE.
                        ;*
                        ;*      $ERRTYP:  ADDED CAPABILITY TO PROCESS ERRORS WITHIN LOOPS (ERRORS
                        ;*                WITH ERROR NUMBERS BETWEEN 201-377) SO THE MESSAGE AND
                        ;*                DATA HEADER ARE PRINTED DURING THE 1ST ERROR ONLY, WITH
                        ;*                DATA ONLY PRINTED FOR 2ND AND SUBSEQUENT ERRORS.  IT
                        ;*                CANCELS DATA PRINTING AFTER 20 (OCTAL) ERRORS HAVE BEEN
                        ;*                DONE SO AS TO ELIMINATE MASSIVE ERROR TYPEOUTS, BUT
                        ;*                CONTINUES TO TALLY THE ERRORS IN LOCATION '$ERTTL' SO THE
                        ;*                EOP MESSAGE WILL SHOW THE TOTAL NUMBER OF ERRORS IN THAT
                        ;*                PASS.  IF AN ERROR NUMBER BELOW 201 IS CALLED, 'ERRCNT'
                        ;*                IS CLEARED SO IF THIS ERROR IS IN A LOOP, ANY SUBSEQUENT
                        ;*                201+ ERRORS WILL HAVE THERE HEADER REPRINTED.
```

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

```
41                              .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        14            LOOP ON TEST
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        12            DO NOT PRINT BOARD CONFIGURATION
                                ;*        11            NOT USED
                                ;*        10            BELL ON ERROR
                                ;*         9            LOOP ON ERROR
                                ;*         8            LOOP ON TEST IN SWR<6:0>
42                              ;*         7            INHIBIT MULTIPLE ERROR TYPEOUTS
```

```
     44                         .SBTTL  BASIC DEFINIT.JNS
                                ;*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
     45         000004                  BUSERR  =ERRVEC
     46         172410                  ABASE   =172410         ;BASE DEVICE ADDRESS
     47         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
```

```
                                .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
```

```
                                .SBTTL  COMPLEMENTS OF BIT DEFINITIONS
                        ;*****************************************************************
                        ;
177776          CBIT0   =177776 ;COMPLIMENT OF BIT0
177775          CBIT1   =177775 ;COMPLIMENT OF BIT1
177773          CBIT2   =177773 ;COMPLIMENT OF BIT2
177767          CBIT3   =177767 ;COMPLIMENT OF BIT3
177757          CBIT4   =177757 ;COMPLIMENT OF BIT4
177737          CBIT5   =177737 ;COMPLIMENT OF BIT5
177677          CBIT6   =177677 ;COMPLIMENT OF BIT6
177577          CBIT7   =177577 ;COMPLIMENT OF BIT7
177377          CBIT8   =177377 ;COMPLIMENT OF BIT8
176777          C9IT9   =176777 ;COMPLIMENT OF BIT9
175777          CBIT10  =175777 ;COMPLIMENT OF BIT10
173777          CBIT11  =173777 ;COMPLIMENT OF BIT11
167777          CBIT12  =167777 ;COMPLIMENT OF BIT12
157777          CBIT13  =157777 ;COMPLIMENT OF BIT13
137777          CBIT14  =137777 ;COMPLIMENT OF BIT14
077777          CBIT15  =77777  ;COMPLIMENT OF BIT15
057777          CB1513  =57777  ;COMPLIMENT OF BIT15 & BIT 13
```

```
                                .SBTTL  PRIORITY LEVELS AND OTHER DEFINITIONS
                        ;******************************************************************
000140                  LEVEL3  =14C
000200                  LEVEL4  =200
000240                  LEVEL5  =240
000300                  LEVEL6  =300
000340                  LEVEL7  =340
000033                  ESC     =33
000003                  CNTLC   =3
000015                  CARETN  =15
177572                  MMR0    =177572
172304                  K!PDR2  =172304
172324                  KDPDR2  =172324
172344                  KIPAR2  =172344
172364                  KDPAR2  =172364
000250                  MMVECT  =250
000252                  MMPS    =252
000004                  TOVECT  =4
000006                  TMOPSW  =6
000003                  BPT     =3
023074                  TST40=ENDEV     ;BRANCH TO TEST 40 = BRANCH TO ENDEV (THERE IS NO TEST 40)
```

```
                                ;*****************************************************************
                                ;* ALL UNUSED LOCATIONS FROM 4-776 WILL CONTAIN A ".+2,BPT" SEQUENCE
                                ;* TO CATCH ILLEGAL TRAPS & INTERRUPTS TO THE 'CATCH' LOCATION
                                ;* 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                                ;*****************************************************************
000014                          .=14                    ;THE BPT TRAP VECTOR POINTS TO THE
000014 005536                   BPTVCT: .WORD   CATCH   ;   ILLEGAL TRAP HANDLER "CATCH"
000016 000340                           .WORD   LEVEL7
000042                          .=42
000042 000000                           .WORD   0       ;CLEAR THIS LOCATION (FOR APT MONITOR STARTING ADDRESS)
000174                          .=174
000174 000000                   DISPRE: .WORD   0
000176 000000                   SWREG:  .WORD   0
                                ;*****************************************************************
                                ;
                                ;PROGRAM STARTING LOCATIONS
                                ;
                                ;*****************************************************************
000200 000137 010266                    JMP     START1  ;NORMAL START
000204 000137 030644                    JMP     MBD     ;ENTER MULTIPLE BOARD DIALOGUE
000210 005037 001416            STAGIN: CLR     $PASS   ;CLEAR $PASS
000214 005037 001420                    CLR     $PASS+2 ;CLEAR $PASS+2
000220 005037 001424                    CLR     $UNIT   ;CLEAR $UNIT
000224 005037 001422                    CLR     $DEVCT  ;CLEAR $DEVCT
000230 005037 001414                    CLR     $TESTN  ;CLEAR $TESTN
000234 005037 002710                    CLR     EOPLOC  ;CLEAR EOPLOC
000240 012737 041172 041170             MOV     #CAPSTK,CAPNTR  ;RESET THE CAPTURE POINTER
000246 000137 011006                    JMP     BEGIN1  ;JUMP TO BEGIN1 FOR RESTART WITHOUT HEADER PRINTING
001000                          .=1000
```

```
    167                                 .SBTTL  ACT11 HOOKS
                                        ;*****************************************************************
                                        ;HOOKS REQUIRED BY ACT11
                001000                          $SVPC=.                 ;SAVE PC
                000046                          .=46
            000046  023464                      $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                000052                          .=52
            000052  000000                      .WORD   0               ;;2)SET LOC.52 TO ZERO
                001000                          .=$SVPC                 ;; RESTORE PC
    168                                 .SBTTL  APT PARAMETER BLOCK
                                        ;*****************************************************************
                                        ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                        ;*****************************************************************
                001000                          .$X=.   ;;SAVE CURRENT LOCATION
                000024                          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
            000024  000200                      200     ;;FOR APT START UP
                000044                          .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
            000044  001000                      $APTHDR ;;POINT TO APT HEADER BLOCK
                001000                          .=.$X   ;;RESET LOCATION COUNTER
                                        ;*****************************************************************
                                        ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                        ;INTERFACE SPEC.
                001000                  $APTHD:
                001000  000000          $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
                001002  001410          $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
                001004  000010          $TSTM:  .WORD   10      ;;RUN TIM OF LONGEST TEST
                001006  000010          $PASTM: .WORD   10      ;;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  000053                  .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
    170                                 .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  000000          $PASS:  .WORD   APASS,0  ;;PASS COUNT ;&& ADDITIONAL WORD LOCATION FOR OVERFLOW
001422  000000                  $DEVCT: .WORD   ADEVCT   ;;DEVICE COUNT
001424  000000                  $UNIT:  .WORD   AUNIT    ;;I/O UNIT NUMBER
001426  000000                  $MSGAD: .WORD   AMSGAD   ;;MESSAGE ADDRESS
001430  000000                  $MSGLG: .WORD   AMSGLG   ;;MESSAGE LENGTH
001432                          $ETABLE:                        ;;APT ENVIRONMENT TABLE
001432     000                  $ENV:   .BYTE   AENV     ;;ENVIRONMENT BYTE
001433     000                  $ENVM:  .BYTE   AENVM    ;;ENVIRONMENT MODE BITS
001434  000000                  $SWREG: .WORD   ASWREG   ;;APT SWITCH REGISTER
001436  000000                  $USWR:  .WORD   AUSWR    ;;USER SWITCHES
001440  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
001442     000                  $MAMS1: .BYTE   AMAMS1   ;;HIGH ADDRESS,M.S. BYTE
001443     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
001444  000000                  $MADR1: .WORD   AMADR1   ;;HIGH ADDRESS,BLK#1
                                ;*                                MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF ''TYPE'' ABOVE
001446     000                  $MAMS2: .BYTE   AMAMS2   ;;HIGH ADDRESS,M.S. BYTE
001447     000                  $MTYP2: .BYTE   AMTYP2   ;;MEM.TYPE,BLK#2
001450  000000                  $MADR2: .WORD   AMADR2   ;;MEM.LAST ADDRESS,BLK#2
001452     000                  $MAMS3: .BYTE   AMAMS3   ;;HIGH ADDRESS,M.S.BYTE
001453     000                  $MTYP3: .BYTE   AMTYP3   ;;MEM.TYPE,BLK#3
001454  000000                  $MADR3: .WORD   AMADR3   ;;MEM.LAST ADDRESS,BLK#3
001456     000                  $MAMS4: .BYTE   AMAMS4   ;;HIGH ADDRESS,M.S.BYTE
001457     000                  $MTYP4: .BYTE   AMTYP4   ;;MEM.TYPE,BLK#4
001460  000000                  $MADR4: .WORD   AMADR4   ;;MEM.LAST ADDRESS,BLK#4
001462  000300                  $VECT1: .WORD   AVECT1   ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001464  000000                  $VECT2: .WORD   AVECT2   ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001466  172410                  $BASE:  .WORD   ABASE    ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001470  000000                  $DEVM:  .WORD   ADEVM    ;;DEVICE MAP
001472  000000                  $CDW1:  .WORD   ACDW1    ;;CONTROLLER DESCRIPTION WORD#1
001474  000000                  $CDW2:  .WORD   ACDW2    ;;CONTROLLER DESCRIPTION WORD#2
001476  000000                  $DDW0:  .WORD   ADDW0    ;;DEVICE DESCRIPTOR WORD#0
001500  000000                  $DDW1:  .WORD   ADDW1    ;;DEVICE DESCRIPTOR WORD#1
001502  000000                  $DDW2:  .WORD   ADDW2    ;;DEVICE DESCRIPTOR WORD#2
001504  000000                  $DDW3:  .WORD   ADDW3    ;;DEVICE DESCRIPTOR WORD#3
001506  000000                  $DDW4:  .WORD   ADDW4    ;;DEVICE DESCRIPTOR WORD#4
001510  000000                  $DDW5:  .WORD   ADDW5    ;;DEVICE DESCRIPTOR WORD#5
001512  000000                  $DDW6:  .WORD   ADDW6    ;;DEVICE DESCRIPTOR WORD#6
001514  000000                  $DDW7:  .WORD   ADDW7    ;;DEVICE DESCRIPTOR WORD#7
001516  000000                  $DDW8:  .WORD   ADDW8    ;;DEVICE DESCRIPTOR WORD#8
001520  000000                  $DDW9:  .WORD   ADDW9    ;;DEVICE DESCRIPTOR WORD#9
```

```
001522  000000          $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
001524  000000          $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
001526  000000          $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
001530  000000          $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
001532  000000          $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
001534  000000          $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
001536                  $ETEND:
                        .MEXIT
```

```
                                .SBTTL  ERROR POINTER TABLE
                                ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
                                ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                ;*      DH              ;;POINTS TO THE DATA HEADER
                                ;*      DT              ;;POINTS TO THE DATA
                                ;*      DF              ;;POINTS TO THE DATA FORMAT
    001536                      $ERRTB:
171                             ;!ITEM 1
172 001536  043502                      .WORD   EM2     ;CANNOT ACCESS DR11 REGISTER
173 001540  047023                      .WORD   DH2     ;TEST #  ERR PC  ABRTPC  REGISTER
174 001542  051566                      .WORD   DT2     ;$TESTN,$ERRPC,OLDPC1,DREG,0
175 001544  000000                      .WORD   0       ;PRINT ALL DATA OCTAL
176
177                             ;ITEM 2
178 001546  043536                      .WORD   EM3     ;DR11-B OR W MODE INCORRECT (0=B, 1=W)
179 001550  047064                      .WORD   DH3     ;TEST #  ERR PC  EXPMOD  ACTMOD  CSRADR
180 001552  051600                      .WORD   DT3     ;$TESTN,$ERRPC,$TMP1,BORW,CSR,0
181 001554  000000                      .WORD   0       ;PRINT ALL DATA OCTAL
182
183                             ;ITEM 3
184 001556  043604                      .WORD   EM4     ;INIT FAILED TO CLEAR WCR
185 001560  047133                      .WORD   DH4     ;TEST #  ERR PC  WCRADR  WCRCONTENTS
186 001562  051614                      .WORD   DT4     ;$TESTN,$ERRPC,WCR,RWCR,0
187 001564  000000                      .WORD   0       ;PRINT ALL DATA OCTAL
188
189                             ;ITEM 4
190 001566  043635                      .WORD   EM5     ;INIT FAILED TO CLEAR BAR
191 001570  047177                      .WORD   DH5     ;TEST #  ERR PC  BARADR  BAREXP  BARRCV
192 001572  051626                      .WORD   DT5     ;$TESTN,$ERRPC,BAR,EBAR,RBAR,0
193 001574  000000                      .WORD   0       ;PRINT ALL DATA OCTAL
194
195                             ;ITEM 5
196 001576  043666                      .WORD   EM6     ;INIT FAILED TO CLEAR BDR
197 001600  047246                      .WORD   DH6     ;TEST #  ERR PC  BDRADR  BDRCONTENTS
198 001602  051642                      .WORD   DT6     ;$TESTN,$ERRPC,BDR,RBDR,0
199 001604  000000                      .WORD   0       ;PRINT ALL DATA OCTAL
200
201                             ;ITEM 6
202 001606  043717                      .WORD   EM7     ;INIT FAILED TO CLEAR ALL CSR R-W BITS
203 001610  047312                      .WORD   DH7     ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
204 001612  051654                      .WORD   DT7     ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
205 001614  000000                      .WORD   0       ;PRINT ALL DATA OCTAL
206
207                             ;ITEM 7
208 001616  043765                      .WORD   EM10    ;RESET FAILED TO CLEAR WCR
209 001620  047133                      .WORD   DH4     ;TEST #  ERR PC  WCRADR  WCRCONTENTS
210 001622  051614                      .WORD   DT4     ;$TESTN,$ERRPC,WCR,RWCR,0
211 001624  000000                      .WORD   0       ;PRINT ALL DATA OCTAL
```

```
                                ;ITEM 10
001626  044017                          .WORD   EM11    ;ATTEMPT TO SET ALL WCR BITS FAILED
001630  047133                          .WORD   DH4     ;TEST #  ERR PC  WCRADR  WCRCONTENTS
001632  051614                          .WORD   DT4     ;$TESTN,$ERRPC,WCR,RWCR,0
001634  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 11
001636  044062                          .WORD   EM12    ;RESET FAILED TO CLEAR BAR
001640  047177                          .WORD   DH5     ;TEST #  ERR PC  BARADR  BAREXP  BARRCV
001642  051626                          .WORD   DT5     ;$TESTN,$ERRPC,BAR,EBAR,RBAR,0
001644  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 12
001646  044114                          .WORD   EM13    ;ATTEMPT TO SET ALL BAR BITS TO 1 FAILED
001650  047177                          .WORD   DH5     ;TEST #  ERR PC  BARADR  BAREXP  BARRCV
001652  051626                          .WORD   DT5     ;$TESTN,$ERRPC,BAR,EBAR,RBAR,0
001654  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 13
001656  044164                          .WORD   EM14    ;CSR BIT TEST FAILED (FATAL - DIAGNOSTIC NOT CONTINUED)
001660  047366                          .WORD   DH14    ;                BIT(S)
                                                        ;TEST #  ERR PC  TESTED  CSRADR  CSREXP  CSRCONTENTS
001662  051670                          .WORD   DT14    ;$TESTN,$ERRPC,BUT,CSR,ECSR,RCSR,0
001664  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 14
001666  044253                          .WORD   EM15    ;CSR BIT TEST FAILED
001670  047366                          .WORD   DH14    ;                BIT(S)
                                                        ;TEST #  ERR PC  TESTED  CSRADR  CSREXP  CSRCONTENTS
001672  051670                          .WORD   DT14    ;$TESTN,$ERRPC,BUT,CSR,ECSR,RCSR,0
001674  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 15
001676  044277                          .WORD   EM16    ;EIR BIT TEST FAILED
001700  047501                          .WORD   DH16    ;                BIT(S)
                                                        ;TEST #  ERR PC  TESTED  EIRADR  EIREXP  EIRCONTENTS
001702  051706                          .WORD   DT16    ;$TESTN,$ERRPC,BUT,CSR,EEIR,REIR,0
001704  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 16
001706  044323                          .WORD   EM17    ;READY AND MAINTENANCE ARE NOT THE ONLY BITS SET IN CSR
001710  047614                          .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
001712  051724                          .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
001714  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 17
001716  044412                          .WORD   EM20    ;ATTN AND ERROR FAILED TO SET PROPERLY
001720  047614                          .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
001722  051724                          .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
001724  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 20
001726  044460                          .WORD   EM21    ;ATTN AND ERROR FAILED TO CLEAR PROPERLY
001730  047614                          .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
001732  051724                          .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
001734  000000                          .WORD   0       ;PRINT ALL DATA OCTAL
```

```
                                ;ITEM 21
001736  044530                          .WORD   EM22    ;ERROR BIT SHOULD HAVE BEEN CLEAR
001740  047614                          .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
001742  051724                          .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
001744  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 22
001746  044640                          .WORD   EM24    ;READY OF CSR WAS NOT SET
001750  047614                          .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
001752  051724                          .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
001754  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 23
001756  044671                          .WORD   EM25    ;BIT 0 OF THE BAR WAS SET
001760  047177                          .WORD   DH5     ;TEST #  ERR PC  BARADR  BAREXP  BARRCV
001762  051626                          .WORD   DT5     ;$TESTN,$ERRPC,BAR,EBAR,RBAR,0
001764  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 24
001766  045016                          .WORD   EM30    ;FUNCTION BIT(S) ARE NOT CLEAR
001770  047614                          .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
001772  051724                          .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
001774  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 25
001776  045054                          .WORD   EM31    ;DSTAT A, B OR C ARE NOT AS EXPECTED
002000  047614                          .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
002002  051724                          .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
002004  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 26
002006  045120                          .WORD   EM32    ;BDR IS NOT CLEAR
002010  047246                          .WORD   DH6     ;TEST #  ERR PC  BDRADR  BDRCONTENTS
002012  051642                          .WORD   DT6     ;$TESTN,$ERRPC,BDR,RBDR,0
002014  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 27
002016  045141                          .WORD   EM33    ;ALL BDR BITS ARE NOT SET
002020  047246                          .WORD   DH6     ;TEST #  ERR PC  BDRADR  BDRCONTENTS
002022  051642                          .WORD   DT6     ;$TESTN,$ERRPC,BDR,RBDR,0
002024  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 30
002026  045232                          .WORD   EM35    ;BDR SHOULD NOT HAVE BEEN LOADED WITH NEW PATTERN
002030  050020                          .WORD   DH34    ;TEST #  ERR PC  BDRADR  BDREXP  BDRCONTENTS
002032  051770                          .WORD   DT34    ;$TESTN,$ERRPC,BDR,EBDR,RBDR,0
002034  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 31
002036  045313                          .WORD   EM36    ;BDR PATTERN NOT CORRECT
002040  050020                          .WORD   DH34    ;TEST #  ERR PC  BDRADR  BDREXP  BDRCONTENTS
002042  051770                          .WORD   DT34    ;$TESTN,$ERRPC,BDR,EBDR,RBDR,0
002044  000000                          .WORD   0       ;PRINT ALL DATA OCTAL
```

```
                                ;ITEM 32
002046  045343                          .WORD   EM37    ;READY IS NOT THE ONLY BIT SET
002050  047614                          .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
002052  05172-                          .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
002054  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 33
002056  045401                          .WORD   EM40    ;READY SHOULD NOT BE SET
002060  047614                          .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
002062  051724                          .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
002064  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 34
002066  045431                          .WORD   EM41    ;READY WAS CLEARED BUT NEVER SET AGAIN
002070  047614                          .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
002072  051724                          .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
002074  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 35
002076  045533                          .WORD   EM43    ;DR11 FAILED TO INTERRUPT
002100  050074                          .WORD   DH43    ;TEST #  ERR PC  CSRADR  CSRCONTENTS
002102  052004                          .WORD   DT43    ;$TESTN,$ERRPC,CSR,RCSR,0
002104  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 36
002106  045564                          .WORD   EM44    ;DR11 INTERRUPTED, BUT IT SHOULDN'T HAVE
002110  050074                          .WORD   DH43    ;TEST #  ERR PC  CSRADR  CSRCONTENTS
002112  052004                          .WORD   DT43    ;$TESTN,$ERRPC,CSR,RCSR,0
002114  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 37
002116  045634                          .WORD   EM45    ;ERROR BIT SHOULD NOT BE CLEAR
002120  047614                          .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
002122  051724                          .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
002124  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 40
002126  045747                          .WORD   EM47    ;CSR IS WRONG
002130  047614                          .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
002132  051724                          .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
002134  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 41
002136  045764                          .WORD   EM50    ;TRANSFERS SHOULD HAVE BEEN INHIBITED
002140  050140                          .WORD   DH50    ;TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  BARADR  BAREXP  BARRCV
002142  052016                          .WORD   DT50    ;$TESTN,$ERRPC,WCR,EWCR,RWCR,BAR,EBAR,RBAR,0
002144  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 42
002146  046031                          .WORD   EM51    ;DR11 SHOULD NOT HAVE INTERRUPTED A SECOND TIME
002150  050074                          .WORD   DH43    ;TEST #  ERR PC  CSRADR  CSRCONTENTS
002152  052004                          .WORD   DT43    ;$TESTN,$ERRPC,CSR,RCSR,0
002154  000000                          .WORD   0       ;PRINT ALL DATA OCTAL
```

```
                                ;ITEM 43
002156  046110                          .WORD   EM52    ;EXPECTED INTERRUPT DID NOT OCCUR
002160  050074                          .WORD   DH43    ;TEST #  ERR PC  CSRADR  CSRCONTENTS
002162  052004                          .WORD   DT43    ;$TESTN,$ERRPC,CSR,RCSR,0
002164  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 44
002166  046177                          .WORD   EM54    ;BAR IS WRONG
002170  047744                          .WORD   DH26    ;TEST #  ERR PC  BARADR  BAREXP  BARCONTENTS
002172  051754                          .WORD   DT26    ;$TESTN,$ERRPC,BAR,EBAR,RBAR,
002174  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 45
002176  046214                          .WORD   EM55    ;BAD DATA IN BDR
002200  050020                          .WORD   DH34    ;TEST #  ERR PC  BDRADR  BDREXP  BDRCONTENTS
002202  051770                          .WORD   DT34    ;$TESTN,$ERRPC,BDR,EBDR,RBDR,
002204  000000                          .WORD   0       ;PPINT ALL DATA OCTAL

                                ;ITEM 46
002206  046272                          .WORD   EM57    ;BUFFER DATA NOT CORRECT
002210  050316                          .WORD   DH57    ;                CHECK   CHECK   INPUT   INPUT
                                                        ;TEST #  ERR PC  BUFADR  BUFDAT  BUFADR  BUFDAT  CSRADR
002212  052056                          .WORD   DT57    ;$TESTN,$ERROC,$TMP4,$TMP2,$TMP5,$TMP3,CSR,0
002214  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 47
002216  046322                          .WORD   EM60    ;TOO MANY WORDS WERE TRANSFERED
002220  050463                          .WORD   DH60    ;                DIDNOT
                                                        ;TEST #  ERR PC  EXPECT  ADRESS  CSRADR
002222  052076                          .WORD   DT60    ;$TESTN,$ERRPC,$TMP2,$TMP3,CSR,0
002224  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 50
002226  046361                          .WORD   EM61    ;UNEXPECTED TRAP OR INTERRUPT TO TRAP ADDRESS BELOW
002230  050561                          .WORD   DH61    ;TEST #  ERR PC  WCRADR  OLDPC   TRAP ADR
002232  052112                          .WORD   DT61    ;$TESTN,$ERRPC,WCR,OLDPC2,BDVECT,0
002234  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 51
002236  046444                          .WORD   EM62    ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
002240  050632                          .WORD   DH62    ;TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  CSREXP  CSRRCV  BAREXP  BAR
002242  052126                          .WORD   DT62    ;$TESTN,$ERRPC,WCR,EWCR,RWCR,ECSR,RCSR,EBAR,RBAR,0
002244  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 52
002246  046513                          .WORD   EM63    ;DR11 INTERRUPTED AT WRONG LEVEL
002250  050741                          .WORD   DH63    ;TEST #  ERR PC  EXPLVL  RCVLVL  CSRADR
002252  052152                          .WORD   DT63    ;$TESTN,$ERRPC,DRLEV,LEVEL,CSR,0
002254  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 53
002256  045533                          .WORD   EM43    ;DR11 FAILED TO INTERRUPT
002260  051010                          .WORD   DH64    ;TEST #  ERR PC  EXPLVL  CSRADR
002262  052166                          .WORD   DT64    ;$TESTN,$ERRPC,$TMP1,CSR,0
002264  000000                          .WORD   0       ;PRINT ALL DATA OCTAL
```

```
    429                                 ;ITEM 54
    430 002266  046553                          .WORD   EM65    ;2-N CYCLE BURST SWITCH IN WRONG POSITION
    431 002270  051047                          .WORD   DH65    ;TEST #  ERR PC  CSRADR  EIREXP  EIRRCV
    432 002272  052200                          .WORD   DT65    ;$TESTN,$ERRPC,CSR,EEIR,REIR,0
    433 002274  000000                          .WORD   0       ;PRINT ALL DATA OCTAL
```

```
                                ER200:                  ;THIS IS THE STARTING POINT FOR ERROR MESSAGES 201
                                                        ;THROUGH 277.  THEY ARE USED FOR MULTIPLE ERROR MESSAGES.
                                ;ITEM 201
002276  046272                          .WORD   EM57    ;BUFFER DATA NOT CORRECT
002300  050316                          .WORD   DH57    ;                CHECK   CHECK   INPUT   INPUT
                                                        ;TEST #  ERR PC  BUFADR  BUFDAT  BUFADR  BUFDAT  CSRADR
002302  052056                          .WORD   DT57    ;$TESTN,$ERRPC,$TMP4,$TMP2,$TMP5,$TMP3,CSR,0
002304  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 202
002306  046667                          .WORD   EM202   ;CSR PATTERN NOT CORRECT
002310  051165                          .WORD   DH202   ;TEST #  ERR PC  CSRADR  PATLDD  CSREXP  CSRRCV
002312  052230                          .WORD   DT202   ;$TESTN,$ERRPC,CSR,BUT,ECSR,RCSR,0
002314  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 203
002316  044722                          .WORD   EM26    ;BIT PATTERN TEST FAILED IN BAR
002320  047744                          .WORD   DH26    ;TEST #  ERR PC  BARADR  BAREXP  BARCONTENTS
002322  051754                          .WORD   DT26    ;$TESTN,$ERRPC,BAR,EBAR,RBAR,
002324  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 204
002326  044761                          .WORD   EM27    ;WCR DATA PATTERN NOT CORRECT
002330  047670                          .WORD   DH23    ;TEST #  ERR PC  WCRADR  WCREXP  WCRCONTENTS
002332  051740                          .WORD   DT23    ;$TESTN,$ERRPC,WCR,EWCR,RWCR,0
002334  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 205
002336  045313                          .WORD   EM36    ;BDR PATTERN NOT CORRECT
002340  050020                          .WORD   DH34    ;TEST #  ERR PC  BDRADR  BDREXP  BDRCONTENTS
002342  051770                          .WORD   DT34    ;$TESTN,$ERRPC,BDR,EBDR,RBDR,0
002344  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 206
002346  046151                          .WORD   EM53    ;WCR NOT EQUAL TO ZERO
002350  051404                          .WORD   DH210   ;TEST #  ERR PC  WCRADR  WCRCONTENTS
002352  052300                          .WORD   DT210   ;$TESTN,$ERRPC,WCR,RWCR,0
002354  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 207
002356  046177                          .WORD   EM54    ;BAR IS WRONG
002360  047744                          .WORD   DH26    ;TEST #  ERR PC  BARADR  BAREXP  BARCONTENTS
002362  051754                          .WORD   DT26    ;$TESTN,$ERRPC,BAR,EBAR,RBAR,
002364  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 210
002366  046234                          .WORD   EM56    ;DATA NOT TRANSFERED CORRECTLY
002370  050237                          .WORD   DH56    ;TEST #  ERR PC  NPR1AD  NPR1EX  NPR1RC  CSRADR
002372  052040                          .WORD   DT56    ;$TESTN,$ERRPC,ANPR1,ENPR1,NPR1,CSR,0
002374  000000                          .WORD   0       ;PRINT ALL DATA OCTAL

                                ;ITEM 211
002376  046717                          .WORD   EM211   ;BDR AND-OR WCR AND-OR BAR ARE INCORECT
002400  051450                          .WORD   DH211   ;TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  BDREXP  BDRRCV  BAREXP  BAR
002402  052312                          .WORD   DT211   ;$TESTN,$ERRPC,WCR,EWCR,RWCR,ECSR,RCSR,EBAR,RBAR,0
002404  000000                          .WORD   0       ;PRINT ALL DATA OCTAL
```

```
    490                                 ;ITEM 212
    491 002406  045672                          .WORD   EM46    ;FUNCTION BITS DIDN'T INCREMENT IN MAINT MODE
    492 002410  047614                          .WORD   DH17    ;TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS
    493 002412  051724                          .WORD   DT17    ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
    494 002414  000000                          .WORD   0       ;PRINT ALL DATA OCTAL
```

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

002416  000001          QTYBRD: .WORD   1       ;TOTAL # DR11 BOARDS BEING TESTED (DEFAULT = 1)

002420                  REGADR: .BLKW   16.     ;TOTAL: 16 LOCATIONS FOR BOARD ADDRESSES
002460                  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
                        ;*******************************************************************
002520  000000          WCR:    .WORD   0
002522  000000          BAR:    .WORD   0
002524  000000          CSR:    .WORD   0
002526  000000          BDR:    .WORD   0
002530  000000          DRINV:  .WORD   0
002532  000000          DRVS:   .WORD   0
002534  000000          SDRINV: .WORD   0
002536  000000          SDRVS:  .WORD   0
                        ;*******************************************************************
                        ;DO NOT INSERT ANY ITEMS BETWEEN ANY OF THE LOCATIONS ABOVE
                        ;*******************************************************************
002540  000000          BUT:    .WORD   0       ;BIT(S) UNDER TEST LOCATION
002542  000000          LEVEL:  .WORD   0       ;BR LEVEL LOCATION
002544  000000          SDVECT: .WORD   0
002546  000000          DEVMSK: .WORD   0
002550  000000          TABINX: .WORD   0
002552  000000          DREG:   .WORD   0
002554  000000          DRLEV:  .WORD   0
002556  000000          NXTTST: .WORD   0
002560  000000          PASCNT: .WORD   0

002562  000000          RCSR:   .WORD   0       ;CSR ACTUALLY READ FROM DEVICE UNDER TEST
002564  000000          REIR:   .WORD   0       ;EIR ACTUALLY READ FROM DEVICE UNDER TEST
002566  000000          RBDR:   .WORD   0       ;BDR ACTUALLY READ FROM DEVICE UNDER TEST
002570  000000          RBAR:   .WORD   0       ;BAR ACTUALLY READ FROM DEVICE UNDER TEST
002572  000000          RWCR:   .WORD   0       ;WCR ACTUALLY READ FROM DEVICE UNDER TEST

002574  000000          ECSR:   .WORD   0       ;CSR EXPECTED
002576  000000          EEIR:   .WORD   0       ;EIR EXPECTED
002600  000000          EBDR:   .WORD   0       ;BDR EXPECTED
002602  000000          EBAR:   .WORD   0       ;BAR EXPECTED
002604  000000          EWCR:   .WORD   0       ;WCR EXPECTED
002606  000000          ENPR1:  .WORD   0       ;EXPECTED OF NPR1
002610  002614          ANPR1:  .WORD   NPR1    ;ADDRESS OF NPR1
002612  000000          BORW:   .WORD   0
002614  052525          NPR1:   .WORD   52525
002616  173000          DIOMEM: .WORD   173000
002620  037164          INBUF:  .WORD   XINBUF
002622  040166          CHKBUF: .WORD   XCHKBU
002624  000000          BUFLEN: .WORD   0
```

```
552 002626  000000            LENCHK: .WORD   0
553 002630  000000            BRWAIT: .WORD   0
554 002632  000000            WCLEN:  .WORD   0
555 002634  000000            RDYCHK: .WORD   0
556 002636  177560            TKS:    .WORD   177560
557 002640  177562            TKB:    .WORD   177562
558 002642  177564            TPS:    .WORD   177564
559 002644  177566            TPB:    .WORD   177566
560 002646  000000            MSG:    .WORD   0
561 002650  000000            ADDR:   .WORD   0
562 002652  000000            MESSAG: .WORD   0
563 002654  000000            FLAG:   .WORD   0
564 002656  000000            FNCCNT: .WORD   0
565 002660  000000            INBUF1: .WORD   0
566 002662  000000            TIME:   .WORD   0       ;GENERAL PURPOSE TIMER
567 002664  000000            LOOP:   .WORD   0       ;GENERAL PURPOSE LOOP COUNTER
568 002666  000000            ANSWER: .WORD   0
569 002670  000000            BDFAIL: .WORD   0
570 002672  000000            MANSIZ: .WORD   0
571 002674  000000            OLDPC1: .WORD   0       ;LOCATION TO STORE RETURN PC IN SUBROUTINES WITH ERROR CALLS
572 002676  000000            OLDPS1: .WORD   0       ;LOCATION TO STORE PS
573 002700  000000            OLDPC2: .WORD   0       ;LOCATION TO STORE RETURN PC IN SUBROUTINES WITH ERROR CALLS
574 002702  000000            OLDPS2: .WORD   0       ;LOCATION TO STORE PS
575 002704  000000            OFL:    .WORD   0       ;FIRST CHAR FLAG
576 002706  000000            LRGSTC: .WORD   0       ;LOCATION FOR LARGEST NUMBER CHARACTER FOR THE READ SUBROUTINE
577 002710  000000            EOPLOC: .WORD   0       ;LOCATION TO HOLD FLAG DECIDING IF EOP MSGS ARE TO BE PRINTED
578 002712  000000            BITTST: .WORD   0       ;LOCATION TO PUT THE BIT STATE TO PRINT - USED BY SUBROUTINE PSTATE
579 002714  000000            MEMGMT: .WORD   0       ;LOCATION TO HOLD FLAG SAYING MEMORY MANAGEMENT IS AVAILABLE
580 002716  000000            ERRCNT: .WORD   0
```

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

```
                        .SBTTL  SUBROUTINE TO INPUT A CHARACTER OR UP TO A 6 DIGIT NUMBER
                ;*****************************************************************
                ;*
                ;*      THIS SUBROUTINE IS USED IN THE EDIT ROUTINE TO INPUT NUMBERS AND A
                ;*      SINGLE CHARACTER.  R3 IS TO BE LOADED WITH THE NUMBER OF DIGITS
                ;*      EXPECTED.  THIS SUBROUTINE WILL EXIT IF A NON-NUMERIC CHARACTER IS
                ;*      INPUTED, LEAVING THE CHARACTER IN LOCATION 'ANSWER'.  IF A NUMERIC
                ;*      CHARACTER IS INPUTED, IT WILL CLEAR ALL BUT THE 1ST 4 BITS EXPOSING
                ;*      THE VALUE OF THE DIGIT INPUTED, AND ADD IT TO R4, WHICH WAS CLEARED
                ;*      AT THE BEGINING OF THIS SUBROUTINE.  LOCATION 'LRGSTC' IS TO BE LOADED
                ;*      WITH THE LARGEST ASCII CHARACTER ACCEPTABLE FOR THIS NUMBER, I.E. 7
                ;*      OR 9 (FOR OCTAL 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  027423                  TSTB    CHARCT          ;SEE IF A CHARACTER WAS INPUTED DURING PRINTING
002730  001406                          BEQ     1$              ;BRANCH TO INPUT A CHARACTER IF NOT
002732  113737  027423  002666          MOVB    CHARCT,ANSWER   ;MOVE THE CHARACTER TO THE ANSWER LOCATION
002740  105037  027423                  CLRB    CHARCT          ;CLEAR THAT SUCKER
002744  000403                          BR      2$              ;GO CHECK IT OUT, YOU DUMMY
002746  104411                  1$:     RDCHR                   ;GET A CHARACTER
002750  012637  002666                  MOV     (SP)+,ANSWER    ;POP INPUTED CHARACTER OFF STACK
002754  022737  000003  002666  2$:     CMP     #CNTLC,ANSWER   ;SEF IF A ^C WAS INPUTED
002762  001003                          BNE     3$              ;BRANCH AROUND ITS PRINTING IF NOT
002764  104401  033607                  TYPE    ,CNTRLC         ;TYPE:  '^C'
002770  000453                          BR      6$              ;KICK OUT OF THIS ROUTINE
002772  022737  000033  002666  3$:     CMP     #ESC,ANSWER     ;SEE IF AN <ESC> WAS INPUTED
003000  001003                          BNE     4$              ;BRANCH AROUND ITS PRINTING IF NOT
003002  104401  033476                  TYPE    ,ESCAPE         ;TYPE:  '<ESC>'
003006  000444                          BR      6$              ;KICK OUT OF THIS ROUTINE
003010  113737  002666  033605  4$:     MOVB    ANSWER,LETNCR   ;MOVE CHARACTER FOR PRINTING
003016  104401  033605                  TYPE    ,LETNCR         ;GO TYPE THE INPUTED CHARACTER
003022  022737  000057  002666          CMP     #'/,ANSWER      ;SEE IF A NON-NUMERIC/ALPH CHARACTER WAS INPUTED
003030  100016                          BPL     5$              ;BRANCH TO R4 TEST IF SO
003032  123737  002706  002666          CMPB    LRGSTC,ANSWER   ;SEE IF A NON-OCTAL/NUMERIC CHARACTER WAS INPUTED
003040  100427                          BMI     6$              ;BRANCH TO EXIT IF SO
003042  013746  002666                  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  002706  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  002666          CMP     #CARETN,ANSWER  ;SEE IF CARRIAGE RETURN WAS INPUTED
003112  001402                          BEQ     6$              ;BRANCH IF SO - R4 IS CORRECT
003114  062704  000002                  ADD     #2,R4           ;ADD 2 TO R4 TO MAKE OCTAL NUMBER THE DECIMAL EQUIVALENT
003120  104401  001405          6$:     TYPE    ,$CRLF          ;PRINT A <CRLF>
003124  000207                          RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO CAPTURE UNIT #, PASS # & TOTAL ERRORS
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE IS CALLED BY $EOP AND ENDEV TO SAVE THE UNIT NUMBER,
                                ;*      PASS NUMBER, AND TOTAL ERRORS FOR THAT DEVICE/PASS FOR SAVING WHENEVER
                                ;*      A DEVICE PASS CONTAINS ERRORS.
                                ;*
                                ;*******************************************************************
003126  022737  043452  041170  ERCAPT: CMP     #ENDSTK,CAPNTR  ;SEE IF STACK IS FULL OF BULL
003134  001414                          BEQ     1$              ;KICK OUT IF FULL
003136  013700  041170                  MOV     CAPNTR,R0       ;MOVE CAPNTR CONTENTS TO R0
003142  013720  001424                  MOV     $UNIT,(R0)+     ;PUT UNIT NUMBER ON STACK
003146  013720  001420                  MOV     $PASS+2,(R0)+   ;PUT OVERFLOW PASS NUMBER ON STACK
003152  013720  001416                  MOV     $PASS,(R0)+     ;PUT PASS NUMBER ON STACK
003156  013720  001312                  MOV     $ERTTL,(R0)+    ;PUT TOTAL ERRORS ON STACK
003162  010037  041170                  MOV     R0,CAPNTR       ;RESTORE CAPNTR TO NEW POINTER VALUE
003166  000207                  1$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO PRINT THE DATA STORED BY SUBROU.INE ERCAPT
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE IS INVOKED BY ENTERING (^Y) DURING THE EXECUTION OF THE
                                ;*      TEST.  IT PRINTS ALL DATA STORED IN THE 'CAPSTK' STACK STORED BY THE
                                ;*      'ERCAPT' SUBROUTINE, REINITIALIZES THE POINTER, AND RETURNS.  *DATA*
                                ;*      *PRINTED*IS*LOST* BECAUSE OF THE REINITIALIZATION OF THE COUNTER.
                                ;*
                                ;*******************************************************************
003170  022737  041172  041170  P^CAPT: CMP     #CAPSTK,CAPNTR  ;SEE IF THERE IS ANY DATA TO PRINT
003176  001003                          BNE     1$              ;BRANCH TO PRINT IT IF THERE IS
003200  104401  032725                  TYPE    ,NODATA         ;TYPE:  'NO ERROR TOTALS TO REPORT'
003204  000462                          BR      7$              ;KICK OUT
003206  022737  043452  041170  1$:     CMP     #ENDSTK,CAPNTR  ;SEE IF STACK IS FULL OF BULL
003214  001002                          BNE     2$              ;BRANCH AROUND STACK IS FULL MESSAGE IF NOT
003216  104401  032534                  TYPE    ,STKIFL         ;TYPE:  'STACK IS FULL - DATA MAY HAVE BEEN LOST'
003222  104401  032607          2$:     TYPE    ,ERCHDR         ;TYPE:  'SUMMATION OF ERRORS SINCE BEGINNING OR LAST REPORT'
                                                                ;       'BOARD #   PASS #   ERRTTL'
003226  012700  041172                  MOV     #CAPSTK,R0      ;MOVE ADDRESS OF STACK TO PRINT TO R0
003232  012701  000020                  MOV     #16.,R1         ;MOVE 16 BOARDS TO SEARCH TO R0
003236  005037  001362                  CLR     $TMP1           ;CLEAR $TMP1, DEVICE POINTER
003242  021037  001362          3$:     CMP     (R0),$TMP1      ;SEE IF UNIT NUMBER IS TO PRINT
003246  001403                          BEQ     4$              ;BRANCH TO PRINT DATA IF SO
003250  062700  000010                  ADD     #10,R0          ;GO TO NEXT SET OF DATA AND
003254  000420                          BR      6$              ;GO SEE IF ANY MORE TO CHECK
003256  012046                  4$:     MOV     (R0)+,-(SP)     ;MOVE UNIT NUMBER TO STACK FOR PRINTING
003260  104405                          TYPDS                   ;GO TYPE UNIT NUMBER IN DECIMAL
003262  104401  033764                  TYPE    ,SPACES         ;TYPE 2 SPACES
003266  012046                          MOV     (R0)+,-(SP)     ;MOVE OVERFLOW PASS NUMBER TO STACK FOR PRINTING
003270  001002                          BNE     5$              ;BRANCH IF NON-ZERO
003272  104401  033764                  TYPE    ,SPACES         ;TYPE AN EXTRA 2 SPACES - NUMBER WILL NOT BE EXTENDED
003276  012046                  5$:     MOV     (R0)+,-(SP)     ;MOVE PASS NUMBER TO STACK FOR PRINTING
003300  104406                          TYPDE                   ;GO TYPE PASS NUMBER IN EXTENDED DECIMAL
003302  104401  033764                  TYPE    ,SPACES         ;TYPE 2 SPACES
003306  012046                          MOV     (R0)+,-(SP)     ;MOVE ERROR TOTAL TO STACK FOR PRINTING
003310  104405                          TYPDS                   ;GO TYPE ERROR TOTAL IN DECIMAL
003312  104401  001405                  TYPE    ,$CRLF          ;TYPE <CRLF>
003316  020037  041170          6$:     CMP     R0,CAPNTR       ;SEE IF ALL DATA HAS BEEN PRINTED
003322  001347                          BNE     3$              ;BRANCH BACK IF NOT
003324  005237  001362                  INC     $TMP1           ;INCREMENT DEVICE POINTER
003330  012700  041172                  MOV     #CAPSTK,R0      ;INITIALIZE TO BEGINNING FOR ANOTHER POSSIBLE PASS
003334  005301                          DEC     R1              ;DECREMENT THE LOOP COUNTER AND
003336  001341                          BNE     3$              ;BRANCH UNTIL ALL DEVICE DATA PRINTED
003340  012737  041172  041170          MOV     #CAPSTK,CAPNTR  ;REINITIALIZE CAPNTR
003346  104401  001405                  TYPE    ,$CRLF          ;TYPE ANOTHER <CRLF>
003352  105037  027423          7$:     CLRB    CHARCT          ;CLEAR ANY CHARACTER ENTERED DURING PRINTING
003356  000207                          RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO FILL ALL TABLE BOARD ENTRIES
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE FILLS ALL TABLE BOARD ENTRIES FOR THE ADDRESSES AND
                                ;*      VECTORS FROM THE VALUES IN 'REGADR' AND 'VECADR', AND SHOULD BE SET
                                ;*      UP BEFORE ENTERING THIS SUBROUTINE.
                                ;*
                                ;*******************************************************************
003360  005037  001470          FIXTBL: CLR     $DEVM           ;CLEAR THE DEVICE MASK
003364  000261                          SEC                     ;SET THE CARRY BIT AND
003366  006137  001470                  ROL     $DEVM           ;ROTATE IT INTO $DEVM FOR 1 BOARD
003372  022737  000001  002416          CMP     #1,QTYBRD       ;SEE IF ONLY 1 BOARD PRESENT
003400  001433                          BEQ     2$              ;KICK OUT IF SO - TABLE DOESN'T NEED FILLING
003402  013701  002416                  MOV     QTYBRD,R1       ;FILL ALL TABLE BOARD ENTRIES FROM FIRST
003406  005301                          DEC     R1              ;DECREMENT SINCE 1ST POSITION IS ALREADY FILLED
003410  005002                          CLR     R2              ;CLEAR INDEX TO SEND POINTER
003412  012703  000002                  MOV     #2,R3           ;PUT 2 IN INDEX TO RECEIVE POINTER
003416  000261                  1$:     SEC                     ;SET THE CARRY BIT AND
003420  006137  001470                  ROL     $DEVM           ;ROTATE IT INTO $DEVM
        002420                  RA=REGADR               ;REDEFINE REGADR AS RA FOR SPACE REASONS
003424  016263  002420  002420          MOV     RA(R2),RA(R3)   ;TRANSFER ADDRESS TO NEXT POSITION AND
003432  062763  000010  002420          ADD     #10,RA(R3)      ;ADD 10 FOR NEXT POSITION
        002460                  VA=VECADR               ;REDEFINE VECADR AS VA FOR SPACE REASONS
003440  016263  002460  002460          MOV     VA(R2),VA(R3)   ;TRANSFER VECTOR TO NEXT POSITION AND
003446  062763  000010  002460          ADD     #10,VA(R3)      ;ADD 10 FOR NEXT POSITION
003454  013763  001476  001476          MOV     $DDW0,$DDW0(R3) ;MOVE DEVICE DESCRIPTOR WORD TO NEXT POSITION
003462  022223                          CMP     (R2)+,(R3)+     ;UPDATE INDEX POINTERS
003464  005301                          DEC     R1              ;DECREMENT THE LOOP COUNTER
003466  001353                          BNE     1$              ;BRANCH BACK IF NOT DONE
003470  000207                  2$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO LOAD INBUF WITH AN INCREMENTING PATTERN
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE CLEARS THE FIRST LOCATION OF THE BUFFER AND LOADS
                                ;*      NUMBERS STARTING WITH 1 INTO THE BUFFER.
                                ;*
                                ;*******************************************************************
003472  013702  002620          LODBUF: MOV     INBUF,R2        ;MOVE STARTING ADDRESS OF INBUF TO R2
003476  013703  002624                  MOV     BUFLEN,R3       ;MOVE LOOP COUNTER TO R3
003502  005203                          INC     R3              ;CORRECT COUNTER
003504  005001                          CLR     R1              ;CLEAR THE LOADING COUNTER
003506  010122                  1$:     MOV     R1,(R2)+        ;LOAD NEXT BUFFER WORD
003510  005201                          INC     R1              ;INCREMENT THE LOADING COUNTER
003512  005303                          DEC     R3              ;DECREMENT THE LOOP COUNTER AND
003514  001374                          BNE     1$              ;BRANCH BACK IF NOT DONE
003516  000207                          RTS     PC              ;EXIT
```

```
                                        .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.
                                ;*
                                ;*******************************************************************
003520  013702  002622          CHKBFF: MOV     CHKBUF,R2       ;STARTING ADDRESS OF CHECK-BUFFER TO R2
003524  013701  002624                  MOV     BUFLEN,R1       ;MOVE LOOP COUNTER TO R1
003530  005003                          CLR     R3              ;WIPE OUT R3
003532  010322                  1$:     MOV     R3,(R2)+        ;MOVE R3 TO CHKBUF ADDRESS AND INC BY 2
003534  010322                          MOV     R3,(R2)+        ;MOVE R3 TO NEXT CHKBUF ADDRESS AND INC BY 2
003536  022341                          CMP     (R3)+,-(R1)     ;ADD 2 TO NUMBER FOR BUFFER & SUBTRACT 2 FROM LOOP COUNTER
003540  005701                          TST     R1              ;SEE IF R1 HAS REACHED ZERO YET
003542  001373                          BNE     1$              ;BRANCH BACK IF NOT DONE
003544  000207                  2$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO CLEAR IE, CHECK ERROR, READY, WCR=0, AND BAR
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE HAS THE NEED TO CALL AN ERROR IN THE TEST.  THE ERROR
                                ;*      IS TO BE LOCATED IN THE TEST JUST AFTER THE JSR CALL.  FUTURE USE OF
                                ;*      THIS SUBROUTINE MUST BE HANDLED AS FOLLOWS:
                                ;*
                                ;*              JSR     PC,INTA         ;SUBROUTINE CALL
                                ;*              ERROR   +51             ;ERROR CALL
                                ;*              CONTINUE                ;SUBROUTINE RETURNS HERE IF NO ERROR
                                ;*
                                ;*******************************************************************
003546  042777  000100  176750  INTA:   BIC     #IE,@CSR        ;CLEAR IE
003554  013702  002624          MOV     BUFLEN,R2       ;BUFFER LENGTH TO R2
003560  063702  002624          ADD     BUFLEN,R2       ;NUMBER OF XFERS TIMES 2
003564  063702  002620          ADD     INBUF,R2        ;CORRECT BAR
003570  017737  176730  002562  MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
003576  032737  010000  002562  BIT     #MA,RCSR        ;SEE IF WE ARE IN MAINTENANCE MODE
003604  001005                  BNE     1$              ;BRANCH AROUND CABLE TEST IF WE ARE, BIT 0 WILL BE CLEAR
003606  032737  000004  002720  BIT     #BIT2,DDW       ;SEE IF THERE IS A CABLE
003614  001401                  BEQ     1$              ;BRANCH IF NO CABLE
003616  005202                  INC     R2              ;CABLE MODE TESTING LEAVES BIT 0 OF BAR SET. CHECK ODD ADRS
003620  017737  176676  002570  1$:     MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
003626  005037  002604          CLR     EWCR            ;CLEAR EXPECTED LOCATION
003632  010237  002602          MOV     R2,EBAR         ;MOVE EXPECTED DATA TO EBAR
003636  013737  002562  002574  MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
003644  042737  100000  002574  BIC     #ER,ECSR        ;MAKE SURE ERROR BIT IS CLEAR
003652  052737  000200  002574  BIS     #RY,ECSR        ;MAKE SURE READY BIT IS SET
003660  017737  176634  002572  MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
003666  001012                  BNE     2$              ;BRANCH TO RETURN WITHOUT UPDATING PC SO ERROR WILL CALL
003670  023737  002562  002574  CMP     RCSR,ECSR       ;DOES CSR CONTAIN WHAT IT SHOULD
003676  001006                  BNE     2$              ;BRANCH TO RETURN WITHOUT UPDATING PC SO ERROR WILL CALL
003700  023737  002570  002602  CMP     RBAR,EBAR       ;DOES BAR CONTAIN WHAT IT SHOULD
003706  001002                  BNE     2$              ;BRANCH TO RETURN WITHOUT UPDATING PC SO ERROR WILL CALL
003710  062716  000002          ADD     #2,(SP)         ;CORRECT PC RETURN TO AFTER THE ERROR CALL
003714  000207          2$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO CHECK INBUF AFTER A MAINTENANCE MODE OPERATION
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE HAS THE NEED TO CALL AN ERROR IN THE TEST AND RETURN
                                ;*      TO THE SUBROUTINE.  THE ERROR AND RETURN JSR ARE TO BE LOCATED IN THE
                                ;*      TEST JUST AFTER THE JSR CALL.  YOU *MUST* CLEAR LOCATION 'ERRCNT'
                                ;*      BEFORE EXECUTION OF THIS SUBROUTINE, OTHERWISE YOU MAY NOT GET ANY
                                ;*      ERRORS PRINTED, OR IF SO, JUST THE DATA WITHOUT THE HEADER. FUTURE
                                ;*      USE OF THIS SUBROUTINE MUST BE HANDLED AS FOLLOWS:
                                ;*
                                ;*      JSR     PC,DATCHK       ;SUBROUTINE CALL
                                ;*      ERROR   +201            ;ERROR CALL
                                ;*      JSR     PC,DATCH2       ;RETURN TO SUBROUTINE AFTER ERROR RTS
                                ;*      CONTINUE                ;SUBROUTINE RETURNS HERE IF NO ERROR(S) OR WHEN DONE
                                ;*
                                ;*******************************************************************
003716  011637  001362          DATCHK: MOV     (SP),$TMP1      ;SAVE PC RETURN
003722  013702  002622                  MOV     CHKBUF,R2       ;STARTING ADDRESS OF CHECK BUFFER TO R2
003726  013703  002620                  MOV     INBUF,R3        ;STARTING ADDRESS OF IN BUFFER TO R3
003732  005037  002626                  CLR     LENCHK          ;CLEAR LENGTH CHECK
003736  005237  002626          DATCH1: INC     LENCHK          ;MAKE A COMPARISON
003742  022223                          CMP     (R2)+,(R3)+     ;IS THE DATA CORRECT?
003744  001423                          BEQ     DATCH2          ;BRANCH IF OK
003746  013716  001362                  MOV     $TMP1,(SP)      ;RESTORE ORIGINAL PC RETURN
003752  016237  177776  001364          MOV     -2(R2),$TMP2    ;MOVE CHECK BUFFER CONTENTS TO $TMP2
003760  010237  001370                  MOV     R2,$TMP4        ;MOVE ADDRESS +2 TO $TMP4
003764  162737  000002  001370          SUB     #2,$TMP4        ;CORRECT SO IT POINTS TO ADDRESS CAUSING ERROR
003772  016337  177776  001366          MOV     -2(R3),$TMP3    ;MOVE INPUT BUFFER CONTENTS TO $TMP3
004000  010337  001372                  MOV     R3,$TMP5        ;MOVE ADDRESS +2 TO $TMP5
004004  162737  000002  001372          SUB     #2,$TMP5        ;CORRECT SO IT POINTS TO ADDRESS CAUSING ERROR
004012  000207                          RTS     PC              ;RETURN TO ERROR CALL - PC ON STACK ALREADY POINTS THERE
004014  023737  002626  002624  DATCH2: CMP     LENCHK,BUFLEN   ;SEE IF THE BUFFER HAS BEEN CHECKED
004022  001345                          BNE     DATCH1          ;GO BACK FOR ANOTHER TRY IF NOT
004024  013716  001362                  MOV     $TMP1,(SP)      ;RESTORE PC RETURN
004030  062716  000006                  ADD     #6,(SP)         ;CORRECT IT SO RETURN IS AFTER THE ERROR CALL
004034  000207                          RTS     PC              ;RETURN
```

```
                                        .SBTTL  SUBROUTINE TO RESTORE DR11 INT VECT & SET CPU PRIORITY TO 7.
                                ;***************************************************************
                                ;*
                                ;*      THIS ROUTINE IS USED IN VARIOUS TESTS TO CLEAR ANY DATA THAT
                                ;*      MAY BE LEFT IN ANY REGISTERS, AND RESTORE CPU PRIORITY TO 7.
                                ;*
                                ;***************************************************************
004036  012737  000340  177776  CLENUP: MOV     #LEVEL7,PSW     ;RESTORE CPU TO PRIORITY LEVEL 7
004044  012777  010000  176452          MOV     #MA,@CSR        ;DO AN INIT CLEARING WCR, BAR & BDR BY SETTING
004052  005077  176446                  CLR     @CSR            ;AND CLEARING THE MAINT BIT AND CLEAR THE CSR
004056  000207                          RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO CHECK FOR CABLE MODE AND ALTER EXPECTED DATA
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE CHECKS THE DDW (DEVICE DESCRIPTOR WORD) FOR THE CABLE
                                ;*      BEING IN OR OUT AND SETS BITS 12, 10, 8 AND 6 IN THE EXPECTED DATA
                                ;*      LOCATION IF THE CABLE IS OUT.
                                ;*
                                ;*******************************************************************
004060  032737  000004  002720  CHKCAB: BIT     #BIT2,DDW       ;CHECK FOR CABLE STATUS
004066  001006                          BNE     1$              ;EXIT IF CABLE DOES EXIST
004070  052737  127000  002574          BIS     #127000,ECSR    ;SET BITS 12, 10, 8, 7 & 6 - CABLE DOESN'T EXIST
004076  052737  127000  002540          BIS     #127000,BUT     ;SET BITS 12, 10, 8, 7 & 6 IN BITS UNDER TEST LOCATION TOO
004104  000207                  1$:     RTS     PC              ;RETURN
```

```
                                        .SBTTL  SUBROUTINE TO CHECK CORRECT DATA PATTERN WAS MOVED TO INBUF
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE HAS THE NEED TO CALL 2 ERRORS.  THE ERRORS ARE TO BE
                                ;*      LOCATED IN THE TEST JUST AFTER THE JSR CALL.  FUTURE USE OF THIS
                                ;*      SUBROUTINE MUST BE HANDLED AS FOLLOWS:
                                ;*
                                ;*      JSR     PC,DATOCK       ;SUBROUTINE CALL
                                ;*      ERROR   +46             ;ERROR CALL
                                ;*      JSR     PC,DATOC2       ;RETURN TO SUBROUTINE AFTER ERROR RTS
                                ;*      ERROR   +47             ;2ND ERROR CALL
                                ;*      CONTINUE                ;SUBROUTINE RETURNS HERE WHEN  .NE
                                ;*
                                ;*******************************************************************
004106  011637  001362          DATOCK: MOV     (SP),$TMP1      ;SAVE PC RETURN
004112  012702  052525                  MOV     #52525,R2       ;DATO NUMBER TO R2
004116  013703  002620                  MOV     INBUF,R3        ;STARTING ADDRESS OF IN BUFFER TO R3
004122  005037  002626                  CLR     LENCHK          ;CLEAR LENGTH CHECK
004126  005237  002626          DATOC1: INC     LENCHK          ;MAKE A COMPARISON
004132  020223                          CMP     R2,(R3)+        ;IS THE DATA CORRECT?
004134  001415                          BEQ     DATOC2          ;BRANCH IF OK
004136  013716  001362                  MOV     $TMP1,(SP)      ;MOVE OLD PC RETURN TO STACK
004142  010237  001364                  MOV     R2,$TMP2        ;MOVE EXPECTED DATA TO $TMP2
004146  016337  177776  001366          MOV     -2(R3),$TMP3    ;MOVE RECEIVED DATA TO $TMP3
004154  010337  001370                  MOV     R3,$TMP4        ;MOVE ADDRESS +2 TO $TMP4
004160  162737  000002  001370          SUB     #2,$TMP4        ;CORRECT ADDRESS SO IT POINTS TO ADDRESS CAUSING ERROR
004166  000207                          RTS     PC              ;RETURN TO ERROR CALL
004170  023737  002626  002624  DATOC2: CMP     LENCHK,BUFLEN   ;SEE IF THE BUFFER HAS BEEN CHECKED
004176  001353                          BNE     DATOC1          ;BUFFER CHECKED?
004200  020223                          CMP     R2,(R3)+        ;CHECK END OF BUFFER + 1
004202  001017                          BNE     1$              ;BRANCH IF NOT LOADED
004204  010237  001364                  MOV     R2,$TMP2        ;MOVE EXPECTED DATA TO $TMP2
004210  016337  177776  001366          MOV     -2(R3),$TMP3    ;MOVE RECEIVED DATA TO $TMP3
004216  010337  001370                  MOV     R3,$TMP4        ;MOVE ADDRESS +2 TO $TMP4
004222  162737  000002  001370          SUB     #2,$TMP4        ;CORRECT ADDRESS SO IT POINTS TO ADDRESS CAUSING ERROR
004230  013716  001362                  MOV     $TMP1,(SP)      ;CORRECT PC RETURN
004234  062716  000006                  ADD     #6,(SP)         ;POINT TO 2ND ERROR CALL AFTER JSR PC,DATOCK
004240  000207                          RTS     PC              ;RETURN TO ERROR CALL
004242  013716  001362          1$:     MOV     $TMP1,(SP)      ;RESTORE RETURN ADDRESS
004246  062716  000010                  ADD     #10,(SP)        ;POINT TO PROPER RETURN AFTER THE ERROR CALLS
004252  000207                          RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO CLEAR IE AND HALT IF ERROR IS SET
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE HAS THE NEED TO CALL AN ERROR IN THE TEST.  THE ERROR
                                ;*      IS TO BE LOCATED IN THE TEST JUST AFTER THE JSR CALL.  FUTURE USE OF
                                ;*      THIS SUBROUTINE MUST BE HANDLED AS FOLLOWS:
                                ;*      JSR     PC,ERRCHK       ;SUBROUTINE CALL
                                ;*      ERROR   +21             ;ERROR CALL
                                ;*      CONTINUE                ;SUBROUTINE RETURNS HERE IF NO ERROR
                                ;*
                                ;*******************************************************************
004254  042777  000100  176242  ERRCHK: BIC     #IE,@CSR        ;CLEAR IE
004262  017737  176236  002562          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
004270  013737  002562  002574          MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
004276  042737  100000  002574          BIC     #ER,ECSR        ;CLEAR THE ERROR BIT
004304  052737  000200  002574          BIS     #RY,ECSR        ;SET THE READY BIT
004312  013701  002562                  MOV     RCSR,R1         ;MOVE DATA TO R1 FOR CHECKING
004316  012737  000200  001360          MOV     #RY,$TMP0       ;MOVE EXPECTED DATA TO $TMP0
004324  042701  077577                  BIC     #77577,R1       ;CLEAR ALL BUT THE ERROR AND READY BITS
004330  022701  000200                  CMP     #RY,R1          ;SEE IF ERROR BIT IS CLEAR AND READY IS SET
004334  001002                          BNE     1$              ;BRANCH AROUND PC CORRECTION SO ERROR WILL CALL
004336  062716  000002                  ADD     #2,(SP)         ;CORRECT PC RETURN - DATA OK
004342  000207                  1$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO GENERATE DEVICE ADDRESS TABLE
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE GENERATES AN ADDRESS TABLE LOCATED AT REGADR STARTING
                                ;*      WITH THE BASE DEVICE ADDRESS (CONTENTS OF $BASE) IN STEPS OF 10.
                                ;*
                                ;*******************************************************************
004344  012702  002420          DEVADS: MOV     #REGADR,R2      ;POINT R2 TO THE DEVICE ADDRESS TABLE
004350  013700  001466                  MOV     $BASE,R0        ;LOAD BASE DEVICE ADDRESS IN R0
004354  012701  000020                  MOV     #16.,R1         ;MOVE LOOP COUNTER TO R1
004360  010022                  1$:     MOV     R0,(R2)+        ;MOVE DEVICE ADDRESS TO TABLE
004362  062700  000010                  ADD     #10,R0          ;POINT R0 TO NEXT DEVICE ADDRESS
004366  005301                          DEC     R1              ;DECREMENT THE LOOP COUNTER AND
004370  001373                          BNE     1$              ;BRANCH IF NOT ALL DONE YET
004372  000207                          RTS     PC              ;EXIT
```

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

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

```
                                        .SBTTL  SUBROUTINE TO EXTRACT INFORMATION ABOUT THE DR11
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE EXTRACTS INFORMATION ABOUT THE DR11 THAT INTERRUPTED
                                ;*      AND LOADS THE DATA FOUND INTO THE DEVICE DESCRIPTOR WORD FOR THAT
                                ;*      BOARD.
                                ;*
                                ;*******************************************************************
004452  012710  010000        DRGET:  MOV     #MA,(R0)        ;SET THE MAINTENANCE BIT AND
004456  005010                        CLR     (R0)            ;CLEAR THE CSR TO DO AN INIT
004460  016612  000002                MOV     2(SP),(R2)      ;MOVE VECTOR+4 FOR THIS MODULE TO VECTOR LOCATION
004464  012616                        MOV     (SP)+,(SP)      ;MOVE RETURN OF THIS SUBROUTINE TO ITS PROPER POSITION
004466  162712  000004                SUB     #4,(R2)         ;DUMB VECTOR IS WRONG - CORRECT IT
004472  013711  002542                MOV     LEVEL,(R1)      ;PUT THE PRIORITY LEVEL INTO THE $DDWXX LOCATION
004476  005237  001470                INC     $DEVM           ;INDICATE DEVICE EXISTENCE IN DEVICE MAP
004502  005237  002416                INC     QTYBRD          ;INCREMENT DEVICE COUNT
004506  052710  100000                BIS     #EIR,(R0)       ;GO TO EIR TO GET B/W STATE
004512  011037  002564                MOV     (R0),REIR       ;MOVE EIR TO REIR
004516  005010                        CLR     (R0)            ;GO BACK TO CSR
004520  012710  010000                MOV     #MA,(R0)        ;SET THE MAINT BIT
004524  005010                        CLR     (R0)            ;DO AN INIT
004526  032737  000001  002564        BIT     #BIT0,REIR      ;TEST FOR B/W STATE
004534  001003                        BNE     1$              ;BRANCH IF A W
004536  052711  000001                BIS     #BIT0,(R1)      ;SET STATE IN DEVICE DESCRIPTOR WORD
004542  000406                        BR      2$              ;GO TO CABLE STATUS TEST
004544  032737  000400  002564  1$:   BIT     #N2,REIR        ;CHECK 2/N CYCLE STATE
004552  001402                        BEQ     2$              ;BRANCH IF 2 CYCLE
004554  052711  000002                BIS     #BIT1,(R1)      ;N CYCLE - SET BIT IN DEVICE DESC
004560  011037  002562          2$:   MOV     (R0),RCSR       ;MOVE RECEIVED DATA TO RCSR TO GET CABLE STATUS
004564  032737  127000  002562        BIT     #127000,RCSR    ;CHECK IF ANY BITS ARE SET - THEY ARE IF NO CABLE
004572  001015                        BNE     3$              ;BRANCH IF NO CABLE
004574  112710  000004                MOVB    #F2,(R0)        ;CABLE IS POSSIBLY IN - SET FNCT2
004600  011037  002562                MOV     (R0),RCSR       ;MOVE RECEIVED DATA TO RCSR
004604  052710  010000                BIS     #MA,(R0)        ;SET THE MAINTENANCE BIT
004610  005010                        CLR     (R0)            ;CLEAR THE CSR TO DO AN INIT
004612  032737  020000  002562        BIT     #AT,RCSR        ;TEST THE ATTN BIT
004620  001402                        BEQ     3$              ;BRANCH IF NOT SET - NO CABLE
004622  052711  000004                BIS     #BIT2,(R1)      ;SET CABLE BIT IN DEVICE DESC
004626  000207                3$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO PRINT THE AUTOSIZED BOARD CONFIGURATIONS
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE PRINTS THE BOARD CONFIGURATIONS FOUND BY ASIZE
                                ;*
                                ;*******************************************************************
004630  005037  002664          TYPCNF: CLR     LOOP            ;CLEAR THE BOARD NUMBER COUNTER
004634  013705  001470                  MOV     $DEVM,R5        ;GET DEVICE MAP
004640  012701  002420                  MOV     #REGADR,R1      ;MOVE THE ADDRESS OF THE REGISTER ADDRESS TABLE TO R1
004644  012702  002460                  MOV     #VECADR,R2      ;MOVE THE ADDRESS OF THE VECTOR ADDRESS TABLE TO R2
004650  005003                          CLR     R3              ;CLEAR THE DEVICE DESCRIPTOR ADDRESS POINTER
004652  104401  034147                  TYPE    ,NOBUT          ;TYPE:  'NO. BOARDS UNDER TEST:  '
004656  013746  002416                  MOV     QTYBRD,-(SP)    ;MOVE THE QUANTITY TO THE STACK FOR TYPEOUT
004662  104405                          TYPDS                   ;TYPE THE NUMBER
004664  104401  034206                  TYPE    ,BRVWPC         ;TYPE THE HEADER
004670  012700  000020                  MOV     #16.,R0         ;SET UP LOOP COUNTER FOR 16 BOARDS
004674  032705  000001          1$:     BIT     #BIT0,R5        ;DEVICE UNDER TEST?
004700  001466                          BEQ     8$              ;BRANCH IF NO
004702  013746  002664                  MOV     LOOP,-(SP)      ;PUT BOARD # ON STACK FOR TYPEOUT
004706  104405                          TYPDS                   ;PRINT BOARD # (0 TO 16)
004710  104401  033773                  TYPE    ,SPACE6         ;TYPE 6 SPACE CHARACTERS
004714  011146                          MOV     (R1),-(SP)      ;SAVE REGISTER ADDRESS FOR TYPEOUT
004716  104402                          TYPOC                   ;PRINT DEVICE REGISTER ADDRESS
004720  104401  033773                  TYPE    ,SPACE6         ;TYPE 6 SPACE CHARACTERS
004724  011246                          MOV     (R2),-(SP)      ;SAVE VECTOR ADDRESS FOR TYPEOUT
004726  104403                          TYPOS                   ;PRINT VECTOR ADDRESS
004730     003     000                  .BYTE   3,0             ;PRINT 3 DIGITS, LEADING ZEROS SUPPRESSED
004732  104401  034002                  TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
004736  016337  001476  002720          MOV     $DDW0(R3),DDW   ;MOVE DEVICE DESCRIPTOR WORD TO DDW
004744  032737  000001  002720          BIT     #BIT0,DDW       ;TEST WHICH STATE, B OR W, FOR THIS BOARD
004752  001403                          BEQ     2$              ;GO PRINT W STATE IF W
004754  104401  034017                  TYPE    ,B              ;TYPE A 'B'
004760  000402                          BR      3$              ;GO TO NEXT CHECK
004762  104401  034021          2$:     TYPE    ,W              ;'TYPE A 'W'
004766  104401  034002          3$:     TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
004772  004737  005630                  JSR     PC,PNTPRI       ;PRINT DEVICE PRIORITY
004776  104401  034002                  TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
005002  032737  000002  002720          BIT     #BIT1,DDW       ;TEST 2/N CYCLE STATE
005010  001403                          BEQ     4$              ;GO PRINT 2 STATE IF 2
005012  104401  034034                  TYPE    ,N              ;TYPE AN 'N'
005016  000402                          BR      5$              ;GO TO NEXT CHECK
005020  104401  034032          4$:     TYPE    ,TWO            ;TYPE A '2'
005024  104401  034002          5$:     TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
005030  032737  000004  002720          BIT     #BIT2,DDW       ;TEST CABLE STATE
005036  001403                          BEQ     6$              ;GO PRINT 'NO' IF NO CABLE
005040  104401  034026                  TYPE    ,YES            ;TYPE 'YES'
005044  000402                          BR      7$              ;GO TO LOOP CHECK
005046  104401  034023          6$:     TYPE    ,NO             ;TYPE 'NO'
005052  104401  001405          7$:     TYPE    ,$CRLF          ;TYPE A <CRLF>
005056  005237  002664          8$:     INC     LOOP            ;INCREMENT BOARD COUNT FOR POSSIBLE NEXT PASS
005062  022122                          CMP     (R1)+,(R2)+     ;INCREMENT COUNTERS
005064  006205                          ASR     R5              ;SHIFT R5 TO THE RIGHT TO MOVE BOARD BIT INTO BIT 0
005066  062703  000002                  ADD     #2,R3           ;ADD 2 TO THE DEVICE DESCRIPTOR WORD POINTER
005072  005300                          DEC     R0              ;DECREMENT THE LOOP COUNTER AND
005074  001277                          BNE     1$              ;BRANCH BACK FOR CHECK IF 16 BOARDS NOT DONE
005076  104401  001405                  TYPE    ,$CRLF          ;TYPE ANOTHER <CRLF>
005102  000207                          RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO CHECK FOR LOCATION BELONGING TO A DR11
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE CHECKS FOR THE LOCATION BELONGING TO A DR11.
                                ;*
                                ;*******************************************************************
005104  012737  000340  002542  CHK4DR: MOV     #LEVEL7,LEVEL   ;MOVE PRIORITY 7 TO LEVEL
005112  012703  005264                  MOV     #LEVELS,R3      ;MOVE ADDRESS OF PRIORITY LEVELS TO R3
005116  012704  000004                  MOV     #4,R4           ;DO 4 PRIORITY CHECKS
005122  012737  000400  002662  1$:     MOV     #400,TIME       ;SET UP WAIT LOOP COUNTER
005130  012710  010000                  MOV     #MA,(R0)        ;SET THE MAINTENANCE BIT AND
005134  005010                          CLR     (R0)            ;CLEAR TO POSSIBLY DO AN INIT
005136  013737  000014  001362          MOV     BPTVCT,$TMP1    ;SAVE BPT TRAP VECTOR
005144  012737  005222  000014          MOV     #3$,BPTVCT      ;INTERRUPTS TO 3$
005152  012337  177776                  MOV     (R3)+,PSW       ;SET CPU PRIORITY TO NEXT LEVEL
005156  000240                          NOP                     ;KILL A LITTLE TIME
005160  012710  000105                  MOV     #IE+F2+GO,(R0)  ;SET IE, FNCT2 AND GO ATTEMPTING ANOTHER INTERRUPT
005164  005337  002662          2$:     DEC     TIME            ;DECREMENT TIME
005170  001375                          BNE     2$              ;BRANCH BACK UNTIL ZERO
005172  012737  000340  177776          MOV     #LEVEL7,PSW     ;SET CPU PRIORITY BACK TO 7
005200  013737  001362  000014          MOV     $TMP1,BPTVCT    ;RESTORE BPT TRAP VECTOR
005206  162737  000040  002542          SUB     #40,LEVEL       ;PUT LOCATION 'LEVEL' AT NEXT PRIORITY - INTERRUPT FAILED
005214  005304                          DEC     R4              ;DECREMENT LOOP COUNTER
005216  001341                          BNE     1$              ;BRANCH BACK IF NOT ALL PRIORITY LEVELS CHECKED YET
005220  000416                          BR      4$              ;EXIT - THIS LOCATION DOESN'T BELONG TO A DR11
005222  012737  000340  177776  3$:     MOV     #LEVEL7,PSW     ;AHAH - THIS *IS* A DR11 - SET CPU PRIORITY BACK TO 7
005230  013737  001362  000014          MOV     $TMP1,BPTVCT    ;RESTORE BPT TRAP VECTOR
005236  016666  000010  000006          MOV     10(SP),6(SP)    ;MOVE THIS SUBROUTINE'S RETURN UP ONE SPOT ON STACK
005244  011666  000010                  MOV     (SP),10(SP)     ;MOVE TRAP ADDRESS TO RETURN'S OLD LOCATION
005250  062706  000006                  ADD     #6,SP           ;KICK GARBAGE' OFF STACK - GOT TO KEEP IT CLEAN
005254  000402                          BR      5$              ;BRANCH TO KICK OUT
005256  062716  000012          4$:     ADD     #12,(SP)        ;CORRECT RETURN TO NOT DO DR11 SETUP
005262  000207                  5$:     RTS     PC              ;KICK OUT
                                
005264  000300  000240          LEVELS: .WORD   LEVEL6,LEVEL5   ;PRIORITY LEVELS TO LOAD INTO THE PSW
005270  000200  000140                  .WORD   LEVEL4,LEVEL3
```

```
                        .SBTTL  SUBROUTINE TO AUTO SIZE DR11 BOARD CONFIGURATION
                ;*******************************************************************
                ;*
                ;*      THIS SUBROUTINE AUTOSIZES THE BOARD CONFIGURATION AND CALLS FOR THE
                ;*      PRINTING OF THE CONFIGURATION IF BIT 12 OF THE SWITCH REGISTER IS
                ;*      CLEAR.
                ;*
                ;*******************************************************************
005274  012700  001476          ASIZE:  MOV     #$DDW0,R0       ;MOVE ADDRESS OF FIRST DEVICE DESCRIPTOR WORD TO R0
005300  012701  000020                  MOV     #16.,R1         ;CLEAR 16 WORDS OUT OF THE DICTIONARY
005304  005020                  1$:     CLR     (R0)+           ;CLEAR THE WORD - SORRY CAN'T LOOK IT UP ANY MORE
005306  077301                          DEC     R1              ;DECREMENT THE LOOP COUNTER AND
005310  001375                          BNE     1$              ;BRANCH BACK IF NOT DONE YET
005312  004737  004374                  JSR     PC,BPINIT       ;GO RESET THE ".+2" AND "BPT" LOCATIONS
005316  004737  004344                  JSR     PC,DEVADS       ;GENERATE DEVICE ADDRESS TABLE
005322  005037  002416                  CLR     QTYBRD          ;CLEAR DEVICE COUNT
005326  005037  001470                  CLR     $DEVM           ;CLEAR DEVICE MAP
005332  013737  000004  002674          MOV     TOVECT,OLDPC1   ;SAVE TIMEOUT VECTOR
005340  012737  005416  000004          MOV     #3$,TOVECT      ;SET TIMEOUT POINTER TO 3$
005346  013737  000006  002676          MOV     TMOPSW,OLDPS1   ;SAVE TIMEOUT PS
005354  012737  000340  000006          MOV     #LEVEL7,TMOPSW  ;CPU PRIORITY TO 7
005362  012700  172604                  MOV     #172604,R0      ;POINT R0 TO UNIT #16 CSR ADDRESS LOCATION
005366  012701  001534                  MOV     #$DDW15,R1      ;LOAD DEVICE DESC ADDRESS
005372  012702  002516                  MOV     #VECADR+36,R2   ;POINT R2 TO UNIT #16 VECTOR ADDRESS LOCATION
005376  012705  000020                  MOV     #16.,R5         ;DO 16 BOARDS
005402  005010                  2$:     CLR     (R0)            ;CHECK FOR REGISTER EXISTENCE
005404  004737  005104                  JSR     PC,CHK4DR       ;GO CHECK FOR A DR11 AT THIS LOCATION IF IT DOES
005410  004737  004452                  JSR     PC,DRGET        ;GO EXTRACT INFO FROM THE DR11
005414  000402                          BR      4$              ;BRANCH OVER THE STACK CORRECTION
005416  062706  000004          3$:     ADD     #4,SP           ;CORRECT STACK AFTER TIMEOUT
005422  162700  000010          4$:     SUB     #10,R0          ;POINT R0 TO NEXT DEVICE ADDRESS
005426  024142                          CMP     -(R1),-(R2)     ;POINT R1 AND R2 TO NEXT DEVICE & VECTOR LOCATIONS
005430  077505                          DEC     R5              ;DECREMENT LOOP COUNTER
005432  001403                          BEQ     5$              ;EXIT IF ALL DONE WITH 16 BOARDS
005434  006337  001470                  ASL     $DEVM           ;ADJUST DEVICE MAP FOR NEXT UNIT CHECK
005440  000760                          BR      2$              ;GO CHECK NEXT LOCATION
005442  013737  002676  000006  5$:     MOV     OLDPS1,TMOPSW   ;RESTORE TIMEOUT PS
005450  013737  002674  000004          MOV     OLDPC1,TOVECT   ;RESTOPE TIMEOUT VECTOR
005456  032777  010000  173654          BIT     #BIT12,@SWR     ;CHECK FOR CONFIGURATION PRINT
005464  001005                          BNE     6$              ;BRANCH IF PRINT NOT REQUESTED
005466  005737  002416                  TST     QTYBRD          ;SEE IF ANY BOARDS WERE FOUND
005472  001402                          BEQ     6$              ;BRANCH TO RETURN IF NOT - NO BOARD INFO TO PRINT
005474  004737  004630                  JSR     PC,TYPCNF       ;GO TYPE THE BOARD CONFIGURATIONS
005500  000207                  6$:     RTS     PC              ;EXIT
```

```
                                        .SBTTL  SUBROUTINE TO GENERATE VECTOR ADDRESS TABLE
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE GENERATES THE VECTOR ADDRESS TABLE.
                                ;*
                                ;*******************************************************************
005502  012702  002460          VCTADS: MOV     #VECADR,R2      ;GET LOCATION OF VECTOR TABLE
005506  013700  001462                  MOV     $VECT1,R0       ;COPY BASE VECTOR
005512  042700  177400                  BIC     #177400,R0      ;CLEAR UPPER BYTE
005516  012701  000020                  MOV     #16.,R1         ;DO 16 VECTORS
005522  010022                  1$:     MOV     R0,(R2)+        ;PUT VECTOR ADDRESS IN TABLE
005524  062700  000010                  ADD     #10,R0          ;POINT R0 TO NEXT VECTOR ADDRESS
005530  005301                          DEC     R1              ;DECREMENT LOOP COUNTER
005532  001373                          BNE     1$              ;BRANCH IF NOT ALL DONE YET
005534  000207                          RTS     PC              ;EXIT
```

```
                                        .SBTTL  ROUTINE TO REPORT UNEXPECTED OR ERRONEOUS TRAPS OR INTERRUPTS
                                ;*******************************************************************
                                ;*
                                ;*      THIS IS THE ROUTINE TO REPORT UNEXPECTED OR ERRONEOUS TRAPS OR INTERRUPTS
                                ;*
                                ;*******************************************************************
005536  012737  000340  177776  CATCH:  MOV     #LEVEL7,PSW     ;REESTABLISH CPU PRIORITY AT 7
005544  012637  002544                  MOV     (SP)+,BDVECT    ;GET ADDRESS OF TRAP VECTOR + 4
005550  012637  002676                  MOV     (SP)+,OLDPS1    ;SAVE PS
005554  012637  002700                  MOV     (SP)+,OLDPC2    ;SAVE PC OF ADDRESS OF INSTRUCTION CAUSING TRAP
005560  012637  002702                  MOV     (SP)+,OLDPS2    ;SAVE 2ND PS
005564  162737  000004  002544          SUB     #4,BDVECT       ;ADJUST TO POINT TO TRAP ADDRESS
005572  104050                          ERROR   +50             ;UNEXPECTED TRAP OR INTERRUPT TO TRAP ADDRESS BELOW
005574  013746  002702                  MOV     OLDPS2,-(SP)    ;RESTORE PS RETURN ON STACK
005600  013746  002700                  MOV     OLDPC2,-(SP)    ;RESTORE PC RETURN ON STACK
005604  000002                          RTI                     ;RETURN
```

```
1182                                            .SBTTL  SUBROUTINE TO PRINT STATE OF A DDW BIT
1183                                    ;*****************************************************************
1184                                    ;*
1185                                    ;*      THIS SUBROUTINE PRINTS THE STATE OF THE BIT IN THE DDW THAT WAS
1186                                    ;*      PRELOADED INTO BITTST.
1187                                    ;*
1188                                    ;*****************************************************************
1189 005606  005046                    PSTATE: CLR     -(SP)           ;SHOW STATE AS ZERO INITIALLY
1190 005610  033737  002712  002720            BIT     BITTST,DDW      ;CHECK STATE OF BIT IN DDW USING BIT SET IN BITTST
1191 005616  001401                            BEQ     1$              ;BRANCH IF NOT SET
1192 005620  005216                            INC     (SP)            ;SHOW A '1' STATE FOR THAT BIT
1193 005622  104403                    1$:     TYPOS                   ;TYPE THE STATE, LEADING ZEROS SUPPRESSED
1194 005624     001     000                    .BYTE   1,0             ;TYPE 1 CHARACTER, SUPPRESS LEADING ZEROS
1195 005626  000207                            RTS     PC              ;EXIT
```

```
1196                                    .SBTTL  SUBROUTINE TO PRINT DEVICE PRIORITY
1197                                ;***************************************************************
1198                                ;*
1199                                ;*      THIS SUBROUTINE PRINTS THE DEVICE PRIORITY
1200                                ;*
1201                                ;***************************************************************
1202 005630  013746  002720         PNTPRI: MOV     DDW,-(SP)       ;PUT DEVICE DESCRIPTOR WORD ON STACK
1203 005634  006216                         ASR     (SP)            ;SHIFT RIGHT STACK LOCATION 5 PLACES
1204 005636  006216                         ASR     (SP)
1205 005640  006216                         ASR     (SP)
1206 005642  006216                         ASR     (SP)
1207 005644  006216                         ASR     (SP)
1208 005646  042716  177770                BIC     #177770,(SP)    ;MASK TO GET PRIORITY
1209 005652  104403                         TYPOS                   ;TYPE THE DEVICE PRIORITY
1210 005654     001     000                 .BYTE   1,0             ;TYPE 1 CHARACTER, SUPPRESS LEADING ZEROS
1211 005656  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
                                ;*
                                ;*****************************************************************
1220 005660  011637  001360    SETTUP: MOV     (SP),$TMP0      ;SAVE RETURN ADDRESS
1221                            ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
     005664  012706  001300             MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
     005670  005026             100$:   CLR     (R6)+           ;;CLEAR MEMORY LOCATION
     005672  022706  001340             CMP     #SWR,R6 ;;DONE?
     005676  001374                     BNE     100$            ;;LOOP BACK IF NO
     005700  012706  001300             MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
     005704  012737  026352  000020     MOV     #$SCOPE,@#IOTVEC;;IOT VECTOR FOR SCOPE ROUTINE
     005712  012737  000340  000022     MOV     #340,@#IOTVEC+2 ;;LEVEL 7
     005720  012737  027564  000030     MOV     #$ERROR,@#EMTVEC;;EMT VECTOR FOR ERROR ROUTINE
     005726  012737  000340  000032     MOV     #340,@#EMTVEC+2 ;;LEVEL 7
     005734  012737  026264  000034     MOV     #$TRAP,@#TRAPVEC;;TRAP VECTOR FOR TRAP CALLS
     005742  012737  000340  000036     MOV     #340,@#TRAPVEC+2;LEVEL 7
     005750  012737  030510  000024     MOV     #$PWRDN,@#PWRVEC;;POWER FAILURE VECTOR
     005756  012737  000340  000026     MOV     #340,@#PWRVEC+2 ;;LEVEL 7
     005764  013737  023302  023274     MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
     005772  005037  001376             CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
     005776  112737  000001  001315     MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
     006004  012737  011246  001306     MOV     #TST1+2,$LPADR  ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
     006012  012737  011246  001310     MOV     #TST1+2,$LPERR  ;;SETUP THE ERROR LOOP ADDRESS
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
     006020  013746  000004             MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
     006024  012737  006060  000004     MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
     006032  012737  177570  001340     MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
     006040  012737  177570  001342     MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
     006046  022777  177777  173264     CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
     006054  001011                     BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
     006056  000402                     BR      65$             ;;BRANCH IF NO TIMEOUT
     006060  062706  000004     64$:    ADD     #4,SP           ;;CLEAN UP STACK AFTER INTERRUPT
     006064  012737  000176  001340     MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
     006072  012737  000174  001342     MOV     #DISPREG,DISPLAY
     006100  012637  000004     66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
     006104  005037  001416             CLR     $PASS           ;;CLEAR PASS COUNT
     006110  132737  000200  001433     BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
     006116  001403                     BEQ     67$             ;;YES,USE NON-APT SWITCH
     006120  012737  001434  001340     MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
     006126                     67$:
1222 006126  013746  001360             MOV     $TMP0,-(SP)     ;PUT RETURN ADDRESS ON STACK AND
1223 006132  000207                     RTS     PC              ;RETURN TO THE CALLING ROUTINE
```

```
                                        .SBTTL  MEMORY MANAGEMENT AND LOCATION CHECK SUBROUTINE
                                ;*******************************************************************
                                ;*
                                ;*      THIS SUBROUTINE CHECKS FOR MEMORY MANAGEMENT EXISTENCE AND WHETHER OR
                                ;*      NOT A LOCATION IN UPPER MEMORY EXISTS.
                                ;*
                                ;*******************************************************************
006134  005737  002714          TSTMM:  TST     MEMGMT          ;TEST TO SEE IF MEMORY MANAGEMENT EXISTS
006140  001440                          BEQ     4$              ;BRANCH IF NOT
006142  012737  000401  006152          MOV     #CY+GO,1$       ;SET UP BIT TEST DATA
006150  040227                          BIC     R2,(PC)+        ;TEST TO SEE IF BOTH THE CYCLE AND GO BITS ARE SET
006152  000401                  1$:     .WORD   CY+GO           ;LOCATION TO STORE THE CYCLE AND GO BITS
006154  001032                          BNE     4$              ;KICK OUT IF CYCLE AND/OR GO ARE CLEAR
006156  032737  000060  002574          BIT     #X6+X7,ECSR     ;SEE IF XBA16 OR XBA17 WERE SET IN EXPECTED DATA
006164  001426                          BEQ     4$              ;BRANCH OUT IF NOT
006166  032737  000040  002574          BIT     #X7,ECSR        ;SEE IF XBA17 IS SET
006174  001005                          BNE     2$              ;GO CHECK STATUS OF XBA16 IF SET
006176  132737  000001  002715          BITB    #BIT0,MEMGMT+1  ;SEE IF 200000+NOCARE WAS FOUND TO EXIST - IF NOT,
006204  001420                          BEQ     5$              ;GO SET EXPECTED ERROR AND NEX BITS AND CHECK FOR ERROR
006206  000415                          BR      4$              ;BRANCH OUT IF LOCATION EXISTS
006210  032737  000020  002574  2$:     BIT     #X6,ECSR        ;SEE IF XBA16 IS SET
006216  001005                          BNE     3$              ;BRANCH TO CHECK 600000+NOCARE IF SET
006220  132737  000002  002715          BITB    #BIT1,MEMGMT+1  ;SEE IF 400000+NOCARE WAS FOUND TO EXIST - IF NOT,
006226  001407                          BEQ     5$              ;GO SET EXPECTED ERROR AND NEX BITS AND CHECK FOR ERROR
006230  000404                          BR      4$              ;BRANCH OUT IF LOCATION EXISTS
006232  132737  000004  002715  3$:     BITB    #BIT2,MEMGMT+1  ;SEE IF 600000+NOCARE WAS FOUND TO EXIST - IF NOT,
006240  001402                          BEQ     5$              ;GO SET EXPECTED ERROR AND NEX BITS AND CHECK FOR ERROR
006242  062716  000002          4$:     ADD     #2,(SP)         ;CORRECT PC RETURN
006246  000207                  5$:     RTS     PC              ;KICK OUT
```

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

000002                          .RADIX  2               ;THIS ENABLES YOU TO SEE THE PATTERNS IN BINARY
006250  177777          PATRNS: .WORD   1111111111111111  ;ALL SET BITS
006252  000000                  .WORD   0000000000000000  ;ALL CLEAR BITS
006254  052525                  .WORD   0101010101010101  ;EVEN BITS SET, ODD BITS CLEAR
006256  125252                  .WORD   1010101010101010  ;ODD BITS SET, EVEN BITS CLEAR
006260  031463                  .WORD   0011001100110011  ;PAIRS OF BITS SET
006262  007417                  .WORD   0000111100001111  ;GROUPS OF 4 BITS SET
006264  000377                  .WORD   0000000011111111  ;UPPER BYTE CLEAR, LOWER BYTE SET
000010                          .RADIX  8               ;THIS RETURNS MODE BACK TO OCTAL
```

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

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

```
1311                                            .SBTTL  EXPECTED DATA TABLE FOR CSR CHECK TEST 3
1312
1313                                            ; X=      0     1     2     3     4     5     6     7
1314 007266  010200  010000  011202  MAISET:   .WORD   10200,10000,11202,11002,12204,12004,13206,13006  ;CSR 01000X EXPECTED
1315 007306  014210  014010  015212            .WORD   14210,14010,15212,15012,16214,16014,17216,17016  ;CSR 01001X EXPECTED
1316 007326  010220  010020  011222            .WORD   10220,10020,11222,11022,12224,12024,13226,13026  ;CSR 01002X EXPECTED
1317 007346  014230  014030  015232            .WORD   14230,14030,15232,15032,16234,16034,17236,17036  ;CSR 01003X EXPECTED
1318 007366  010240  010040  011242            .WORD   10240,10040,11242,11042,12244,12044,13246,13046  ;CSR 01004X EXPECTED
1319 007406  014250  014050  015252            .WORD   14250,14050,15252,15052,16254,16054,17256,17056  ;CSR 01005X EXPECTED
1320 007426  010260  010060  011262            .WORD   10260,10060,11262,11062,12264,12064,13266,13066  ;CSR 01006X EXPECTED
1321 007446  014270  014070  015272            .WORD   14270,14070,15272,15072,16274,16074,17276,17076  ;CSR 01007X EXPECTED
1322 007466  010300  010100  011302            .WORD   10300,10100,11302,11102,12304,12104,13306,13106  ;CSR 01010X EXPECTED
1323 007506  014310  014110  015312            .WORD   14310,14110,15312,15112,16314,16114,17316,17116  ;CSR 01011X EXPECTED
1324 007526  010320  010120  011322            .WORD   10320,10120,11322,11122,12324,12124,13326,13126  ;CSR 01012X EXPECTED
1325 007546  014330  014130  015332            .WORD   14330,14130,15332,15132,16334,16134,17336,17136  ;CSR 01013X EXPECTED
1326 007566  010340  010140  011342            .WORD   10340,10140,11342,11142,12344,12144,13346,13146  ;CSR 01014X EXPECTED
1327 007606  014350  014150  015352            .WORD   14350,14150,15352,15152,16354,16154,17356,17156  ;CSR 01015X EXPECTED
1328 007626  010360  010160  011362            .WORD   10360,10160,11362,11162,12364,12164,13366,13166  ;CSR 01016X EXPECTED
1329 007646  014370  014170  015372            .WORD   14370,14170,15372,15172,16374,16174,17376,17176  ;CSR 01017X EXPECTED
1330 007666  010600  011202  011602            .WORD   10600,11202,11602,12204,12604,13206,13606,14210  ;CSR 01040X EXPECTED
1331 007706  014610  015212  015612            .WORD   14610,15212,15612,16214,16614,17216,17616,10200  ;CSR 01041X EXPECTED
1332 007726  010620  011222  011622            .WORD   10620,11222,11622,12224,12624,13226,13626,14230  ;CSR 01042X EXPECTED
1333 007746  014630  015232  015632            .WORD   14630,15232,15632,16234,16634,17236,17636,10220  ;CSR 01043X EXPECTED
1334 007766  010640  011242  011642            .WORD   10640,11242,11642,12244,12644,13246,13646,14250  ;CSR 01044X EXPECTED
1335 010006  014650  015252  015652            .WORD   14650,15252,15652,16254,16654,17256,17656,10240  ;CSR 01045X EXPECTED
1336 010026  010660  011262  011662            .WORD   10660,11262,11662,12264,12664,13266,13666,14270  ;CSR 01046X EXPECTED
1337 010046  014670  015272  015672            .WORD   14670,15272,15672,16274,16674,17276,17676,10260  ;CSR 01047X EXPECTED
1338 010066  010700  011302  011702            .WORD   10700,11302,11702,12304,12704,13306,13706,14310  ;CSR 01050X EXPECTED
1339 010106  014710  015312  015712            .WORD   14710,15312,15712,16314,16714,17316,17716,10300  ;CSR 01051X EXPECTED
1340 010126  010720  011322  011722            .WORD   10720,11322,11722,12324,12724,13326,13726,14330  ;CSR 01052X EXPECTED
1341 010146  014730  015332  015732            .WORD   14730,15332,15732,16334,16734,17336,17736,10320  ;CSR 01053X EXPECTED
1342 010166  010740  011342  011742            .WORD   10740,11342,11742,12344,12744,13346,13746,14350  ;CSR 01054X EXPECTED
1343 010206  014750  015352  015752            .WORD   14750,15352,15752,16354,16754,17356,17756,10340  ;CSR 01055X EXPECTED
1344 010226  010760  011362  011762            .WORD   10760,11362,11762,12364,12764,13366,13766,14370  ;CSR 01056X EXPECTED
1345 010246  014770  015372  015772            .WORD   14770,15372,15772,16374,16774,17376,17776,10360  ;CSR 01057X EXPECTED
```

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

010266  005037  002672          START1: CLR     MANSIZE         ;CLEAR THE MANSIZE SO WE WILL AUTOSIZE
010272  005037  001416          START:  CLR     $PASS           ;CLEAR $PASS
010276  005037  001420                  CLR     $PASS+2         ;CLEAR $PASS+2
010302  005037  001412                  CLR     $FATAL          ;CLEAR ERROR NO.
010306  005037  001410                  CLR     $MSGTYP         ;CLEAR MESSAGE TYPE
010312  005037  001414                  CLR     $TESTN          ;CLEAR TEST NO.
010316  005037  001422                  CLR     $DEVCT          ;CLEAR DEVICE COUNT
010322  005037  001424                  CLR     $UNIT           ;CLEAR UNIT NUMBER
010326  012737  000006  000004          MOV     #TOVECT+2,TOVECT;INITIALIZE TIMEOUT VECTORS TO 6
010334  012737  000003  000006          MOV     #BPT,TMOPSW     ;CATCHER ROUTINE
010342  012706  001300                  MOV     #STACK,SP       ;INITIALIZE THE STACK
010346  004737  005660                  JSR     PC,SETTUP       ;GO TO THE SETUP ROUTINE TO INITIALIZE VECTORS
010352  005037  001424                  CLR     $UNIT           ;CLEAR $UNIT
010356  005037  001422                  CLR     $DEVCT          ;CLEAR $DEVCT
010362  005037  001414                  CLR     $TESTN          ;CLEAR $TESTN
010366  005037  002710                  CLR     EOPLOC          ;CLEAR EOPLOC
010372  132737  000001  001432          BITB    #BIT0,$ENV      ;CHECK IF ON APT
010400  001404                          BEQ     1$              ;BR IF NOT APT
010402  132737  000200  001433          BITB    #BIT7,$ENVM     ;DID APT SIZE
010410  001007                          BNE     2$              ;BR, IF APT SIZED
010412  022737  177777  002672  1$:     CMP     #-1,MANSIZE     ;WAS CONFIGURATION SET UP IN MULT. BOARD ROUTINE?
010420  001422                          BEQ     BEGIN           ;IF YES, SKIP SELF-SIZING
010422  004737  005274                  JSR     PC,ASIZE        ;AUTOMATICALLY SIZE FOR BOARD CONFIGURATION
010426  000417                          BR      BEGIN           ;BRANCH
010430  005037  002416          2$:     CLR     QTYBRD          ;CLEAR  DEVICE CNT
010434  013702  001470                  MOV     $DEVM,R2        ;MOVE DEVICE MAP TO R2
010440  005702                  3$:     TST     R2              ;TEST MSB OF DEVICE MAP
010442  100002                          BPL     4$              ;BR, IF MSB IS ZERO
010444  005237  002416                  INC     QTYBRD          ;INCREMENT DEVICE COUNT, IF MSB=1
010450  000241                  4$:     CLC                     ;CLEAR THE CARRY BIT FOR THE ROL
010452  006102                          ROL     R2              ;SHIFT NEXT BIT INTO MSB POSITION
010454  001371                          BNE     3$              ;CONTINUE CHECKING $DEVM, IF MORE BITS SET
010456  004737  004344                  JSR     PC,DEVADS       ;GENERATE DEVICE ADDRESS TABLE
010462  004737  005502                  JSR     PC,VCTADS       ;GENERATE VECTOR ADDRESS TABLE
```

```
1387                                                .SBTTL  DETERMINE MEM MGMT AND UPPER MEMORY EXISTENCE
1388 010466  005737  001416          BEGIN:  TST     $PASS           ;SEE IF THIS IS THE FIRST PASS
1389 010472  001145                         BNE     BEGIN1          ;BRANCH IF NOT
1390 010474  005737  001420                 TST     $PASS+2         ;SEE IF UPPER LOCATION HAS BEEN SET
1391 010500  001142                         BNE     BEGIN1          ;BRANCH IF NOT
1392 010502  005037  002714                 CLR     MEMGMT          ;CLEAR THE MEMORY MANAGEMENT FLAG
1393 010506  013737  000004  002674         MOV     TOVECT,OLDPC1   ;SAVE TIMEOUT VECTOR
1394 010514  012737  010750  000004         MOV     #3$,TOVECT      ;TIMEOUT VECTOR TO 3$
1395 010522  013737  000006  002676         MOV     TMOPSW,OLDPS1   ;SAVE TIMEOUT PS
1396 010530  012737  000340  000006         MOV     #LEVEL7,TMOPSW  ;PS TIMEOUT TO PRIORITY 7
1397 010536  005737  177572                 TST     MMR0            ;TEST FOR THE PRESENCE OF MEMORY MANAGEMENT
1398 010542  105237  002714                 INCB    MEMGMT          ;INCREMENT FLAG SHOWING MEMORY MANAGEMENT EXISTS
1399 010546  012737  077406  172304         MOV     #77406,KIPDR2   ;MAKE KIPDR2 RESIDENT
1400 010554  012737  077406  172324         MOV     #77406,KDPDR2   ;MAKE KDPDR2 RESIDENT
1401 010562  013737  000250  002700         MOV     MMVECT,OLDPC2   ;SAVE MEMORY MANAGEMENT VECTOR
1402 010570  012737  010722  000250         MOV     #1$,MMVECT      ;MEMORY MANAGEMENT VECTOR TO 1$
1403 010576  013737  000252  002702         MOV     MMPS,OLDPS2     ;SAVE MEMORY MANAGEMENT PS
1404 010604  012737  000340  000252         MOV     #LEVEL7,MMPS    ;MEMORY MANAGEMENT PS TO PRIORITY 7
1405 010612  005237  177572                 INC     MMR0            ;TURN ON MEMORY MANAGEMENT
1406 010616  012737  002400  172344         MOV     #2400,KIPAR2    ;SET UP KIPAR2 TO ACCESS LOCATION 240000+BITS 12-0 OF NOCARE
1407 010624  012737  002400  172364         MOV     #2400,KDPAR2    ;SET UP KDPAR2 TO ACCESS LOCATION 240000+BITS 12-0 OF NOCARE
1408 010632  005737  052414                 TST     NOCARE          ;SEE IF BITS 12-0 OF NOCARE ADRS +240000 EXISTS
1409 010636  152737  000001  002715         BISB    #BIT0,MEMGMT+1  ;SET BIT 0 OF UPPER BYTE OF MEMGMT IF IT DOES
1410 010644  012737  004400  172344         MOV     #4400,KIPAR2    ;SET UP KIPAR2 TO ACCESS LOCATION 440000+BITS 12-0 OF NOCARE
1411 010652  012737  004400  172364         MOV     #4400,KDPAR2    ;SET UP KDPAR2 TO ACCESS LOCATION 440000+BITS 12-0 OF NOCARE
1412 010660  005737  052414                 TST     NOCARE          ;SEE IF BITS 12-0 OF NOCARE ADRS +440000 EXISTS
1413 010664  152737  000002  002715         BISB    #BIT1,MEMGMT+1  ;SET BIT 1 OF UPPER BYTE OF MEMGMT IF IT DOES
1414 010672  012737  006400  172344         MOV     #6400,KIPAR2    ;SET UP KIPAR2 TO ACCESS LOCATION 640000+BITS 12-0 OF NOCARE
1415 010700  012737  006400  172364         MOV     #6400,KDPAR2    ;SET UP KDPAR2 TO ACCESS LOCATION 640000+BITS 12-0 OF NOCARE
1416 010706  005737  052414                 TST     NOCARE          ;SEE IF BITS 12-0 OF NOCARE ADRS +640000 EXISTS
1417 010712  152737  000004  002715         BISB    #BIT2,MEMGMT+1  ;SET BIT 2 OF UPPER BYTE OF MEMGMT IF IT DOES
1418 010720  000402                         BR      2$              ;BRANCH OVER STACK CORRECTION
1419 010722  062706  000004          1$:    ADD     #4,SP           ;CORRECT STACK AFTER MM TRAP
1420 010726  005037  177572          2$:    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
1421 010732  013737  002702  000252         MOV     OLDPS2,MMPS     ;RESTORE MEMORY MANAGEMENT PS
1422 010740  013737  002700  000250         MOV     OLDPC2,MMVECT   ;RESTORE MEMORY MANAGEMENT VECTOR
1423 010746  000402                         BR      4$              ;BRANCH OVER STACK CORRECTION
1424 010750  062706  000004          3$:    ADD     #4,SP           ;CORRECT STACK AFTER TIMEOUT
1425 010754  013737  002676  000006  4$:    MOV     OLDPS1,TMOPSW   ;RESTORE TIMEOUT PS
1426 010762  013737  002674  000004         MOV     OLDPC1,TOVECT   ;RESTORE TIMEOUT VECTOR
1427 010770  104401  035704                 TYPE    ,M1             ;TYPE:  '(^X) INHIBITS EOP'S, (^Y) FOR ERROR SUMMARY'
1428                                                                 ;       'UNIBUS HANG?  RESTART AT ADDRESS '
1429 010774  012746  052336                 MOV     #UBHANG,-(SP)   ;MOVE ADDRESS OF HANG ROUTINE TO STACK
1430 011000  104402                         TYPOC                   ;GO TYPE THE ADDRESS IN OCTAL
1431 011002  104401  036023                 TYPE    ,M1A            ;       'CZDRLB0 DR11 GEN NPR INTFC LOGIC TEST'
```

```
                                        .SBTTL  PREPARE ADDRESSES AND VECTORS FOR UUT
011006  012737  000001  002546  BEGIN1: MOV     #BIT0,DEVMSK    ;SET UP BIT MASK TO TEST $DEVM FOR DEVICES
011014  005037  002550                CLR     TABINX          ;CLEAR LOCATION TO STORE TABLE OFFSETS
011020  033737  002546  001470  TSTDEV: BIT     DEVMSK,$DEVM    ;CHECK TO SEE IF DEVICE IS TO BE TESTED
011026  001026                        BNE     2$              ;BR, IF YES
011030  005737  002546                TST     DEVMSK          ;SEE IF BIT 15 IS SET
011034  100013                        BPL     1$              ;BRANCH TO CONTINUE IF NOT SET
011036  005737  001416                TST     $PASS           ;SEE IF THIS IS THE FIRST PASS
011042  001361                        BNE     BEGIN1          ;BRANCH TO REINITIALIZE THE DEVMSK LOCATION IF NOT
011044  005737  001420                TST     $PASS+2         ;SEE IF THIS IS THE FIRST PASS
011050  001356                        BNE     BEGIN1          ;BRANCH TO REINITIALIZE THE DEVMSK LOCATION IF NOT
011052  104401  035553                TYPE    ,NODVPR         ;TYPE:  'NO DEVICES RECOGNIZED - DIAGNOSTIC CANNOT BE RUN'
                                                                ;       'RESTART AT 204 IF A DEVICE IS PRESENT'
011056  000000                        HALT                    ;FATAL ERROR - HALT HERE
011060  000137  010266                JMP     START1          ;JUMP TO START1 TO CHECK AGAIN FOR A MODULE
011064  006337  002546          1$:   ASL     DEVMSK          ;SHIFT MASK TO CHECK NEXT $DEVM BIT
011070  062737  000002  002550        ADD     #2,TABINX       ;INCREMENT TABLE INDEX
011076  005237  001424                INC     $UNIT           ;INCREMENT UNIT NUMBER
011102  000746                        BR      TSTDEV          ;GO TEST NEXT BIT OF DEVICE MAP

011104  006337  002546          2$:   ASL     DEVMSK          ;UPDATE DEVICE MAP TEST MASK
011110  013702  002550                MOV     TABINX,R2       ;MOVE TABLE OFFSET TO R2
011114  062737  000002  002550        ADD     #2,TABINX       ;UPDATE TABLE OFFSET FOR NEXT DEVICE
011122  016200  002420                MOV     REGADR(R2),R0   ;PUT UUT ADDRESS INTO R0
011126  012701  002520                MOV     #WCR,R1         ;POINT R1 TO STORAGE AREA FOR UUT ADDRESSES
011132  012703  000004                MOV     #4,R3           ;MOVE 4 ADDRESSES
011136  010021                  3$:   MOV     R0,(R1)+        ;TRANSFER UUT ADDRESS
011140  062700  000002                ADD     #2,R0           ;POINT TO NEXT UUT REGISTER
011144  005303                        DEC     R3              ;DECREMENT THE LOOP COUNTER AND
011146  001373                        BNE     3$              ;BRANCH IF NOT DONE TRANSFERING
011150  016200  002460                MOV     VECADR(R2),R0   ;PUT UUT VECTOR INTO R0
011154  010021                        MOV     R0,(R1)+        ;TRANSFER UUT VECTORS TO ACTIVE TABLE AREA
011156  062700  000002                ADD     #2,R0           ;POINT TO NEXT VECTOR
011162  010011                        MOV     R0,(R1)         ;TRANSFER VECTOR TO TABLE AREA
011164  016237  001476  002720        MOV     $DDW0(R2),DDW   ;SET UP DDW TO PROPER DEVICE DESCRIPTOR WORD
011172  013737  002720  002554        MOV     DDW,DRLEV       ;MOVE THE WORD TO THE DRLEV LOCATION
011200  042737  177437  002554        BIC     #177437,DRLEV   ;STRIP ALL BITS EXCEPT BR LEVEL
011206  105037  027423          REINIT: CLRB    CHARCT          ;CLEAR THE CHARACTER LOCATION OF ANY CHARACTER
011212  004737  004374                JSR     PC,BPINIT       ;GO RESET THE ".+2" AND "BPT" LOCATIONS
011216  004737  004036                JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
011222  105737  002710                TSTB    EOPLOC          ;SEE IF ^X IS ENABLED (IS THE PRINTER DISABLED)
011226  001006                        BNE     TST1            ;GO DO TEST IF NOT
                                    ;*******************************************************************
                                    ;*      *DO*NOT*REMOVE*THE*WAIT*LOOP*ROUTINE*BELOW*.  BECAUSE OF THE SPEED OF THIS DIAGNOSTI
                                    ;*      SOME VIDEO TERMINALS PRINT ERRONEOUS CHARACTER(S) WITH THE EOP MESSAGE DUE TO THE
                                    ;*      RESET EXECUTED IN TEST 4.  THIS WAIT LOOP ENABLES THOSE TERMINALS TO 'CATCH UP'
                                    ;*      BEFORE ITS EXECUTION.
                                    ;*******************************************************************
011230  012737  010000  011240        MOV     #10000,2$       ;REESTABLISH THE WAIT LOOP COUNTER
011236  005327                  1$:   DEC     (PC)+           ;DECREMENT THE LOCATION TO KILL TIME
011240  010000                  2$:   .WORD   10000           ;LOCATION TO BE DECREMENTED
011242  001375                        BNE     1$              ;BRANCH BACK UNTIL ZERO
                                    ;*******************************************************************
                                    ;
                                    ;MAIN PROGRAM - DEVICE TESTS
                                    ;
                                    ;*******************************************************************
```

```
                                        .SBTTL  TEST #1 - CAN ALL DR11 REG BE ADDRESSED WITHOUT ERROR?
                                        ;*********************************************************************
                                        ;*TEST 1        CAN ALL DR11 REG BE ADDRESSED WITHOUT ERROR?
                                        ;*
                                        ;*      THIS TEST INSURES THAT THE CSR, BAR, BDR AND WCR REGISTERS CAN BE
                                        ;*      ACCESSED FOR THIS DEVICE.  IF NOT, THE REST OF THE DIAGNOSTIC CANNOT
                                        ;*      BE RUN.
                                        ;*
                                        ;*********************************************************************
     011244                             TST1:
     011244  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     011246  012737  011254  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
1497 011254  005037  001312             999$:   CLR     $ERTTL          ;CLEAR THE ERROR TOTAL - NEW PASS
1498 011260  012737  011362  000004             MOV     #1$,BUSERR      ;CPU ERROR VECTOR TO 1$
1499 011266  013737  002520  002552             MOV     WCR,DREG        ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
1500 011274  005777  171220                     TST     @WCR            ;ACCESS REGISTER
1501 011300  013737  002522  002552             MOV     BAR,DREG        ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
1502 011306  005777  171210                     TST     @BAR            ;ACCESS REGISTER
1503 011312  013737  002524  002552             MOV     CSR,DREG        ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
1504 011320  005777  171200                     TST     @CSR            ;ACCESS REGISTER
1505 011324  013737  002526  002552             MOV     BDR,DREG        ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
1506 011332  005777  171170                     TST     @BDR            ;ACCESS REGISTER
1507 011336  013737  002530  002552             MOV     DRINV,DREG      ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
1508 011344  005777  171160                     TST     @DRINV          ;ACCESS REGISTER
1509 011350  005737  001312                     TST     $ERTTL          ;SEE IF THERE WERE ANY ERRORS
1510 011354  001414                             BEQ     2$              ;BRANCH TO CONTINUE IF NONE
1511 011356  000137  023074                     JMP     ENDEV           ;GO TO END OF DEVICE ROUTINE - FATAL ERRORS
1512 011362  012637  002674             1$:     MOV     (SP)+,OLDPC1    ;SAVE PC OF TRAP FOR ERROR PRINTOUT
1513 011366  012637  002676                     MOV     (SP)+,OLDPS1    ;SAVE PS FOR RESTORATION AFTER ERROR CALL
1514 011372  104001                             ERROR   +1              ;CANNOT ACCESS DR11 REGISTER
1515 011374  013746  002676                     MOV     OLDPS1,-(SP)    ;PUT PS BACK ON STACK
1516 011400  013746  002674                     MOV     OLDPC1,-(SP)    ;PUT PC BACK ON STACK
1517 011404  000002                             RTI                     ;RETURN TO PROGRAM
1518 011406  012737  000006  000004     2$:     MOV     #6,BUSERR       ;RESTORE #6 TO BUS ERROR
```

```
1525                                    .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.
                                        ;*
                                        ;*********************************************************************
     011414                             TST2:
     011414  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     011416  012737  011424  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
1526 011424  005077  171074             999$:   CLR     @CSR            ;GO TO CSR
1527 011430  012777  100000  171066             MOV     #EIR,@CSR       ;FORCE TO BE EIR
1528 011436  017737  171062  002612             MOV     @CSR,BORW       ;ATTEMPT EIR READ
1529 011444  042737  177776  002612             BIC     #CBIT0,BORW     ;MASK OFF ALL BITS EXCEPT BIT 0
1530 011452  013737  002720  001362             MOV     DDW,$TMP1       ;GET DEVICE DESCRIPTOR WORD
1531 011460  042737  177776  001362             BIC     #CBIT0,$TMP1    ;MASK OFF ALL BUT B OR W BIT
1532 011466  001403                             BEQ     1$              ;BRANCH IF IT IS CLEAR
1533 011470  005037  001362                     CLR     $TMP1           ;CLEAR THE BIT
1534 011474  000403                             BR      2$              ;GO TEST THE BIT
1535 011476  012737  000001  001362  1$:        MOV     #1,$TMP1        ;SET THE BIT
1536 011504  023737  002612  001362  2$:        CMP     BORW,$TMP1      ;B OR W STATE AS EXPECTED?
1537 011512  001401                             BEQ     TST3            ;BRANCH IF OK
1538 011514  104002                             ERROR   +2              ;DR11-B OR W MODE INCORRECT (0=B, 1=W)
```

```
1546                                    .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.
                                        ;*
                                        ;*************************************************************
     011516                             TST3:
     011516  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     011520  012737  011602  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
1547 011526  004737  004036                     JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
1548 011532  012737  000200  001362             MOV     #RY,$TMP1       ;MOVE READY BIT TO $TMP1
1549 011540  032737  000004  002720             BIT     #BIT2,DDW       ;TEST TO SEE IF CABLE IS IN
1550 011546  001003                             BNE     1$              ;BRANCH AROUND NON-CABLE SETUP IF IN
1551 011550  052737  127000  001362             BIS     #127000,$TMP1   ;SET THE BITS TO BE EXPECTED IN $TMP1
1552 011556  012701  007266          1$:        MOV     #MAISET,R1      ;MOVE ADDRESS OF EXPECTED PATTERNS TO R1
1553 011562  012702  010000                     MOV     #MA,R2          ;START WITH JUST THE MAINTENANCE BIT
1554 011566  005037  002716                     CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +202
1555 011572  012700  000002                     MOV     #2,R0           ;DO 2 SETS OF 200 PATTERNS
1556 011576  012703  000200          2$:        MOV     #200,R3         ;MOVE 200 TO THE LOOP COUNTER
1557 011602  052777  010000  170714  999$:      BIS     #MA,@CSR        ;SET MAINTENANCE AND
1558 011610  005077  170710                     CLR     @CSR            ;CLEAR TO DO AN INIT
1559 011614  017737  170704  002562             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
1560 011622  023737  001362  002562             CMP     $TMP1,RCSR      ;MAKE SURE EXPECTED DATA CAME UP
1561 011630  001404                             BEQ     3$              ;BRANCH IF SO
1562 011632  013737  001362  002574             MOV     $TMP1,ECSR      ;MOVE EXPECTED DATA TO ECSR
1563 011640  104032                             ERROR   +32             ;CSR IS WRONG
1564 011642  012777  177777  170650  3$:        MOV     #-1,@WCR        ;MOVE 1 WORD COUNT TO WCR IN CASE OF IE ENABLED
1565 011650  012777  052414  170644             MOV     #NOCARE,@BAR    ;MOVE A NOT-CARE ADDRESS TO BAR FOR SAME REASON
1566 011656  010277  170642                     MOV     R2,@CSR         ;SET THE PARTICULAR FUNCTION BITS IN CSR
1567 011662  017737  170636  002562             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
1568 011670  011137  002574                     MOV     (R1),ECSR       ;MOVE EXPECTED DATA TO ECSR
1569 011674  023737  002574  002562             CMP     ECSR,RCSR       ;COMPARE EXPECTED WITH RECEIVED
1570 011702  001430                             BEQ     7$              ;BRANCH IF OK
1571 011704  012737  000401  011714             MOV     #CY+GO,4$       ;REESTABLISH TEST PATTERN
1572 011712  040227                             BIC     R2,(PC)+        ;SEE IF BOTH CYCLE AND GO WERE SET
1573 011714  000401                  4$:        .WORD   CY+GO           ;LOCATION TO HOLD BOTH CYCLE AND GO BITS
1574 011716  001016                             BNE     6$              ;BRANCH TO ERROR ONLY IF CYCLE AND GO WERE SET
1575 011720  005737  002714                     TST     MEMGMT          ;SEE IF MEMORY MANAGEMENT IS OUT THERE
1576 011724  001404                             BEQ     5$              ;BRANCH IF SO TO CHECK LOCATION EXISTENCE
1577 011726  032737  000060  002574             BIT     #X6+X7,ECSR     ;SEE IF EITHER XBA16 OR XBA17 ARE SET
1578 011734  001407                             BEQ     6$              ;BRANCH TO ERROR IF BOTH ARE CLEAR
1579 011736  052737  140000  002574  5$:        BIS     #ER+NX,ECSR     ;SET THE ERROR AND NEX BITS - EXPECT THEM TO SET
1580 011744  023737  002562  002574             CMP     RCSR,ECSR       ;NOW SEE IF DATA MATCHES
1581 011752  001415                             BEQ     10$             ;BRANCH AROUND ERROR IF IT DOES
1582 011754  010237  002540          6$:        MOV     R2,BUT          ;MOVE THE BITS SET INTO CSR TO THE BUT LOCATION
1583 011760  104202                             ERROR   +202            ;CSR PATTERN NOT CORRECT
1584 011762  000411                             BR      10$             ;BRANCH AROUND MM TESTS
1585 011764  012737  000401  011774  7$:        MOV     #CY+GO,8$       ;REESTABLISH TEST PATTERN
1586 011772  040227                             BIC     R2,(PC)+        ;SEE IF BOTH CYCLE AND GO WERE SET
1587 011774  000401                  8$:        .WORD   CY+GO           ;LOCATION TO HOLD BOTH CYCLE AND GO BITS
1588 011776  001003                             BNE     10$             ;BRANCH AROUND MEM MGMT TEST IF EITHER OR BOTH WERE CLEAR
1589 012000  004737  006134          9$:        JSR     PC,TSTMM        ;GO CHECK FOR MEMORY MANAGEMENT EXISTENCE
1590 012004  000754                             BR      5$              ;IF RETURN IS HERE, GO BACK TO SET EXPECTED DATA
1591 012006  062701  000002          10$:       ADD     #2,R1           ;INCREMENT R1 TO NEXT EXPECTED PATTERN
```

```
 1592 012012  005202                    INC     R2              ;INCREMENT THE PATTERN
 1593 012014  005303                    DEC     R3              ;DECREMENT THE LOOP COUNTER
 1594 012016  001271                    BNE     999$            ;BRANCH BACK IF NOT DONE
 1595 012020  062702  000200            ADD     #200,R2         ;ADD 200 TO PATTERN LOCATION
 1596 012024  005300                    DEC     R0              ;DECREMENT THE LOOP COUNTER AND
 1597 012026  001263                    BNE     2$              ;BRANCH BACK IF 2ND OCTAL GROUP NOT DONE
```

```
1604                                    .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.
                                        ;*
                                        ;*****************************************************************
     012030                             TST4:
     012030  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     012032  012737  012040  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
1605 012040  012777  010000  170456  999$:      MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
1606 012046  005077  170452                     CLR     @CSR            ;CLEAR TO DO AN INIT
1607 012052  012777  177777  170440             MOV     #-1,@WCR        ;ALL ONES TO WCR
1608 012060  017737  170434  002572             MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
1609 012066  022737  177777  002572             CMP     #-1,RWCR        ;SEE IF DATA WAS LOADED PROPERLY
1610 012074  001423                             BEQ     4$              ;BRANCH IF OK
1611 012076  012737  012106  001310             MOV     #1$,$LPERR      ;MOVE NEW LOOP ON ERROR LOCATION TO $LPERR
1612 012104  000412                             BR      2$              ;BRANCH OVER LOOP SETUP
1613 012106  012777  177777  170404  1$:       MOV     #-1,@WCR        ;ALL ONES TO WCR
1614 012114  017737  170400  002572             MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
1615 012122  022737  177777  002572             CMP     #-1,RWCR        ;SEE IF DATA WAS LOADED PROPERLY
1616 012130  001401                             BEQ     3$              ;BRANCH IF OK
1617 012132  104010                     2$:     ERROR   +10             ;ATTEMPT TO SET ALL WCR BITS FAILED
1618 012134  032777  001000  167176  3$:       BIT     #BIT9,@CSR      ;SEE IF WE SHOULD LOOP BACK
1619 012142  001361                             BNE     1$              ;BRANCH BACK IF SO
1620 012144  012777  177777  170350  4$:       MOV     #-1,@BAR        ;ALL ONES TO BAR
1621 012152  017737  170344  002570             MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
1622 012160  022737  177776  002570             CMP     #-2,RBAR        ;SEE IF ALL BITS WERE SET (DON'T EXPECT BIT 0 TO SET)
1623 012166  001426                             BEQ     8$              ;BRANCH IF OK
1624 012170  012737  012206  001310             MOV     #5$,$LPERR      ;MOVE NEW LOOP ON ERROR LOCATION TO $LPERR
1625 012176  012737  177776  002602             MOV     #-2,EBAR        ;MOVE EXPECTED DATA TO EBAR
1626 012204  000412                             BR      6$              ;BRANCH OVER LOOP SETUP
1627 012206  012777  177777  170306  5$:       MOV     #-1,@BAR        ;ALL ONES TO BAR
1628 012214  017737  170302  002570             MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
1629 012222  022737  177776  002570             CMP     #-2,RBAR        ;SEE IF ALL BITS WERE SET (DON'T EXPECT BIT 0 TO SET)
1630 012230  001401                             BEQ     7$              ;BRANCH IF OK
1631 012232  104012                     6$:     ERROR   +12             ;ATTEMPT TO SET ALL BAR BITS TO 1 FAILED
1632 012234  032777  001000  167076  7$:       BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
1633 012242  001361                             BNE     5$              ;BRANCH BACK IF SO
1634 012244  017737  170254  002562  8$:       MOV     @CSR,RCSR       ;ACCESS CSR TO SET BIT 0 OF BAR
1635 012252  012777  177777  170242             MOV     #-1,@BAR        ;ALL ONES TO BAR
1636 012260  017737  170236  002570             MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
1637 012266  022737  177777  002570             CMP     #-1,RBAR        ;SEE IF ALL BITS WERE SET (*DO* EXPECT BIT 0 TO SET)
1638 012274  001431                             BEQ     12$             ;BRANCH IF OK
1639 012276  012737  012314  001310             MOV     #9$,$LPERR      ;MOVE NEW LOOP ON ERROR LOCATION TO $LPERR
1640 012304  012737  177777  002602             MOV     #-1,EBAR        ;MOVE EXPECTED DATA TO EBAR
1641 012312  000415                             BR      10$             ;BRANCH OVER LOOP SETUP
1642 012314  017737  170204  002562  9$:       MOV     @CSR,RCSR       ;ACCESS CSR TO SET BIT 0 OF BAR
1643 012322  012777  177777  170172             MOV     #-1,@BAR        ;ALL ONES TO BAR
1644 012330  017737  170166  002570             MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
1645 012336  022737  177777  002570             CMP     #-1,RBAR        ;SEE IF ALL BITS WERE SET (*DO* EXPECT BIT 0 TO SET)
1646 012344  001401                             BEQ     11$             ;BRANCH IF OK
1647 012346  104012                     10$:    ERROR   +12             ;ATTEMPT TO SET ALL BAR BITS TO 1 FAILED
1648 012350  032777  001000  166762  11$:      BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
1649 012356  001356                             BNE     9$              ;BRANCH BACK IF SO
1650 012360  012777  177777  170140  12$:      MOV     #-1,@BDR        ;ALL ONES TO BDR
```

```
1651 012366  017737  170134  002566          MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
1652 012374  022737  177777  002566          CMP     #-1,RBDR        ;SEE IF DATA WAS LOADED PROPERLY
1653 012402  001423                          BEQ     16$             ;BRANCH IF OK
1654 012404  012737  012414  001310          MOV     #13$,$LPERR     ;MOVE NEW LOOP ON ERROR LOCATION TO $LPERR
1655 012412  000412                          BR      14$             ;BRANCH OVER LOOP SETUP
1656 012414  012777  177777  170104  13$:    MOV     #-1,@BDR        ;ALL ONES TO BDR
1657 012422  017737  170100  002566          MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
1658 012430  022737  177777  002566          CMP     #-1,RBDR        ;SEE IF DATA WAS LOADED PROPERLY
1659 012436  001401                          BEQ     15$             ;BRANCH IF OK
1660 012440  104027                  14$:    ERROR   +27             ;ALL BDR BITS ARE NOT SET
1661 012442  032777  001000  166670  15$:    BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
1662 012450  001361                          BNE     13$             ;BRANCH BACK IF SO
1663 012452  012777  010576  170044  16$:    MOV     #10576,@CSR     ;SET ALL CSR WRITEABLE BITS
1664 012460  000005                          RESET                   ;RESET THE WORLD OF ITS TROUBLES - HOPEFULLY
1665 012462  012737  012040  001310          MOV     #999$,$LPERR    ;RESET THE LOOP ON ERROR LOCATION
1666 012470  017737  170024  002572          MOV     @WCR,RWCR       ;WAS WCR CLEARED?
1667 012476  001401                          BEQ     17$             ;BRANCH IF WCR WAS CLEARED
1668 012500  104007                          ERROR   +7              ;RESET FAILED TO CLEAR WCR
1669 012502  017737  170014  002570  17$:    MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
1670 012510  001403                          BEQ     18$             ;BRANCH IF BAR WAS CLEARED
1671 012512  005037  002602                  CLR     EBAR            ;CLEAR EXPECTED LOCATION
1672 012516  104011                          ERROR   +11             ;RESET FAILED TO CLEAR BAR
1673 012520  017737  170000  002562  18$:    MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
1674 012526  012737  000200  002574          MOV     #RY,ECSR        ;MOVE EXPECTED DATA TO ECSR
1675 012534  004737  004060                  JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
1676 012540  023737  002574  002562          CMP     ECSR,RCSR       ;SEE IF EXPECTED DATA WAS RECEIVED
1677 012546  001401                          BEQ     19$             ;BRANCH IF IT WAS
1678 012550  104032                          ERROR   +32             ;READY IS NOT THE ONLY BIT SET
1679 012552  052777  010000  167744  19$:    BIS     #MA,@CSR        ;MAINT MODE (SO THAT IDR GETS ODR CONTENTS)
1680 012560  017737  167742  002566          MOV     @BDR,RBDR       ;MOVE CONTENTS OF BDR TO RBDR
1681 012566  001401                          BEQ     TST5            ;BRANCH IF IT CORRECTLY REMAINS ZERO
1682 012570  104026                          ERROR   +26             ;BDR IS NOT CLEAR
```

```
1688                                    .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.
                                        ;*
                                        ;**********************************************************************
    012572  000004                      TST5:   SCOPE
1689 012574  005077  167724             999$:   CLR     @CSR            ;FORCE ACCESS TO CSR
1690 012600  012777  177777  167712             MOV     #-1,@WCR        ;ALL ONES TO WCR
1691 012606  012777  177777  167712             MOV     #-1,@BDR        ;ALL ONES TO BDR
1692 012614  012777  177777  167700             MOV     #-1,@BAR        ;ALL ONES TO BAR
1693 012622  012777  010576  167674             MOV     #10576,@CSR     ;SET ALL WRITEABLE BITS IN THE CSR
1694 012630  042777  010000  167666             BIC     #MA,@CSR        ;CLEAR THE MAINT BIT TO DO AN INIT
1695 012636  017737  167656  002572             MOV     @WCR,RWCR       ;MOVE RECEIVED CONTENTS TO RWCR
1696 012644  001401                             BEQ     1$              ;BRANCH IF WCR WAS CLEARED
1697 012646  104003                             ERROR   +3              ;INIT FAILED TO CLEAR WCR
1698 012650  017737  167646  002570  1$:        MOV     @BAR,RBAR       ;MOVE RECEIVED CONTENTS TO RBAR
1699 012656  001403                             BEQ     2$              ;BRANCH IF BAR WAS CLEARED
1700 012660  005037  002602                     CLR     EBAR            ;CLEAR EXPECTED LOCATION
1701 012664  104004                             ERROR   +4              ;INIT FAILED TO CLEAR BAR
1702 012666  017737  167632  002562  2$:        MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
1703 012674  012737  000200  002574             MOV     #RY,ECSR        ;EXPECT READY BIT ONLY TO BE SET
1704 012702  004737  004060                     JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
1705 012706  023737  002574  002562             CMP     ECSR,RCSR       ;SEE IF EXPECTED DATA WAS RECEIVED
1706 012714  001401                             BEQ     3$              ;BRANCH IF THEY WERE ALL CLEAR
1707 012716  104006                             ERROR   +6              ;INIT FAILED TO CLEAR ALL CSR R-W BITS
1708 012720  012777  010000  167576  3$:        MOV     #MA,@CSR        ;GO BACK INTO MAINT MODE (SO THAT IDR GETS ODR CONTENTS)
1709 012726  017737  167574  002566             MOV     @BDR,RBDR       ;MOVE RECEIVED CONTENTS TO RBDR
1710 012734  001401                             BEQ     TST6            ;BRANCH IF IT WAS CLEARED
1711 012736  104005                             ERROR   +5              ;INIT FAILED TO CLEAR BDR
```

```
1721                                    .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 $LPERR IS INITIALIZED TO A ROUTINE SPECIFI-
                                        ;*      CALLY WRITTEN FOR THAT PARTICULAR ERROR TO CREATE A VERY TIGHT LOOP.
                                        ;*
                                        ;*****************************************************************
     012740  000004                     TST6:   SCOPE
1722 012742  004737  004036                     JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
1723 012746  005037  002716                     CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +204
1724 012752  012701  006250                     MOV     #PATRNS,R1      ;MOVE ADDRESS OF BIT PATTERNS TO R1
1725 012756  012702  000007                     MOV     #7,R2           ;DO 7 PATTERNS
1726 012762  012777  010000  167534             MOV     #MA,@CSR        ;GO TO MAINTENANCE MODE
1727 012770  011177  167524             1$:     MOV     (R1),@WCR       ;MOVE THE DATA TO WCR
1728 012774  017737  167520  002572             MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
1729 013002  021137  002572                     CMP     (R1),RWCR       ;SEE IF EXPECTED DATA WAS RECEIVED
1730 013006  001423                             BEQ     5$              ;BRANCH IF SO
1731 013010  012737  013024  001310             MOV     #2$,$LPERR      ;SET UP LOOP ON ERROR LOCATION
1732 013016  011137  002604                     MOV     (R1),EWCR       ;MOVE EXPECTED DATA TO EWCR
1733 013022  000410                             BR      3$              ;SKIP OVER LOOP ON ERROR SETUP
1734 013024  011177  167470             2$:     MOV     (R1),@WCR       ;LOAD BIT PATTERN TO WCR
1735 013030  017737  167464  002572             MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
1736 013036  021137  002572                     CMP     (R1),RWCR       ;SEE IF DATA IS OK NOW
1737 013042  001401                             BEQ     4$              ;BRANCH OUT IF SO - OK NOW
1738 013044  104204                     3$:     ERROR   +204            ;WCR DATA PATTERN NOT CORRECT
1739 013046  032777  001000  166264     4$:     BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
1740 013054  001363                             BNE     2$              ;BRANCH BACK IF SO
1741 013056  011177  167440             5$:     MOV     (R1),@BAR       ;MOVE THE DATA TO BAR
1742 013062  017737  167434  002570             MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
1743 013070  011137  002602                     MOV     (R1),EBAR       ;MOVE EXPECTED DATA TO EBAR
1744 013074  042737  000001  002602             BIC     #BIT0,EBAR      ;DO NOT EXPECT BIT 0 TO BE READ
1745 013102  023737  002602  002570             CMP     EBAR,RBAR       ;SEE IF EXPECTED DATA WAS RECEIVED
1746 013110  001423                             BEQ     9$              ;BRANCH IF SO
1747 013112  012737  013126  001310             MOV     #6$,$LPERR      ;SET UP LOOP ON ERROR LOCATION
1748 013120  011137  002602                     MOV     (R1),EBAR       ;MOVE EXPECTED DATA TO EBAR
1749 013124  000410                             BR      7$              ;SKIP OVER LOOP ON ERROR SETUP
1750 013126  011177  167370             6$:     MOV     (R1),@BAR       ;LOAD BIT PATTERN TO BAR
1751 013132  017737  167364  002570             MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
1752 013140  021137  002570                     CMP     (R1),RBAR       ;SEE IF DATA IS OK NOW
1753 013144  001401                             BEQ     8$              ;BRANCH OUT IF SO - OK NOW
1754 013146  104203                     7$:     ERROR   +203            ;BAR DATA PATTERN NOT CORRECT
1755 013150  032777  001000  166162     8$:     BIT     #BIT9,@SWR      ;SEE IF WE SHOULD LOOP BACK
1756 013156  001363                             BNE     6$              ;BRANCH BACK IF SO
1757 013160  011177  167342             9$:     MOV     (R1),@BDR       ;MOVE THE DATA TO BDR
1758 013164  017737  167336  002566             MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
1759 013172  021137  002566                     CMP     (R1),RBDR       ;SEE IF EXPECTED DATA WAS RECEIVED
1760 013176  001423                             BEQ     13$             ;BRANCH IF SO
1761 013200  012737  013214  001310             MOV     #10$,$LPERR     ;SET UP LOOP ON ERROR LOCATION
1762 013206  011137  002600                     MOV     (R1),EBDR       ;MOVE EXPECTED DATA TO EBDR
1763 013212  000410                             BR      11$             ;SKIP OVER LOOP ON ERROR SETUP
1764 013214  011177  167306             10$:    MOV     (R1),@BDR       ;LOAD BIT PATTERN TO BDR
1765 013220  017737  167302  002566             MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
1766 013226  021137  002566                     CMP     (R1),RBDR       ;SEE IF DATA IS OK NOW
```

```
1767 013232  001401                        BEQ     12$             ;BRANCH OUT IF SO - OK NOW
1768 013234  104205                 11$:    ERROR   +205            ;BDR PATTERN NOT CORRECT
1769 013236  032777  001000  166074  12$:    BIT     #BIT9,@SWR     ;SEE IF WE SHOULD LOOP BACK
1770 013244  001363                        BNE     10$             ;BRANCH BACK IF SO
1771 013246  005721                 13$:    TST     (R1)+           ;GO TO NEXT PATTERN
1772 013250  005302                         DEC     R2              ;DECREMENT THE LOOP COUNTER AND
1773 013252  001246                         BNE     1$              ;BRANCH BACK IF NOT DONE
```

```
                                .SBTTL  TEST #7 - TEST CSR AND EIR BIT0
                                ;*********************************************************************
                                ;*TEST 7         TEST CSR AND EIR BIT0
                                ;*
                                ;*      THIS TEST INSURES THAT BIT 0 OF THE CSR IS CLEAR WHEN IN CSR MODE (BIT
                                ;*      15 CLEAR), AND SET WHEN IN EIR MODE (BIT 15 SET).
                                ;*
                                ;*********************************************************************
     013254                     TST7:
     013254  000004                     SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     013256  012737  013264  001310     MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
1781 013264  032737  000001  002612 999$:  BIT     #BIT0,BORW      ;TEST TO SEE IF WE ARE TESTING A DR11-W
1782 013272  001444                     BEQ     TST10           ;BRANCH TO NEXT TEST IF A DR11-B
1783 013274  005077  167224             CLR     @CSR            ;FORCE ACCESS TO CSR
1784 013300  012737  000001  002540     MOV     #BIT0,BUT       ;MOVE BIT 0 INDICATOR TO BIT UNDER TEST LOCATION
1785 013306  017737  167212  002562     MOV     @CSR,RCSR       ;MOVE CSR CONTENTS TO RCSR
1786 013314  032737  000001  002562     BIT     #BIT0,RCSR      ;CLEAR ALL BUT BIT 0
1787 013322  001407                     BEQ     1$              ;BRANCH IF A ZERO
1788 013324  013737  002562  002574     MOV     RCSR,ECSR       ;MOVE CSR TO EXPECTED DATA, ECSR AND
1789 013332  042737  000001  002574     BIC     #BIT0,ECSR      ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
1790 013340  104014                     ERROR   +14             ;CSR BIT TEST FAILED
1791 013342  012777  100000  167154 1$:  MOV     #EIR,@CSR       ;GO TO EIR MODE
1792 013350  017737  167150  002564     MOV     @CSR,REIR       ;MOVE CSR CONTENTS TO RCSR
1793 013356  032737  000001  002564     BIT     #BIT0,REIR      ;CLEAR ALL BUT BIT 0
1794 013364  001007                     BNE     TST10           ;BRANCH IF NOT A ZERO
1795 013366  013737  002564  002576     MOV     REIR,EEIR       ;MOVE CONTENTS TO ECSR ALSO AND
1796 013374  052737  000001  002576     BIS     #BIT0,EEIR      ;SET THE 0 BIT - EXPECTED IT TO BE 1
1797 013402  104015                     ERROR   +15             ;EIR BIT TEST FAILED
```

```
1804                                    .SBTTL  TEST #10 - ATTN CAN BE SET VIA FNCT2 & ERRCR 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.
                                        ;*
                                        ;*********************************************************************
     013404                             TST10:
     013404  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     013406  012737  013424  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
1805 013414  032737  000001  002720             BIT     #BIT0,DDW       ;TEST TO SEE IF WE ARE TESTING A DR11-W
1806 013422  001073                             BNE     TST11           ;BRANCH TO NEXT TEST IF A DR11-B
1807 013424  005077  167074         999$:       CLR     @CSR            ;FORCE ACCESS TO CSR
1808 013430  012777  010000  167066             MOV     #MA,@CSR        ;MAINT
1809 013436  017737  167062  002562             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
1810 013444  022737  010200  002562             CMP     #MA+RY,RCSR     ;SEE IF EXPECTED DATA WAS RECEIVED
1811 013452  001404                             BEQ     1$              ;BRANCH IF THEY ARE
1812 013454  012737  010200  002574             MOV     #MA+RY,ECSR     ;MOVE EXPECTED DATA TO ECSR
1813 013462  104016                             ERROR   +16             ;READY AND MAINTENANCE ARE NOT THE ONLY BITS SET IN CSR
1814 013464  112777  000004  167032  1$:        MOVB    #F2,@CSR        ;SET FNCT2
1815 013472  017737  167026  002562             MOV     @CSR,RCSR       ;MOVE THE CONTENTS TO RCSR
1816 013500  013701  002562                     MOV     RCSR,R1         ;MOVE CONTENTS TO R1 FOR BIT TEST
1817 013504  042701  057777                     BIC     #CB1513,R1      ;CLEAR ALL BUT BITS ERROR & ATTN FOR TEST
1818 013510  022701  120000                     CMP     #ER+AT,R1       ;TEST TO SEE IF ERROR AND ATTN ARE SET
1819 013514  001411                             BEQ     2$              ;BRANCH IF IT IS PROPERLY SET
1820 013516  013737  002562  002574             MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
1821 013524  004737  004060                     JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
1822 013530  052737  120000  002574             BIS     #ER+AT,ECSR     ;SET THE BITS THAT SHOULD HAVE BEEN SET
1823 013536  104017                             ERROR   +17             ;ATTN AND ERROR FAILED TO SET PROPERLY
1824 013540  042777  020004  166756  2$:        BIC     #AT+F2,@CSR     ;CLEAR ATTN & FNCT2
1825 013546  017737  166752  002562             MOV     @CSR,RCSR       ;MOVE CSR DATA TO RCSR
1826 013554  032737  120000  002562             BIT     #ER+AT,RCSR     ;BIT TEST ATTN AND ERROR BITS TO SEE IF THEY ARE CLEAR
1827 013562  001411                             BEQ     3$              ;BRANCH IF ATTN IS CLEAR
1828 013564  013737  002562  002574             MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
1829 013572  042737  120000  002574             BIC     #ER+AT,ECSR     ;CLEAR THE BITS THAT WERE SUPPOSED TO BE CLEAR
1830 013600  004737  004060                     JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
1831 013604  104020                             ERROR   +20             ;ATTN AND ERROR FAILED TO CLEAR PROPERLY
1832 013606  005077  166712         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.
                                        ;*
                                        ;*************************************************************
     013612                             TST11:
     013612  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     013614  012737  013622  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
1839 013622  005077  166676             999$:   CLR     @CSR            ;CLR FUNCT BITS AND FORCE ACCESS TO CSR
1840 013626  012777  010000  166670             MOV     #MA,@CSR        ;MAINT MODE
1841 013634  017737  166664  002562             MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
1842 013642  013737  002562  002574             MOV     RCSR,ECSR       ;MOVE EXPECTED TO ECSR
1843 013650  013701  002574                     MOV     ECSR,R1         ;MOVE CONTENTS TO R1 FOR TESTING
1844 013654  042701  177761                     BIC     #CFNC,R1        ;CLEAR ALL BUT THE FNCT BITS
1845 013660  001404                             BEQ     1$              ;BRANCH IF THE FUNCTION BITS ARE CLEAR
1846 013662  042737  000016  002574             BIC     #FNC,ECSR       ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
1847 013670  104024                             ERROR   +24             ;FUNCTION BIT(S) ARE NOT CLEAR
1848 013672  052777  000002  166624  1$:        BIS     #F1,@CSR        ;SET FNCT1
1849 013700  017737  166620  002562             MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
1850 013706  013737  002562  002574             MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
1851 013714  013701  002574                     MOV     ECSR,R1         ;MOVE CONTENTS TO R1 FOR TEST
1852 013720  042701  170777                     BIC     #CDST,R1        ;CLEAR ALL BUT BITS 9, 10 & 11
1853 013724  022701  001000                     CMP     #DSC,R1         ;SEE IF DSTAT A AND B ARE CLEAR & C IS SET
1854 013730  001407                             BEQ     TST12           ;BRANCH TO NEXT TEST IF ALL CLEAR
1855 013732  042737  006000  002574             BIC     #DAB,ECSR       ;CLEAR THE DSTAT A & B BITS THAT SHOULD HAVE BEEN CLEAR
1856 013740  052737  001000  002574             BIS     #DSC,ECSR       ;SET THE DSTAT C BIT THAT SHOULD HAVE BEEN SET
1857 013746  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.
                                ;*
                                ;*********************************************************************
     013750                     TST12:
     013750  000004                     SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     013752  012737  013760  001310     MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
1864 013760  005077  166540     999$:  CLR     @CSR            ;CLR FUNCT BITS AND FORCE ACCESS TO CSR
1865 013764  012777  010000  166532     MOV     #MA,@CSR        ;MAINT MODE
1866 013772  017737  166526  002562     MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
1867 014000  013737  002562  002574     MOV     RCSR,ECSR       ;MOVE EXPECTED TO ECSR
1868 014006  013701  002574             MOV     ECSR,R1         ;MOVE CONTENTS TO R1 FOR TESTING
1869 014012  042701  177761             BIC     #CFNC,R1        ;CLEAR ALL BUT THE FNCT BITS
1870 014016  001404                     BEQ     1$              ;BRANCH IF THE FUNCTION BITS ARE CLEAR
1871 014020  042737  000016  002574     BIC     #FNC,ECSR       ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
1872 014026  104024                     ERROR   +24             ;FUNCTION BIT(S) ARE NOT CLEAR
1873 014030  052777  000004  166466  1$:BIS     #F2,@CSR        ;SET FNCT2
1874 014036  017737  166462  002562     MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
1875 014044  013737  002562  002574     MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
1876 014052  013701  002574             MOV     ECSR,R1         ;MOVE CONTENTS TO R1 FOR TEST
1877 014056  042701  170777             BIC     #CDST,R1        ;CLEAR ALL BUT THE DSTAT BITS
1878 014062  022701  002000             CMP     #DSB,R1         ;IF DSTAT A AND C ARE CLEAR & B IS SET
1879 014066  001407                     BEQ     TST13           ;BRANCH TO NEXT TEST IF AS EXPECTED
1880 014070  042737  005000  002574     BIC     #DAC,ECSR       ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
1881 014076  052737  002000  002574     BIS     #DSB,ECSR       ;SET THE BIT THAT SHOULD HAVE BEEN SET
1882 014104  104025                     ERROR   +25             ;DSTAT A, B OR C ARE NOT AS EXPECTED
```

```
1888                                    .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.
                                        ;*
                                        ;*********************************************************************
     014106                             TST13:
     014106  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     014110  012737  014116  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
1889 014116  005077  166402             999$:   CLR     @CSR            ;CLR FUNCT BITS AND FORCE ACCESS TO CSR
1890 014122  012777  010000  166374             MOV     #MA,@CSR        ;MAINT MODE
1891 014130  017737  166370  002562             MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
1892 014136  013737  002562  002574             MOV     RCSR,ECSR       ;MOVE EXPECTED TO ECSR
1893 014144  013701  002574                     MOV     ECSR,R1         ;MOVE CONTENTS TO R1 FOR TESTING
1894 014150  042701  177761                     BIC     #CFNC,R1        ;CLEAR ALL BUT THE FNCT BITS
1895 014154  001404                             BEQ     1$              ;BRANCH IF THE FUNCTION BITS ARE CLEAR
1896 014156  042737  000016  002574             BIC     #FNC,ECSR       ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
1897 014164  104024                             ERROR   +24             ;FUNCTION BIT(S) ARE NOT CLEAR
1898 014166  052777  000010  166330  1$:       BIS     #F3,@CSR        ;SET FNCT3
1899 014174  017737  166324  002562             MOV     @CSR,RCSR       ;MOVE CONTENTS OF CSR TO RCSR
1900 014202  013737  002562  002574             MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
1901 014210  013701  002574                     MOV     FCSR,R1         ;MOVE CONTENTS TO R1 FOR TEST
1902 014214  042701  170777                     BIC     #CDST,R1        ;CLEAR ALL SUT DSTAT BITS
1903 014220  022701  004000                     CMP     #DSA,R1         ;SEE IF DSTAT B AND C ARE CLEAR & A IS SET
1904 014224  001407                             BEQ     TST14           ;BRANCH TO NEXT TEST IF DATA OK
1905 014226  042737  003000  002574             BIC     #DBC,ECSR       ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
1906 014234  052737  004000  002574             BIS     #DSA,ECSR       ;SET THE BIT THAT SHOULD HAVE BEEN SET
1907 014242  104025                             ERROR   +25             ;DSTAT A, B OR C ARE NOT AS EXPECTED
```

```
1915                                    .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).
                                        ;*
                                        ;**************************************************************
     014244                             TST14:
     014244  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     014246  012737  014264  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
1916 014254  032737  000001  002612             BIT     #BIT0,BORW      ;TEST TO SEE IF WE ARE TESTING A DR11-W
1917 014262  001451                             BEQ     TST15           ;BRANCH TO NEXT TEST IF A DR11-B
1918 014264  005077  166234         999$:       CLR     @CSR            ;FORCE ACCESS TO CSR
1919 014270  012777  010000  166226             MOV     #MA,@CSR        ;SET MAINT MODE (SO THAT IDR GETS ODR CONTENTS)
1920 014276  012777  052525  166222             MOV     #52525,@BDR     ;SET ALT 0'S AND 1'S TO BDR
1921 014304  017737  166216  002566             MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
1922 014312  022737  052525  002566             CMP     #52525,RBDR     ;SEE IF DATA WAS LOADED PROPERLY
1923 014320  001404                             BEQ     1$              ;BRANCH IF IT WAS
1924 014322  012737  052525  002600             MOV     #52525,EBDR     ;MOVE EXPECTED DATA TO EBDR
1925 014330  104031                             ERROR   +31             ;BDR PATTERN NOT CORRECT
1926 014332  052777  100000  166164  1$:        BIS     #EIR,@CSR       ;GO TO EIR
1927 014340  012737  052525  002600             MOV     #52525,EBDR     ;MOVE EXPECTED DATA TO EBDR
1928 014346  012777  125252  166152             MOV     #125252,@BDR    ;SET ALT 1'S AND 0'S TO BDR
1929 014354  017737  166146  002566             MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
1930 014362  022737  052525  002566             CMP     #52525,RBDR     ;TEST FOR OLD PATTERN
1931 014370  001404                             BEQ     2$              ;BRANCH IF ORIGINAL PATTERN STILL THERE
1932 014372  012737  052525  002600             MOV     #52525,EBDR     ;MOVE EXPECTED DATA TO EBDR
1933 014400  104030                             ERROR   +30             ;BDR SHOULD NOT HAVE BEEN LOADED WITH NEW PATTERN
1934 014402  004737  004036         2$:         JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
```

```
1941                                    .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.
                                        ;*
                                        ;**********************************************************************
     014406                             TST15:
     014406  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     014410  012737  014416  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
1942 014416  012777  010000  166100 999$:      MOV     #MA,@CSR        ;SET MAINTENANCE BIT AND
1943 014424  005077  166074                    CLR     @CSR            ;CLEAR CSR TO DO AN INIT
1944 014430  012737  000140  177776             MOV     #LEVEL3,PSW     ;STATUS AT LEVEL 3
1945 014436  017737  166062  002562             MOV     @CSR,RCSR       ;MOVE CSR CONTENTS TO RCSR
1946 014444  105737  002562                    TSTB    RCSR            ;SEE IF READY BIT (BIT 7) IS SET
1947 014450  100406                            BMI     1$              ;BRANCH IF IT IS
1948 014452  012737  000200  002574             MOV     #RY,ECSR        ;SET THE BIT THAT SHOULD HAVE BEEN SET
1949 014460  004737  004060                    JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
1950 014464  104022                            ERROR   +22             ;READY OF CSR WAS NOT SET
1951 014466  017737  166036  002534 1$:       MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
1952 014474  017737  166032  002536             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
1953 014502  012777  014572  166020             MOV     #3$,@DRINV      ;SET UP INTERRUPT VECTOR
1954 014510  012777  010000  166006             MOV     #MA,@CSR        ;MAINT MODE
1955 014516  012737  001000  002662             MOV     #1000,TIME      ;SET THE TIME COUNTER
1956 014524  052777  000105  165772             BIS     #IE+F2+GO,@CSR  ;IE, FNCT2 AND GO
1957 014532  005337  002662         2$:         DEC     TIME            ;DECREMENT DOWN TO ZERO
1958 014536  001375                            BNE     2$              ;BRANCH IF NOT THERE YET
1959 014540  017737  165760  002562             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
1960 014546  013777  002534  165754             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
1961 014554  013777  002536  165750             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
1962 014562  104035                            ERROR   +35             ;DR11 FAILED TO INTERRUPT
1963 014564  005077  165734                    CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
1964 014570  000410                            BR      TST16           ;BRANCH TO THE NEXT TEST
1965 014572  062706  000004         3$:         ADD     #4,SP           ;CLEAN THE STACK AFTER THE INTERRUPT
1966 014576  013777  002534  165724             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
1967 014604  013777  002536  165720             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
```

```
1974                                    .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.
                                        ;*
                                        ;*********************************************************************
     014612                             TST16:
     014612  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     014614  012737  014622  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
1975 014622  004737  004036             999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
1976 014626  017737  165672  002562             MOV     @CSR,RCSR       ;MOVE CSR DATA TO RCSR
1977 014634  105737  002562                     TSTB    RCSR            ;CLEAR ALL BUT THE READY BIT (BIT 7)
1978 014640  100411                             BMI     1$              ;BRANCH IF IT IS SET
1979 014642  012737  000200  002574             MOV     #RY,ECSR        ;MOVE EXPECTED DATA TO ECSR
1980 014650  004737  004060                     JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
1981 014654  052737  000200  002574             BIS     #RY,ECSR        ;SET THE BIT THAT SHOULD HAVE BEEN SET
1982 014662  104022                             ERROR   +22             ;READY OF CSR WAS NOT SET
1983 014664  017737  165640  002534     1$:     MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
1984 014672  017737  165634  002536             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
1985 014700  012777  014766  165622             MOV     #3$,@DRINV      ;SET UP INT VECTOR
1986 014706  012777  000340  165616             MOV     #LEVEL7,@DRVS
1987 014714  012737  001000  002662             MOV     #1000,TIME      ;SET TIME DELAY COUNTER
1988 014722  012777  010000  165574             MOV     #MA,@CSR        ;MAINT MODE
1989 014730  052777  000105  165566             BIS     #IE+F2+GO,@CSR  ;IE, FNCT2 AND GO
1990 014736  005337  002662             2$:     DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
1991 014742  001375                             BNE     2$              ;BRANCH BACK IF NOT ZERO YET
1992 014744  005077  165554                     CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
1993 014750  013777  002534  165552             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
1994 014756  013777  002536  165546             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
1995 014764  000416                             BR      TST17           ;BRANCH TO THE NEXT TEST
1996 014766  062706  000004             3$:     ADD     #4,SP           ;RESTORE STACK
1997 014772  017737  165526  002562             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
1998 015000  005077  165520                     CLR     @CSR            ;CLEAR IE
1999 015004  013777  002534  165516             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2000 015012  013777  002536  165512             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2001 015020  104036                             ERROR   +36             ;DR11 INTERRUPTED, BUT IT SHOULDN'T HAVE
```

```
                                .SBTTL  TEST #17 - DR11 INTERRUPTS AT CORRECT BR LEVEL
                                ;*********************************************************************
                                ;*TEST 17       DR11 INTERRUPTS AT CORRECT BR LEVEL
                                ;*
                                ;*      THIS TEST INSURES THAT THE DR11 WILL INTERRUPT AT THE CORRECT LEVEL AS
                                ;*      DEFINED IN THE DEVICE DESCRIPTOR WORD.
                                ;*
                                ;*********************************************************************
     015022                     TST17:
     015022  000004                     SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     015024  012737  015106  001310     MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2009 015032  004737  004036             JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2010 015036  013737  002720  001362     MOV     DDW,$TMP1       ;MOVE DEVICE DESCRIPTOR WORD TO $TMP1
2011 015044  006237  001362             ASR     $TMP1           ;SHIFT THE LEVEL TO THE RIGHT 5 PLACES
2012 015050  006237  001362             ASR     $TMP1
2013 015054  006237  001362             ASR     $TMP1
2014 015060  006237  001362             ASR     $TMP1
2015 015064  006237  001362             ASR     $TMP1
2016 015070  042737  177770  001362     BIC     #177770,$TMP1   ;CLEAR ALL BUT THE PRIORITY
2017 015076  012700  000003         1$: MOV     #3,R0           ;DO 3 PRIORITY LEVELS
2018 015102  012701  005264             MOV     #LEVELS,R1      ;MOVE ADDRESS OF CPU PRIORITIES TO R1
2019 015106  012777  010000  165410 999$: MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
2020 015114  005077  165404             CLR     @CSR            ;CLEAR TO DO AN INIT
2021 015120  011137  177776             MOV     (R1),PSW        ;PUT PRIORITY INTO PSW
2022 015124  017737  165374  002562     MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2023 015132  012737  000200  002574     MOV     #RY,ECSR        ;MOVE READY BIT TO ECSR
2024 015140  004737  004060             JSR     PC,CHKCAB       ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
2025 015144  023737  002562  002574     CMP     RCSR,ECSR       ;SEE IF RECEIVED DATA MATCHES EXPECTED
2026 015152  001412                     BEQ     2$              ;BRANCH IF OK
2027 015154  012737  015076  001310     MOV     #1$,$LPERR      ;SET UP FOR POSSIBLE LOOP ON ERROR FOR THIS ERROR ONLY
2028 015162  012737  000200  002574     MOV     #RY,ECSR        ;MOVE EXPECTED DATA TO ECSR
2029 015170  104022                     ERROR   +22             ;READY OF CSR WAS NOT SET
2030 015172  012737  015106  001310     MOV     #999$,$LPERR    ;RETURN ORIGINAL LOOP ON ERROR ADDRESS - DID NOT LOOP
2031 015200  017737  165324  002534 2$: MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2032 015206  017737  165320  002536     MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2033 015214  012777  015326  165306     MOV     #4$,@DRINV      ;SET UP INTERRUPT VECTOR
2034 015222  012777  000340  165302     MOV     #LEVEL7,@DRVS   ;SET UP INTERRUPT PS
2035 015230  012777  010000  165266     MOV     #MA,@CSR        ;MAINT MODE
2036 015236  012737  000400  002662     MOV     #400,TIME       ;SET DELAY COUNTER
2037 015244  052777  000105  165252     BIS     #IE+F2+GO,@CSR  ;IE, FNCT2 AND GO
2038 015252  005337  002662         3$: DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
2039 015256  001375                     BNE     3$              ;BRANCH BACK IF NOT ZERO YET
2040 015260  005077  165240             CLR     @CSR            ;CLEAR CSR TO DO AN INIT
2041 015264  013777  002534  165236     MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2042 015272  013777  002536  165232     MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2043 015300  013737  177776  002542     MOV     PSW,LEVEL       ;SAVE OLD STATUS LEVEL
2044 015306  005721                     TST     (R1)+           ;INCREMENT R1 TO POINT TO NEXT PRIORITY LEVEL
2045 015310  005300                     DEC     R0              ;DECREMENT LOOP COUNTER AND
2046 015312  001275                     BNE     999$            ;BRANCH BACK FOR ANOTHER TRY IF NOT DONE
2047 015314  104053                     ERROR   +53             ;DR11 FAILED TO INTERRUPT
2048 015316  013737  002554  002542     MOV     DRLEV,LEVEL     ;SET LEVEL TO CONTAIN THE ANTICIPATED LEVEL
2049 015324  000422                     BR      TST20           ;BRANCH TO THE NEXT TEST
2050 015326  062706  000004         4$: ADD     #4,SP           ;RESTORE STACK
2051 015332  005077  165166             CLR     @CSR            ;CLEAR IE
2052 015336  013777  002534  165164     MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2053 015344  013777  002536  165160     MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2054 015352  042737  177437  002542     BIC     #177437,LEVEL   ;CLEAR ALL BITS BUT THE BR LEVEL
```

```
   2055 015360  023737  002542  002554          CMP     LEVEL,DRLEV     ;SEE IF LEVEL INTERRUPTED MATCHES EXPECTED
   2056 015366  001401                          BEQ     TST20           ;BRANCH AROUND ERROR CALL IF IT IS AS EXPECTED
   2057 015370  104052                          ERROR   +52             ;DR11 INTERRUPTED AT WRONG LEVEL
```

```
                                .SBTTL  TEST #20 - A GO WITHOUT CLEARING ERROR CAUSES INTRPT
                                ;*************************************************************
                                ;*TEST 20       A GO WITHOUT CLEARING ERROR CAUSES INTRPT
                                ;*
                                ;*      THIS TEST INSURES THAT SETTING THE GO BIT WITHOUT CLEARING THE ERROR
                                ;*      BIT CAUSES AN INTERRUPT.
                                ;*
                                ;*************************************************************
     015372                     TST20:
     015372  000004                     SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     015374  012737  015402  001310     MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2065 015402  004737  004036     999$:  JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2066 015406  017737  165116  002534     MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2067 015414  017737  165112  002536     MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2068 015422  012777  015524  165100     MOV     #2$,@DRINV      ;INTERRUPT VECTOR TO 3$
2069 015430  012777  000140  165074     MOV     #LEVEL3,@DRVS   ;INTERRUPT STATUS TO LEVEL 3
2070 015436  005037  177776             CLR     PSW             ;LET THE DR11 INTERRUPT
2071 015442  012737  001000  002662     MOV     #1000,TIME      ;MOVE DELAY COUNTER TO LOCATION
2072 015450  012777  010101  165046     MOV     #MA+IE+GO,@CSR  ;SET MAINT, IE AND GO
2073 015456  052777  020004  165040     BIS     #AT+F2,@CSR     ;SET ATTN AND FNCT2
2074 015464  005337  002662         1$: DEC     TIME            ;DECREMENT UNTIL WE REACH ZERO
2075 015470  001375                     BNE     1$              ;BRANCH IF NOT ZERO YET
2076 015472  013777  002534  165030     MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2077 015500  013777  002536  165024     MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2078 015506  017737  165012  002562     MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2079 015514  104035                     ERROR   +35             ;DR11 FAILED TO INTERRUPT
2080 015516  005077  165002             CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
2081 015522  000512                     BR      TST21           ;BRANCH TO THE NEXT TEST
2082 015524  062706  000004         2$: ADD     #4,SP           ;READJUST STACK AFTER THE INTERRUPT
2083 015530  013777  002534  164772     MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2084 015536  013777  002536  164766     MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2085 015544  017737  164754  002562     MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2086 015552  100407                     BMI     3$              ;BRANCH IF ERROR IS SET
2087 015554  013737  002562  002574     MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
2088 015562  052737  100000  002574     BIS     #ER,ECSR        ;SET THE BIT THAT SHOULD HAVE BEEN SET
2089 015570  104037                     ERROR   +37             ;ERROR BIT SHOULD NOT BE CLEAR
2090 015572  017737  164732  002534 3$: MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2091 015600  017737  164726  002536     MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2092 015606  012777  015702  164714     MOV     #5$,@DRINV      ;INTERRUPT VECTOR TO 6$
2093 015614  005077  164702             CLR     @BAR            ;PREVENT CAUSING ANOTHER ERROR
2094 015620  012777  177777  164672     MOV     #-1,@WCR        ;SET-UP WCR
2095 015626  012737  001000  002662     MOV     #1000,TIME      ;LOAD 1000 IN LOCATION TIME FOR WAIT LOOP
2096 015634  052777  000001  164662     BIS     #GO,@CSR        ;SET 'GO' IN CSR
2097 015642  005237  002662         4$: INC     TIME            ;DELAY - WAIT FOR INTERRUPT
2098 015646  001375                     BNE     4$              ;BRANCH BACK IF NOT ZERO
2099 015650  013777  002534  164652     MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2100 015656  013777  002536  164646     MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2101 015664  017737  164634  002562     MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2102 015672  104035                     ERROR   +35             ;DR11 FAILED TO INTERRUPT
2103 015674  005077  164624             CLR     @CSR            ;CLEAR CSR TO DO AN INIT
2104 015700  000423                     BR      TST21           ;BRANCH TO THE NEXT TEST
2105 015702  062706  000004         5$: ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
2106 015706  013777  002534  164614     MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2107 015714  013777  002536  164610     MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2108 015722  017737  164576  002562     MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR - IS ERROR CLEAR
2109 015730  100007                     BPL     TST21           ;BRANCH TO NEXT TEST IF IT IS
2110 015732  013737  002562  002574     MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
```

```
2111 015740  042737  100000  002574          BIC     #ER,ECSR        ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
2112 015746  104021                          ERROR   +21             ;ERROR BIT SHOULD HAVE BEEN CLEAR
```

```
2119                                    .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 MAINTE***NCE
                                        ;*      MODE TRANSFERS.
                                        ;*
                                        ;**********************************************************************
     015750                             TST21:
     015750  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     015752  012737  015774  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2120 015760  012737  000016  001364             MOV     #16,$TMP2       ;SET UP FUNCTION COUNT COMPARE
2121 015766  012737  177771  001360             MOV     #-7,$TMP0       ;SET UP WCR LOAD VARIABLE
2122 015774  004737  004036             999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2123 016000  013777  002620  164514             MOV     INBUF,@BAR      ;SET-UP BAR
2124 016006  017737  164516  002534     1$:     MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2125 016014  017737  164512  002536             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2126 016022  012777  016132  164500             MOV     #3$,@DRINV      ;INTERRUPT VECTOR
2127 016030  013777  002542  164474             MOV     LEVEL,@DRVS     ;INTERRUPT VECTOR PRIORITY TO LEVEL OF DEVICE
2128 016036  013777  001360  164454             MOV     $TMP0,@WCR      ;SET UP FOR NUMBER OF TRANSFERS IN $TMP0
2129 016044  005037  177776                     CLR     PSW             ;LET THE DR11 INTERRUPT
2130 016050  012777  010000  164446             MOV     #MA,@CSR        ;MAINT MODE
2131 016056  012737  001000  002662             MOV     #1000,TIME      ;MOVE WAIT COUNTER TO LOCATION TIME
2132 016064  052777  000501  164432             BIS     #IE+CY+GO,@CSR  ;IE, CYCLE & GO
2133 016072  005337  002662             2$:     DEC     TIME            ;DECREMENT UNTIL ZERO
2134 016076  001375                             BNE     2$              ;BRANCH BACK IF NOT
2135 016100  017737  164420  002562             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2136 016106  013777  002534  164414             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2137 016114  013777  002536  164410             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2138 016122  104035                             ERROR   +35             ;DR11 FAILED TO INTERRUPT
2139 016124  005077  164374                     CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
2140 016130  000442                             BR      TST22           ;BRANCH TO NEXT TEST
2141 016132  062706  000004             3$:     ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
2142 016136  013777  002534  164364             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2143 016144  013777  002536  164360             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2144 016152  017737  164346  002562             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2145 016160  013701  002562                     MOV     RCSR,R1         ;MOVE RECEIVED DATA TO R1 ALSO AND
2146 016164  042701  177761                     BIC     #CFNC,R1        ;CLEAR ALL BUT THE FUNCTION BITS
2147 016170  020137  001364                     CMP     R1,$TMP2        ;SEE IF FUNCTION BIT(S) HAD INCREMENTED PROPERLY
2148 016174  001412                             BEQ     4$              ;BRANCH IF THEY HAD
2149 016176  013737  002562  002574             MOV     RCSR,ECSR       ;MOVE RECEIVED DATA TO EXPECTED LOCATION
2150 016204  042737  000016  002574             BIC     #FNC,ECSR       ;CLEAR THE FUNCTION BIT(S) THAT WERE THERE AND
2151 016212  053737  001364  002574             BIS     $TMP2,ECSR      ;PUT FUNCTION BIT(S) EXPECTED IN THEIR PLACE
2152 016220  104212                             ERROR   +212            ;FUNCTION BITS DIDN'T INCREMENT IN MAINT MODE
2153 016222  005237  001360             4$:     INC     $TMP0           ;ADJUST WCR LOAD LOCATION
2154 016226  162737  000002  001364             SUB     #2,$TMP2        ;SUBTRACT 2 FROM FUNCTION COUNT TEST LOCATION
2155 016234  001264                             BNE     1$              ;BRANCH BACK FOR ANOTHER TRY
```

```
                                .SBTTL  TEST #22 - TEST FOR 10 MAINT MODE TRANSFERS
                                ;*****************************************************************
                                ;*TEST 22       TEST FOR 10 MAINT MODE TRANSFERS
                                ;*
                                ;*      THIS TEST CHECKS IF 10 MAINTENANCE MODE TRANSFERS CAN BE DONE.
                                ;*
                                ;*****************************************************************
016236                          TST22:
016236  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
016240  012737  016246  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2162 016246  005077  164252     999$:   CLR     @CSR            ;FORCE ACCESS TO CSR
2163 016252  012737  000010  002624      MOV     #10,BUFLEN      ;BUFLEN=10
2164 016260  013737  002624  002632      MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
2165 016266  005437  002632              NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
2166 016272  004737  003472              JSR     PC,LODBUF       ;LOAD IN BUFFER WITH INCREMENTING PATTERN
2167 016276  004737  003520              JSR     PC,CHKBFF       ;LOAD CHECK BUFFER WITH MODIFIED INCREMENTING PATTERN
2168 016302  013777  002632  164210      MOV     WCLEN,@WCR      ;SET UP WCR
2169 016310  013777  002620  164204      MOV     INBUF,@BAR      ;SET UP BAR
2170 016316  012777  177777  164202      MOV     #-1,@BDR        ;MAINT AIDE
2171 016324  017737  164200  002534      MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2172 016332  017737  164174  002536      MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2173 016340  012777  016442  164162      MOV     #2$,@DRINV      ;INTERRUPT VECTOR
2174 016346  013777  002542  164156      MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
2175 016354  005037  177776              CLR     PSW             ;LET DR11 INTERRUPT
2176 016360  012777  010000  164136      MOV     #MA,@CSR        ;MAINT MODE
2177 016366  052777  000501  164130      BIS     #IE+CY+GO,@CSR  ;IE, CYCLE & GO
2178 016374  012737  001000  002662      MOV     #1000,TIME      ;SET LOOP COUNTER FOR WAIT
2179 016402  005337  002662      1$:     DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
2180 016406  001375                      BNE     1$              ;BRANCH BACK IF NOT ZERO
2181 016410  013777  002534  164112      MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2182 016416  013777  002536  164106      MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2183 016424  017737  164074  002562      MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2184 016432  104035                      ERROR   +35             ;DR11 FAILED TO INTERRUPT
2185 016434  005077  164064              CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
2186 016440  000402                      BR      3$              ;BRANCH AROUND THE STACK CLEANUP
2187 016442  062706  000004      2$:     ADD     #4,SP           ;CLEAN UP STACK AFTER THE INTERRUPT
2188 016446  013777  002534  164054  3$: MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2189 016454  013777  002536  164050      MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2190 016462  004737  003546              JSR     PC,IN'4         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
2191 016466  104051                      ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
2192 016470  005037  002716              CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
2193 016474  004737  003716              JSR     PC,DATCHK       ;CHECK INBUF AFTER A MAINTENANCE MODE OPERATION
2194 016500  104201                      ERROR   +201            ;BUFFER DATA NOT CORRECT
2195 016502  004737  004014              JSR     PC,DATCH2       ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATCHK
2196
```

```
2203                                    .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.
                                        ;*
                                        ;*****************************************************************
     016506                             TST23:
     016506  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     016510  012737  016516  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2204 016516  004737  004036             999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2205 016522  012737  001000  002662             MOV     #1000,TIME      ;SET DELAY
2206 016530  012737  000010  002624             MOV     #10,BUFLEN      ;BUFLEN=10
2207 016536  013737  002624  002632             MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
2208 016544  005437  002632                     NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
2209 016550  004737  003472                     JSR     PC,LODBUF       ;LOAD IN BUFFER WITH INCREMENTING PATTERN
2210 016554  004737  003520                     JSR     PC,CHKBFF       ;LOAD CHECK BUFFER WITH MODIFIED INCREMENTING PATTERN
2211 016560  013777  002632  163732             MOV     WCLEN,@WCR      ;SET UP WCR
2212 016566  013777  002620  163726             MOV     INBUF,@BAR      ;SET UP BAR
2213 016574  012777  177777  163724             MOV     #-1,@BDR        ;MAINT AIDE
2214 016602  017737  163722  002534             MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2215 016610  017737  163716  002536             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2216 016616  012777  016716  163704             MOV     #2$,@DRINV      ;INTERRUPT VECTOR
2217 016624  013777  002542  163700             MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
2218 016632  012777  010000  163664             MOV     #MA,@CSR        ;MAINT MODE
2219 016640  052777  000501  163656             BIS     #IE+CY+GO,@CSR  ;IE, CYCLE & GO
2220 016646  005337  002662             1$:     DEC     TIME            ;WAIT FOR TRANSFERS TO COMPLETE
2221 016652  001375                             BNE     1$              ;BRANCH BACK IF WE ARE STILL WAITING
2222 016654  013777  002534  163646             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2223 016662  013777  002536  163642             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2224 016670  004737  003546                     JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
2225 016674  104051                             ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
2226 016676  005037  002716                     CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
2227 016702  004737  003716                     JSR     PC,DATCHK       ;CHECK INBUF AFTER A MAINTENANCE MODE OPERATION
2228 016706  104201                             ERROR   +201            ;BUFFER DATA NOT CORRECT
2229 016710  004737  004014                     JSR     PC,DATCH2       ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATCHK
2230 016714  000415                             BR      3$              ;BRANCH TO CONTINUE
2231 016716  062706  000004             2$:     ADD     #4,SP           ;CLEAN UP THE STACK FROM THIS INTERRUPT
2232 016722  013777  002534  163600             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2233 016730  013777  002536  163574             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2234 016736  017737  163562  002562             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2235 016744  104036                             ERROR   +36             ;DR11 INTERRUPTED, BUT IT SHOULDN'T HAVE
2236 016746  000523                             BR      TST24           ;BRANCH TO NEXT TEST
2237 016750  012777  010000  163546  3$:     MOV     #MA,@CSR        ;MAINT MODE
2238 016756  012737  001000  002662             MOV     #1000,TIME      ;SET TIME LOOP COUNTER
2239 016764  013777  002632  163526             MOV     WCLEN,@WCR      ;MOVE WCLEN TO WCR
2240 016772  013777  002620  163522             MOV     INBUF,@BAR      ;MOVE INBUF TO BAR
2241 017000  017737  163524  002534             MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2242 017006  017737  163520  002536             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2243 017014  012777  017166  163506             MOV     #8$,@DRINV      ;SET UP INTERRUPT VECTOR
2244 017022  012777  010000  163474             MOV     #MA,@CSR        ;MAINT MODE
2245 017030  052777  000501  163466             BIS     #IE+CY+GO,@CSR  ;IE, CYCLE & GO
2246 017036  005337  002662             4$:     DEC     TIME            ;DECREMENT TO ZERO WHILE WAITING
2247 017042  001375                             BNE     4$              ;BRANCH BACK IF NOT ZERO
2248 017044  013777  002534  163456             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2249 017052  013777  002536  163452             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
```

```
2250 017060  017737  163440  002562          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2251 017066  022737  010700  002562          CMP     #10700,RCSR     ;SEE IF ONLY READY, MAINT, IE & CYCLE ARE SET
2252 017074  001404                          BEQ     5$              ;BRANCH IF THEY ARE
2253 017076  012737  010700  002574          MOV     #10700,ECSR     ;MOVE EXPECTED DATA TO ECSR
2254 017104  104040                          ERROR   +40             ;CSR IS WRONG
2255 017106  017737  163410  002570  5$:     MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
2256 017114  022777  177770  163376          CMP     #-10,@WCR       ;CHECK THAT NO TRANSFERS WERE MADE
2257 017122  001004                          BNE     6$              ;BRANCH TO ERROR IF THERE WERE
2258 017124  023737  002620  002570          CMP     INBUF,RBAR      ;CHECK THAT NO TRANSFERS WERE MADE
2259 017132  001412                          BEQ     7$              ;BRANCH AROUND ERROR IF NONE
2260 017134  017737  163360  002572  6$:     MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
2261 017142  012737  177770  002604          MOV     #-10,EWCR       ;MOVE EXPECTED DATA TO EWCR
2262 017150  013737  002620  002602          MOV     INBUF,EBAR      ;MOVE EXPECTED DATA TO EBAR
2263 017156  104041                          ERROR   +41             ;TRANSFERS SHOULD HAVE BEEN INHIBITED
2264 017160  005077  163340          7$:     CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
2265 017164  000414                          BR      TST24           ;BRANCH TO NEXT TEST
2266 017166  062706  000004          8$:     ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
2267 017172  013777  002534  163330          MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2268 017200  013777  002536  163324          MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2269 017206  017737  163312  002562          MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2270 017214  104042                          ERROR   +42             ;DR11 SHOULD NOT HAVE INTERRUPTED A 2ND TIME
```

```
                                .SBTTL  TEST #24 - TEST FOR 200 NPR TRANSFERS IN MAINT MODE
                                ;*********************************************************************
                                ;*TEST 24       TEST FOR 200 NPR TRANSFERS IN MAINT MODE
                                ;*
                                ;*      THIS TEST CHECKS FOR 200 NPR TRANSFERS IN MAINTENANCE MODE.
                                ;*
                                ;*********************************************************************
017216                          TST24:
017216  000004                          SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
017220  012737  017226  001310          MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2277 017226  004737  004036     999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2278 017232  005077  163266             CLR     @CSR            ;FORCE ACCESS TO CSR
2279 017236  012737  000200  002624     MOV     #200,BUFLEN     ;LENGTH OF BUFFER = 200
2280 017244  013737  002624  002632     MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
2281 017252  005437  002632             NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
2282 017256  004737  003472             JSR     PC,LODBUF       ;LOAD INBUF WITH INCREMENTING PATTERN
2283 017262  004737  003520             JSR     PC,CHKBFF       ;LOAD CHKBUF WITH MODIFIED INCREMENTED PATTERN
2284 017266  013777  002632  163224     MOV     WCLEN,@WCR      ;SET UP WCR
2285 017274  013777  002620  163220     MOV     INBUF,@BAR      ;SET UP BAR
2286 017302  012777  177777  163216     MOV     #-1,@BDR        ;MAINT AIDE
2287 017310  017737  163214  002534     MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2288 017316  017737  163210  002536     MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2289 017324  012777  017426  163176     MOV     #2$,@DRINV      ;INT VECTOR
2290 017332  013777  002542  163172     MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
2291 017340  005037  177776             CLR     PSW             ;LET THE DR11 INTERRUPT
2292 017344  012777  010000  163152     MOV     #MA,@CSR        ;MAINT MODE
2293 017352  012737  001000  002662     MOV     #1000,TIME      ;SET WAIT LOOP COUNTER
2294 017360  052777  000501  163136     BIS     #IE+CY+GO,@CSR  ;IE, CYCLE & GO
2295 017366  005337  002662     1$:     DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
2296 017372  001375                     BNE     1$              ;BRANCH BACK IF NOT ZERO
2297 017374  017737  163124  002562     MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2298 017402  013777  002534  163120     MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2299 017410  013777  002536  163114     MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2300 017416  104043                     ERROR   +43             ;EXPECTED INTERRUPT DID NOT OCCUR
2301 017420  005077  163100             CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
2302 017424  000402                     BR      3$              ;BRANCH AROUND THE STACK CLEANUP
2303 017426  062706  000004     2$:     ADD     #4,SP           ;CLEAN UP THE STACK AFTER INTERRUPT
2304 017432  013777  002534  163070     MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2305 017440  013777  002536  163064     MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2306 017446  004737  003546             JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
2307 017452  104051                     ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORRECT
2308 017454  005037  002716             CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
2309 017460  004737  003716             JSR     PC,DATCHK       ;CHECK INBUF AFTER A MAINTENANCE MODE OPERATION
2310 017464  104201                     ERROR   +201            ;BUFFER DATA NOT CORRECT
2311 017466  004737  004014             JSR     PC,DATCH2       ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATCHK
```

```
2318                                    .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.
                                        ;*
                                        ;**************************************************************
     017472                             TST25:
     017472  000004                     .       SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     017474  012737  017502  001310     .       MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2319 017502  004737  004036             999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2320 017506  005077  163012             .       CLR     @CSR            ;FORCE ACCESS TO CSR
2321 017512  012777  177776  163000     .       MOV     #-2,@WCR        ;SET UP WCR
2322 017520  013777  002616  162774     .       MOV     DIOMEM,@BAR     ;SET UP BAR
2323 017526  017737  162776  002534     .       MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2324 017534  017737  162772  002536     .       MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2325 017542  012777  017652  162760     .       MOV     #2$,@DRINV      ;INTERRUPT VECTOR TO 3$
2326 017550  013777  002542  162754     .       MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
2327 017556  005037  177776             .       CLR     PSW             ;LET THE DR11 INTERRUPT
2328 017562  012777  010000  162734     .       MOV     #MA,@CSR        ;MAINT MODE
2329 017570  052777  000062  162726     .       BIS     #F1+X6+X7,@CSR  ;SET FNCT1, XBA16, AND XBA17
2330 017576  052777  000501  162720     .       BIS     #IE+CY+GO,@CSR  ;SET IE, CYCLE, AND GO
2331 017604  012737  001000  002662     .       MOV     #1000,TIME      ;SET DELAY COUNTER
2332 017612  005337  002662             1$:     DEC     TIME            ;DECREMENT UNTIL ZERO
2333 017616  001375                     .       BNE     1$              ;BRANCH BACK IF NOT
2334 017620  013777  002534  162702     .       MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2335 017626  013777  002536  162676     .       MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2336 017634  017737  162664  002562     .       MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2337 017642  104035                     .       ERROR   +35             ;DR11 FAILED TO INTERRUPT
2338 017644  005077  162654             .       CLR     @CSR            ;CLEAR THE CSR TO DO AN .NIT
2339 017650  000431                     .       BR      TST26           ;BRANCH TO THE NEXT TEST
2340 017652  062706  000004             2$:     ADD     #4,SP           ;RESTORE THE STACK
2341 017656  013777  002534  162644     .       MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2342 017664  013777  002536  162640     .       MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2343 017672  017737  162626  002562     .       MOV     @CSR,RCSR       ;MOVE CSR DATA TO RCSR
2344 017700  013701  002562             .       MOV     RCSR,R1         ;MOVE DATA TO R1 FOR CHECKING
2345 017704  042701  037577             .       BIC     #37577,R1       ;CLEAR ALL BUT ERROR, NEX AND READY BITS
2346 017710  022701  140200             .       CMP     #ER+NX+RY,R1    ;SEE IF ALL THESE BITS ARE SET
2347 017714  001407                     .       BEQ     TST26           ;BRANCH TO THE NEXT TEST IF THEY ARE ALL SET
2348 017716  013737  002562  002574     .       MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
2349 017724  052737  140200  002574     .       BIS     #ER+NX+RY,ECSR  ;SET THE BIT THAT SHOULD HAVE BEEN SET
2350 017732  104040                     .       ERROR   +40             ;CSR IS WRONG
```

```
2357                                    .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.
                                        ;*
                                        ;*****************************************************************
     017734                             TST26:
     017734  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     017736  012737  017744  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2358 017744  004737  004036             999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2359 017750  012777  177760  162542             MOV     #-20,@WCR       ;SET UP WCR
2360 017756  012777  177776  162536             MOV     #-2,@BAR        ;SET UP BAR FOR PROC STATUS ADDRESS
2361 017764  017737  162540  002534             MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2362 017772  017737  162534  002536             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2363 020000  012777  020076  162522             MOV     #2$,@DRINV      ;INTERRUPT VECTOR TO 3$
2364 020006  013777  002542  162516             MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
2365 020014  012737  001000  002662             MOV     #1000,TIME      ;SET WAIT LOOP COUNTER
2366 020022  005037  177776                     CLR     PSW             ;LET THE DR11 INTERRUPT
2367 020026  012777  010000  162470             MOV     #MA,@CSR        ;MAINT MODE
2368 020034  052777  000563  162462             BIS     #563,@CSR       ;CYCLE, IE, FNCT1, XBA17, XBA16, AND GO TO CSR
2369 020042  005337  002662             1$:     DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
2370 020046  001375                             BNE     1$              ;BRANCH BACK IF NOT ZERO
2371 020050  013777  002534  162452             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2372 020056  013777  002536  162446             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2373 020064  017737  162434  002562             MOV     @CSR,RCSR       ;MOVE CSR CONTENTS TO RCSR
2374 020072  104035                             ERROR   +35             ;DR11 FAILED TO INTERRUPT
2375 020074  000433                             BR      TST27           ;BRANCH TO NEXT TEST
2376 020076  062706  000004             2$:     ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
2377 020102  013777  002534  162420             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2378 020110  013777  002536  162414             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2379 020116  017737  162402  002562             MOV     @CSR,RCSR       ;MOVE CSR CONTENTS TO RCSR
2380 020124  013701  002562                     MOV     RCSR,R1         ;MOVE CONTENTS TO R1 FOR TESTING
2381 020130  042701  037577                     BIC     #37577,R1       ;CLEAR ALL BUT THE ERROR AND READY BITS
2382 020134  022701  140200                     CMP     #ER+RY+NX,R1    ;SEE IF ERROR, READY AND NEX ARE SET
2383 020140  001411                             BEQ     TST27           ;BRANCH TO NEXT TEST IF THEY ARE
2384 020142  013737  002562  002574             MOV     RCSR,ECSR       ;MOVE EXPECTED DATA TO ECSR
2385 020150  052737  140200  002574             BIS     #ER+RY+NX,ECSR  ;SET THE BITS THAT SHOULD HAVE BEEN SET
2386 020156  104040                             ERROR   +40             ;CSR IS WRONG
2387 020160  005077  162340                     CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
```

```
                                        .SBTTL  TEST #27 - CHECK ACTUAL POSITION OF 2-N BURST SWITCH
                                        ;**********************************************************************
                                        ;*TEST 27       CHECK ACTUAL POSITION OF 2-N BURST SWITCH
                                        ;*
                                        ;*      THIS TEST INSURES THAT THE 2-N BURST SWITCH IS IN THE POSITION THAT
                                        ;*      THE DEVICE DESCRIPTOR WORD SAYS IT SHOULD BE.
                                        ;*
                                        ;**********************************************************************
     020164                             TST27:
     020164  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     020166  012737  020204  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2395 020174  032737  000001  002612             BIT     #BIT0,BORW      ;TESTING A 'B' OR A 'W'?
2396 020202  001456                             BEQ     INOOUT          ;BRANCH TO INOOUT IF DR11-B
2397 020204  004737  004036             999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2398 020210  012777  100000  162306             MOV     #EIR,@CSR       ;GO TO EIR MODE
2399 020216  017737  162302  002564             MOV     @CSR,REIR       ;MOVE EIR DATA TO REIR
2400 020224  005077  162274                     CLR     @CSR            ;GO BACK TO CSR
2401 020230  013701  002564                     MOV     REIR,R1         ;MOVE DATA TO R1 ALSO
2402 020234  000301                             SWAB    R1              ;GET BIT 8 INTO BIT 0 BY SWAPPING BYTES
2403 020236  006301                             ASL     R1              ;MOVE BIT 0 INTO BIT 1
2404 020240  042701  177775                     BIC     #CBIT1,R1       ;CLEAR ALL BUT BIT 1
2405 020244  013702  002720                     MOV     DDW,R2          ;PUT DEVICE DESCRIPTOR WORD IN R2
2406 020250  042702  177775                     BIC     #CBIT1,R2       ;CLEAR ALL BUT BIT 1
2407 020254  001402                             BEQ     1$              ;BRANCH IF IT IS CLEAR
2408 020256  005002                             CLR     R2              ;CLEAR THE BIT
2409 020260  000402                             BR      2$              ;GO TEST THE BIT
2410 020262  012702  000002             1$:     MOV     #BIT1,R2        ;SET THE BIT
2411 020266  020102                     2$:     CMP     R1,R2           ;SEE IF RECEIVED MATCHES EXPECTED
2412 020270  001017                             BNE     5$              ;BRANCH TO CHECK FOR LOOP ON TEST
2413 020272  013737  002564  002576             MOV     REIR,EEIR       ;MOVE EXPECTED DATA TO EEIR
2414 020300  032737  000400  002576             BIT     #BIT8,EEIR      ;TEST STATE OF BIT 8
2415 020306  001404                             BEQ     3$              ;BRANCH IF IT IS CLEAR
2416 020310  042737  000400  002576             BIC     #BIT8,EEIR      ;REVERSE STATE - EXPECTED CLEAR
2417 020316  000403                             BR      4$              ;GO CALL ERROR
2418 020320  052737  000400  002576     3$:     BIS     #BIT8,EEIR      ;REVERSE STATE - EXPECTED SET
2419 020326  104054                     4$:     ERROR   +54             ;2-N CYCLE BURST SWITCH IN WRONG POSITION
2420 020330  032777  040400  161002     5$:     BIT     #BIT14+BIT8,@SWR;SEE IF WE SHOULD LOOP ON THIS TEST
2421 020336  001322                             BNE     999$            ;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.

020340  032737  000004  002720  INOOUT: BIT     #BIT2,DDW       ;SEE IF CABLE IS IN
020346  001002                          BNE     TST30           ;BRANCH TO NEXT TEST IF CABLE IS IN
020350  000137  023074                  JMP     ENDEV           ;JUMP TO ENDEV - TESTS ARE NOT TO BE DONE
```

```
                                .SBTTL  TEST #30 - CHECK CSR BIT PATTERNS WITH MAINT BIT CLEAR
                                ;**************************************************************
                                ;*TEST 30       CHECK CSR BIT PATTERNS WITH MAINT BIT CLEAR
                                ;*
                                ;*      THIS TEST SETS ALL POSSIBLE COMBINATIONS OF SET BITS IN THE CSR WITH
                                ;*      THE MAINTENANCE BIT CLEAR, AND COMPARES THE RECEIVED CSR CONTENTS WITH
                                ;*      THAT OF THE EXPECTED PATTERNS IN 'MAICLR'.
                                ;*
                                ;**************************************************************
     020354                     TST30:
     020354  000004                     SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     020356  012737  020412  001310     MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2439 020364  004737  004036             JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2440 020370  005037  002716             CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +202
2441 020374  012700  000002             MOV     #2,R0           ;DO 2 SETS OF 200 PATTERNS
2442 020400  012701  006266             MOV     #MAICLR,R1      ;MOVE ADDRESS OF EXPECTED PATTERNS TO R1
2443 020404  005002                     CLR     R2              ;START WITH PATTERN ZERO
2444 020406  012703  000200     1$:     MOV     #200,R3         ;MOVE 200 TO THE LOOP COUNTER
2445 020412  052777  010000  162104 999$:  BIS     #MA,@CSR        ;SET MAINTENANCE AND
2446 020420  005077  162100             CLR     @CSR            ;CLEAR TO DO AN INIT
2447 020424  017737  162074  002562     MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2448 020432  022737  000200  002562     CMP     #RY,RCSR        ;MAKE SURE READY BIT IS THE ONLY BIT SET
2449 020440  001404                     BEQ     2$              ;BRANCH IF SO
2450 020442  012737  000200  002574     MOV     #RY,ECSR        ;MOVE EXPECTED DATA TO ECSR
2451 020450  104032                     ERROR   +32             ;READY IS NOT THE ONLY BIT SET
2452 020452  012777  177777  162040 2$:  MOV     #-1,@WCR        ;MOVE 1 WORD COUNT TO WCR IN CASE OF IE ENABLED
2453 020460  012777  052414  162034     MOV     #NOCARE,@BAR    ;MOVE A NOT-CARE ADDRESS TO BAR FOR SAME REASON
2454 020466  010277  162032             MOV     R2,@CSR         ;SET THE PARTICULAR FUNCTION BITS IN CSR
2455 020472  017737  162026  002562     MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2456 020500  011137  002574             MOV     (R1),ECSR       ;MOVE EXPECTED DATA TO ECSR
2457 020504  023737  002574  002562     CMP     ECSR,RCSR       ;COMPARE EXPECTED WITH RECEIVED
2458 020512  001427                     BEQ     6$              ;BRANCH IF OK
2459 020514  012737  000401  020524     MOV     #CY+GO,3$       ;REESTABLISH BIT PATTERN
2460 020522  040227                     BIC     R2,(PC)+        ;SEE IF BOTH CYCLE AND GO WERE SET
2461 020524  000401             3$:     .WORD   CY+GO           ;LOCATION TO HOLD BOTH CYCLE AND GO BITS
2462 020526  001016                     BNE     5$              ;BRANCH TO ERROR ONLY IF EITHER OR BOTH BITS WERE CLEAR
2463 020530  005737  002714             TST     MEMGMT          ;SEE IF MEMORY MANAGEMENT IS OUT THERE
2464 020534  001404                     BEQ     4$              ;BRANCH IF NOT
2465 020536  032737  000060  002574     BIT     #X6+X7,ECSR     ;SEE IF EITHER XBA16 OR XBA17 ARE SET
2466 020544  001407                     BEQ     5$              ;BRANCH TO ERROR IF BOTH ARE CLEAR
2467 020546  052737  140000  002574 4$:  BIS     #ER+NX,ECSR     ;SET THE ERROR AND NEX BITS - EXPECT THEM TO SET
2468 020554  023737  002574  002562     CMP     ECSR,RCSR       ;NOW SEE IF DATA MATCHES
2469 020562  001403                     BEQ     6$              ;BRANCH AROUND ERROR IF IT DOES
2470 020564  010237  002540     5$:     MOV     R2,BUT          ;MOVE THE BITS SET INTO CSR TO THE BUT LOCATION
2471 020570  104202                     ERROR   +202            ;CSR PATTERN NOT CORRECT
2472 020572  062701  000002     6$:     ADD     #2,R1           ;INCREMENT R1 TO NEXT EXPECTED PATTERN
2473 020576  005202                     INC     R2              ;INCREMENT THE PATTERN
2474 020600  005303                     DEC     R3              ;DECREMENT THE LOOP COUNTER
2475 020602  001303                     BNE     999$            ;BRANCH BACK IF NOT DONE
2476 020604  062702  000200             ADD     #200,R2         ;ADD 200 TO PATTERN LOCATION
2477 020610  005300                     DEC     R0              ;DECREMENT THE LOOP COUNTER AND
2478 020612  001275                     BNE     1$              ;BRANCH BACK IF 2ND OCTAL GROUP NOT DONE
```

```
                                .SBTTL  TEST #31 - CHECK BAR WITH CSR CLEAR
                                ;*************************************************************
                                ;*TEST 31       CHECK BAR WITH CSR CLEAR
                                ;*
                                ;*      THIS TEST CHECKS THAT BAR BIT 0 IS CLEAR WITH CSR CLEAR (CSR=0).
                                ;*
                                ;*************************************************************
020614                          TST31:
020614  000004                        SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
020616  012737  020624  001310       MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2485 020624  004737  004036   999$:  JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2486 020630  012777  001360  161664  MOV     #$TMP0,@BAR     ;PUT AN ADDRESS IN THE BAR
2487 020636  012777  000001  161660  MOV     #GO,@CSR        ;SET JUST THE GO BIT TO CLEAR THE READY BIT
2488 020644  017737  161654  002562  MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2489 020652  001403                   BEQ     1$              ;BRANCH AROUND ERROR IF EQUAL TO ZERO
2490 020654  005037  002574          CLR     ECSR            ;MOVE EXPECTED DATA TO ECSR
2491 020660  104040                   ERROR   +40             ;CSR IS WRONG
2492 020662  017737  161634  002570  1$:     MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
2493 020670  032737  000001  002570  BIT     #BIT0,RBAR      ;SEE IF THIS BIT IS CLEAR
2494 020676  001407                   BEQ     TST32           ;BRANCH TO NEXT TEST IF IT WAS
2495 020700  013737  002570  002602  MOV     RBAR,EBAR       ;MOVE EXPECTED DATA TO EBAR
2496 020706  042737  000001  002602  BIC     #BIT0,EBAR      ;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
2497 020714  104044                   ERROR   +44             ;BAR IS WRONG
```

```
                                .SBTTL  TEST #32 - TEST 7 SINGLE DATI NON BURST MODE TRANSFERS
                                ;*********************************************************************
                                ;*TEST 32       TEST 7 SINGLE DATI NON BURST MODE TRANSFERS
                                ;*
                                ;*      THIS TEST DOES 7 BIT PATTERNS OF SINGLE DATI NON BURST MODE TRANSFERS,
                                ;*      AND THAT THEY ARE DONE PROPERLY.
                                ;*
                                ;*********************************************************************
020716                          TST32:
020716  000004                        SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
020720  012737  020742  001310        MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2505 020726  012702  000007    1$:  MOV     #7,R2           ;SET UP LOOP COUNTER - DO 7 BIT PATTERNS
2506 020732  005037  002716        CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
2507 020736  012703  006250        MOV     #PATRNS,R3      ;MOVE ADDRESS OF PATTERNS TO R3
2508 020742  017737  161562  002534 999$:  MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2509 020750  017737  161556  002536       MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2510 020756  012777  021120  161544       MOV     #3$,@DRINV      ;INTERRUPT VECTOR TO 4$
2511 020764  013777  002542  161540       MOV     LEVEL,@DRVS     ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
2512 020772  005037  177776               CLR     PSW             ;LET THE DR11 INTERRUPT
2513 020776  012777  010000  161520       MOV     #MA,@CSR        ;DO AN INIT BY SETTING AND
2514 021004  005077  161514               CLR     @CSR            ;CLEARING THE CSR MAINTENANCE BIT
2515 021010  012777  177777  161502       MOV     #-1,@WCR        ;SET UP FOR 1 TRANSFER
2516 021016  012777  002614  161476       MOV     #NPR1,@BAR      ;TRANSFER FROM BUS ADDRESS IN NPR1
2517 021024  005077  161476               CLR     @BDR            ;GET READY TO RECEIVE DATA
2518 021030  011337  002614               MOV     (R3),NPR1       ;SET UP TRANSFER DATA
2519 021034  012777  000110  161462       MOV     #F3+IE,@CSR     ;SET THE NON BURST (FNCT3) AND IE
2520 021042  052777  000401  161454       BIS     #CY+GO,@CSR     ;SET THE CYCLE AND GO BITS
2521 021050  005037  002662               CLR     TIME            ;CLEAR THE TIME LOCATION FOR WAIT LOOP
2522 021054  005237  002662        2$:  INC     TIME            ;INCREMENT UNTIL WE GET TO ZERO AGAIN
2523 021060  001375               BNE     2$              ;BRANCH BACK IF WE AREN'T THERE YET
2524 021062  013777  002534  161440       MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2525 021070  013777  002536  161434       MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2526 021076  017737  161422  002562       MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2527 021104  011337  002606               MOV     (R3),ENPR1      ;MOVE PATTERN TO ENPR1
2528 021110  104035               ERROR   +35             ;DR11 FAILED TO INTERRUPT
2529 021112  005077  161406               CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
2530 021116  000450               BR      5$              ;BRANCH TO SEE IF THERE ARE ANY MORE PATTERNS TO CHECK
2531 021120  062706  000004    3$:  ADD     #4,SP           ;CLEAN STACK AFTER INTERRUPT
2532 021124  013777  002534  161376       MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2533 021132  013777  002536  161372       MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2534 021140  004737  004254               JSR     PC,ERRCHK       ;CLEAR IE, CHECK FOR ERROR
2535 021144  104021               ERROR   +21             ;ERROR BIT SHOULD HAVE BEEN CLEAR
2536 021146  017737  161350  002570       MOV     @BAR,RBAR       ;MOVE RECEIVED DATA TO RBAR
2537 021154  012737  002617  002602       MOV     #NPR1+3,EBAR    ;MOVE EXPECTED DATA TO EBAR
2538 021162  017737  161340  002566       MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
2539 021170  011337  002600               MOV     (R3),EBDR       ;MOVE EXPECTED DATA TO EBDR
2540 021174  017737  161320  002572       MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
2541 021202  005037  002604               CLR     EWCR            ;MOVE EXPECTED DATA TO EWCR
2542 021206  023737  002570  002602       CMP     RBAR,EBAR       ;COMPARE RECEIVED WITH EXPECTED
2543 021214  001010               BNE     4$              ;BRANCH IF WRONG
2544 021216  023737  002566  002600       CMP     RBDR,EBDR       ;COMPARE RECEIVED WITH EXPECTED
2545 021224  001004               BNE     4$              ;BRANCH IF WRONG
2546 021226  023737  002572  002604       CMP     RWCR,EWCR       ;COMPARE RECEIVED WITH EXPECTED
2547 021234  001401               BEQ     5$              ;BRANCH IF OK
2548 021236  104211              4$:  ERROR   +211            ;CSR AND-OR WCR AND-OR BAR ARE INCORRECT
2549 021240  062703  000002        5$:  ADD     #2,R3           ;INCREMENT TO NEXT PATTERN
2550 021244  005302               DEC     R2              ;DECREMENT THE LOOP COUNTER
```

```
2551 021246  001235                          BNE     999$            ;BRANCH BACK IF NOT ZERO YET
```

```
                        .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.
                        ;*
                        ;*****************************************************************
021250                  TST33:
021250  000004                  SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
021252  012737  021260  001310        MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2558 021260  004737  004036    999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2559 021264  012737  000200  002624        MOV     #200,BUFLEN     ;LENGTH OF BUFFER=200
2560 021272  004737  003472              JSR     PC,LODBUF       ;LOAD THE BUFFER WITH INCREMENTING PATTERN
2561 021276  012737  177600  002632        MOV     #-200,WCLEN     ;PREPARE NUMBER FOR WCR
2562 021304  013777  002632  161206        MOV     WCLEN,@WCR      ;SET UP WCR
2563 021312  013777  002620  161202        MOV     INBUF,@BAR      ;SET UP BAR
2564 021320  012777  177777  161200        MOV     #-1,@BDR        ;MAINT AIDE
2565 021326  017737  161176  002534        MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2566 021334  017737  161172  002536        MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2567 021342  012777  021452  161160        MOV     #2$,@DRINV      ;INT VECTOR
2568 021350  013777  002542  161154        MOV     LEVEL,@DRVS     ;INTERRUPT STATUS TO LEVEL OF DEVICE
2569 021356  005037  177776              CLR     PSW             ;LET THE DR11 INTERRUPT
2570 021362  012777  000100  161134        MOV     #IE,@CSR        ;SET INTERRUPT ENABLE
2571 021370  052777  000401  161126        BIS     #CY+GO,@CSR     ;CYCLE, GO
2572 021376  012737  001000  002662        MOV     #1000,TIME      ;WAIT FOR INTERRUPT
2573 021404  005337  002662    1$:     DEC     TIME            ;DECREMENT UNTIL WE REACH ZERO
2574 021410  001375              BNE     1$              ;BRANCH BACK IF NOT ZERO
2575 021412  013777  002534  161110        MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2576 021420  013777  002536  161104        MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2577 021426  017737  161072  002562        MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2578 021434  104035              ERROR   +35             ;DR11 FAILED TO INTERRUPT
2579 021436  012777  010000  161060        MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
2580 021444  005077  161054              CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
2581 021450  000426              BR      TST34           ;BRANCH TO NEXT TEST
2582 021452  062706  000004    2$:     ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
2583 021456  013777  002534  161044        MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2584 021464  013777  002536  161040        MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2585 021472  004737  003546              JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
2586 021476  104051              ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
2587 021500  017737  161022  002566        MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
25 3 021506  022737  000177  002566        CMP     #177,RBDR       ;CHECK THAT WORD #200 OF INBUF IS IN BDR
2589 021514  001404              BEQ     TST34           ;BRANCH TO NEXT TEST IF IT IS
2590 021516  012737  000177  002600        MOV     #177,EBDR       ;MOVE EXPECTED DATA TO EBDR
2591 021524  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.
                                ;*
                                ;*********************************************************************
     021526                     TST34:
     021526  000004                    SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     021530  012737  021552  001310    MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2599 021536  012702  000007    1$:     MOV     #7,R2           ;DO 7 BIT PATTERNS
2600 021542  005037  002716            CLR     ERRCNT          ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
2601 021546  012703  006250            MOV     #PATRNS,R3      ;MOVE ADDRESS OF PATTERNS TO R3
2602 021552  017737  160752  002534  999$:    MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2603 021560  017737  160746  002536            MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2604 021566  012777  021740  160734            MOV     #3$,@DRINV      ;INTERRUPT VECTOR TO 3$
2605 021574  013777  002542  160730            MOV     LEVEL,@DRVS     ;INTERRUPT STATUS TO LEVEL OF DEVICE
2606 021602  005037  177776            CLR     PSW             ;LET DR11 INTERRUPT
2607 021606  012777  010000  160710            MOV     #MA,@CSR        ;DO AN INIT
2608 021614  005077  160704            CLR     @CSR            ;FORCE ACCESS TO CSR
2609 021620  012777  177777  160672            MOV     #-1,@WCR        ;SET UP FOR 1 TRANSFER
2610 021626  012777  002614  160666            MOV     #NPR1,@BAR      ;TRANSFER TO BUS ADDRESS IN NPR1
2611 021634  005037  002614            CLR     NPR1            ;GET READY TO RECEIVE DATA
2612 021640  011377  160662            MOV     (R3),@BDR       ;SET UP TO TRANSFER DATA
2613 021644  012777  000112  160652            MOV     #F1+F3+IE,@CSR  ;DATO (FNCT1), FNCT3, IE
2614 021652  052777  000401  160644            BIS     #CY+GO,@CSR     ;CYCLE, GO
2615 021660  012737  001000  002662            MOV     #1000,TIME      ;CLEAR THE TIME LOCATION FOR WAIT LOOP
2616 021666  005337  002662    2$:     DEC     TIME            ;DECREMENT UNTIL WE GET BACK TO ZERO
2617 021672  001375            BNE     2$              ;BRANCH BACK IF NOT ZERO
2618 021674  013777  002534  160626            MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2619 021702  013777  002536  160622            MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2620 021710  017737  160610  002562            MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2621 021716  011337  001360            MOV     (R3),$TMP0      ;MOVE PATTERN TO $TMP0
2622 021722  104035            ERROR   +35             ;DR11 FAILED TO INTERRUPT
2623 021724  012777  010000  160572            MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
2624 021732  005077  160566            CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
2625 021736  000445            BR      6$              ;BRANCH TO DO NEXT PATTERN
2626 021740  062706  000004    3$:     ADD     #4,SP           ;CLEAN UP STACK FROM INTERRUPT
2627 021744  013777  002534  160556            MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2628 021752  013777  002536  160552            MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2629 021760  004737  004254            JSR     PC,ERRCHK       ;CLEAR IE, CHECK FOR ERROR
2630 021764  104021            ERROR   +21             ;ERROR BIT SHOULD HAVE BEEN CLEAR
2631 021766  017737  160526  002572            MOV     @WCR,RWCR       ;MOVE RECEIVED DATA TO RWCR
2632 021774  001403            BEQ     4$              ;BRANCH IF IT IS EQUAL TO ZERO
2633 021776  011337  001362            MOV     (R3),$TMP1      ;MOVE PATTERN TO $TMP0
2634 022002  104206            ERROR   +206            ;WCR NOT EQUAL TO ZERO
2635 022004  017737  160512  002570  4$:     MOV     @BAR,RBAR       ;MOVE RECEIVED ADDRESS TO RBAR
2636 022012  022737  002617  002570            CMP     #NPR1+3,RBAR    ;COMPARE CORRECT BAR WITH BAR CABLE MODE TESTING LEAVES
2637                                                            ;BIT 0 OF BAR SET. THEREFORE MUST CHECK FOR ODD ADDRESS
2638 022020  001406            BEQ     5$              ;BRANCH IF IT IS OK
2639 022022  011737  002617  002602            MOV     #NPR1+3,EBAR    ;MOVE EXPECTED DATA TO EBAR
2640 022030  011337  001362            MOV     (R3),$TMP1      ;MOVE PATTERN TO $TMP0
2641 022034  104207            ERROR   +207            ;BAR IS WRONG
2642 022036  021337  002614    5$:     CMP     (R3),NPR1       ;CHECK FOR CORRECT DATA
2643 022042  001403            BEQ     6$              ;BRANCH IF CORRECT DATA WAS TRANSFERRED
2644 022044  011337  002606            MOV     (R3),ENPR1      ;MOVE EXPECTED DATA TO ENPR1
```

```
2645 022050  104210                    ERROR   +210          ;DATA NOT TRANSFERED CORRECTLY
2646 022052  062703  000002      6$:    ADD     #2,R3         ;POINT TO NEXT BIT PATTERN
2647 022056  005302                    DEC     R2            ;COUNT 1 PATTERN DONE
2648 022060  001234                    BNE     999$          ;BRANCH BACK IF NOT DONE
```

```
                                .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.
                                ;*
                                ;*****************************************************************
     022062                     TST35:
     022062  000004                     SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     022064  012737  022072  001310     MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2655 022072  005077  160426     999$:  CLR     @CSR            ;FORCE ACCESS TO CSR
2656 022076  012737  000200  002624     MOV     #200,BUFLEN     ;LENGTH OF BUFFER=200
2657 022104  004737  003472             JSR     PC,LODBUF       ;LOAD THE BUFFER WITH INCREMENTING PATTERN
2658 022110  013737  002624  002632     MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
2659 022116  005437  002632             NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
2660 022122  013777  002632  160370     MOV     WCLEN,@WCR      ;SET UP WCR
2661 022130  013777  002620  160364     MOV     INBUF,@BAR      ;SET UP BAR
2662 022136  012777  052525  160362     MOV     #52525,@BDR     ;SET UP BDR
2663 022144  017737  160360  002534     MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2664 022152  017737  160354  002536     MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2665 022160  012777  022270  160342     MOV     #2$,@DRINV      ;INTERRUPT VECTOR
2666 022166  013777  002542  160336     MOV     LEVEL,@DRVS     ;INTERRUPT STATUS TO LEVEL OF DEVICE
2667 022174  005037  177776             CLR     PSW             ;LET THE DR11 INTERRUPT
2668 022200  012777  000102  160316     MOV     #IE+F1,@CSR     ;IE, FNCT1
2669 022206  052777  000401  160310     BIS     #CY+GO,@CSR     ;CYCLE, GO
2670 022214  012737  001000  002662     MOV     #1000,TIME      ;MOVE WAIT LOOP VALUE TO TIME LOCATION
2671 022222  005337  002662     1$:    DEC     TIME            ;DECREMENT UNTIL WE REACH ZERO
2672 022226  001375                     BNE     1$              ;BRANCH BACK IF NOT ZERO
2673 022230  013777  002534  160272     MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2674 022236  013777  002536  160266     MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2675 022244  017737  160254  002562     MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2676 022252  104035                     ERROR   +35             ;DR11 FAILED TO INTERRUPT
2677 022254  012777  010000  160242     MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
2678 022262  005077  160236             CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
2679 022266  000421                     BR      TST36           ;BRANCH TO NEXT TEST
2680 022270  062706  000004     2$:    ADD     #4,SP           ;CLEAN STACK AFTER INTERRUPT
2681 022274  013777  002534  160226     MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2682 022302  013777  002536  160222     MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2683 022310  004737  003546             JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
2684 022314  104051                     ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
2685 022316  004737  004106             JSR     PC,DATOCK       ;CHECK INBUF
2686 022322  104046                     ERROR   +46             ;BUFFER DATA NOT CORRECT
2687 022324  004737  004170             JSR     PC,DATOC2       ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATOCK
2688 022330  104047                     ERROR   +47             ;TOO MANY WORDS WERE TRANSFERED
```

```
                                        .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.
                                        ;*
                                        ;*********************************************************************
     022332                             TST36:
     022332  000004                             SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     022334  012737  022342  001310             MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2695 022342  004737  004036             999$:   JSR     PC,CLRNUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2696 022346  005077  160152                     CLR     @CSR            ;FORCE ACCESS TO CSR
2697 022352  012737  000200  002624             MOV     #200,BUFLEN     ;LENGTH OF BUFFER=200
2698 022360  004737  003472                     JSR     PC,LODBUF       ;LOAD THE BUFFER WITH INCREMENTING PATTERN
2699 022364  013737  002624  002632             MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
2700 022372  005437  002632                     NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
2701 022376  013777  002632  160114             MOV     WCLEN,@WCR      ;SET-UP WCR
2702 022404  013777  002620  160110             MOV     INBUF,@BAR      ;SET-UP BAR
2703 022412  012777  177777  160106             MOV     #-1,@BDR        ;MAINT AIDE
2704 022420  017737  160104  002534             MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2705 022426  017737  160100  002536             MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2706 022434  012777  022544  160066             MOV     #2$,@DRINV      ;INT VECTOR
2707 022442  013777  002542  160062             MOV     LEVEL,@DRVS     ;INTERRUPT STATUS TO LEVEL OF DEVICE
2708 022450  005037  177776                     CLR     PSW             ;LET THE DR11 INTERRUPT
2709 022454  012777  000110  160042             MOV     #F3+IE,@CSR     ;FNCT3, IE
2710 022462  052777  000401  160034             BIS     #CY+GO,@CSR     ;CYCLE, GO
2711 022470  012737  001000  002662             MOV     #1000,TIME      ;SET WAIT LOOP COUNTER
2712 022476  005337  002662             1$:     DEC     TIME            ;DECREMENT UNTIL WE REACH ZERO
2713 022502  001375                             BNE     1$              ;BRANCH BACK IF NOT ZERO
2714 022504  013777  002534  160016             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2715 022512  013777  002536  160012             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2716 022520  017737  160000  002562             MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2717 022526  104035                             ERROR   +35             ;DR11 FAILED TO INTERRUPT
2718 022530  012777  010000  157766             MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
2719 022536  005077  157762                     CLR     @CSR            ;CLEAR THE CSR TO DO AN  INIT
2720 022542  000426                             BR      TST37           ;BRANCH TO NEXT TEST
2721 022544  062706  000004             2$:     ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
2722 022550  013777  002534  157752             MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2723 022556  013777  002536  157746             MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2724 022564  004737  003546                     JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
2725 022570  104051                             ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
2726 022572  017737  157730  002566             MOV     @BDR,RBDR       ;MOVE RECEIVED DATA TO RBDR
2727 022600  022737  000177  002566             CMP     #177,RBDR       ;CHECK THAT WORD #200 OF INBUF IS IN BDR
2728 022606  001404                             BEQ     TST37           ;BRANCH TO NEXT TEST IF OK
2729 022610  012737  000177  002600             MOV     #177,EBDR       ;MOVE EXPECTED DATA TO EBDR
2730 022616  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.
                                ;*
                                ;*****************************************************************
     022620                     TST37:
     022620  000004                    SCOPE                   ;PROCESS LOOPING AND TEST NUMBER INCREMENT
     022622  012737  022630  001310    MOV     #999$,$LPERR    ;SET LOOP ON ERROR TO 999$
2737 022630  004737  004036    999$:   JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2738 022634  005077  157664            CLR     @CSR            ;FORCE ACCESS TO CSR
2739 022640  012737  000200  002624    MOV     #200,BUFLEN     ;LENGTH OF BUFFER=200
2740 022646  004737  003472            JSR     PC,LODBUF       ;LOAD THE BUFFER WITH INCREMENTING PATTERN
2741 022652  013737  002624  002632    MOV     BUFLEN,WCLEN    ;PREPARE NUMBER FOR WCR
2742 022660  005437  002632            NEG     WCLEN           ;2'S COMPLEMENT OF BUFLEN
2743 022664  013777  002632  157626    MOV     WCLEN,@WCR      ;SET UP WCR
2744 022672  013777  002620  157622    MOV     INBUF,@BAR      ;SET UP BAR
2745 022700  012777  052525  157620    MOV     #52525,@BDR     ;SET UP BDR
2746 022706  017737  157616  002534    MOV     @DRINV,SDRINV   ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2747 022714  017737  157612  002536    MOV     @DRVS,SDRVS     ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2748 022722  012777  023032  157600    MOV     #2$,@DRINV      ;INTERRUPT VECTOR
2749 022730  013777  002542  157574    MOV     LEVEL,@DRVS     ;INTERRUPT STATUS TO LEVEL OF DEVICE
2750 022736  005037  177776            CLR     PSW             ;LET THE DR11 INTERRUPT
2751 022742  012777  000112  157554    MOV     #F1+F3+IE,@CSR  ;FNCT1, FNCT3, IE
2752 022750  052777  000401  157546    BIS     #CY+GO,@CSR     ;CYCLE, GO
2753 022756  012737  001000  002662    MOV     #1000,TIME      ;SET WAIT LOOP COUNTER
2754 022764  005337  002662    1$:     DEC     TIME            ;DECREMENT UNTIL WE GET TO ZERO
2755 022770  001375                    BNE     1$              ;BRANCH BACK IF NOT ZERO
2756 022772  013777  002534  157530    MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2757 023000  013777  002536  157524    MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2758 023006  017737  157512  002562    MOV     @CSR,RCSR       ;MOVE RECEIVED DATA TO RCSR
2759 023014  104035                    ERROR   +35             ;DR11 FAILED TO INTERRUPT
2760 023016  012777  010000  157500    MOV     #MA,@CSR        ;SET THE MAINTENANCE BIT AND
2761 023024  005077  157474            CLR     @CSR            ;CLEAR THE CSR TO DO AN INIT
2762 023030  000421                    BR      TST40           ;BRANCH TO NEXT TEST
2763 023032  062706  000004    2$:     ADD     #4,SP           ;CLEAN UP STACK AFTER INTERRUPT
2764 023036  013777  002534  157464    MOV     SDRINV,@DRINV   ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2765 023044  013777  002536  157460    MOV     SDRVS,@DRVS     ;RESTORE LOCATION USED AS THE INTERRUPT PS
2766 023052  004737  003546            JSR     PC,INTA         ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
2767 023056  104051                    ERROR   +51             ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
2768 023060  004737  004106            JSR     PC,DATOCK       ;CHECK INBUF
2769 023064  104046                    ERROR   +46             ;BUFFER DATA NOT CORRECT
2770 023066  004737  004170            JSR     PC,DATOC2       ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATOCK
2771 023072  104047                    ERROR   +47             ;TOO MANY WORDS WERE TRANSFERED
```

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

```
 2804                                   .SBTTL  END OF PASS ROUTINE
                                        ;*********************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO GOAGIN
023246                                  $EOP:
023246  000004                                SCOPE
023250  005037  001302                       CLR     $TSTNM          ;;ZERO THE TEST NUMBER
023254  005237  001416                       INC     $PASS           ;;INCREMENT THE PASS NUMBER
023260  100004                             BPL     10$             ;&& BRANCH IF STILL POSITIVE
023262  005037  001416                       CLR     $PASS           ;&& CLEAR THE PASS LOCATION AND
023266  005237  001420                       INC     $PASS+2         ;&& INCREMENT $PASS+2 TO SHOW 1 OVERFLOW
023272  005327                          10$:    DEC     (PC)+           ;;LOOP?
023274  000001                          $EOPCT: .WORD   1
023276  003076                             BGT     $DOAGN          ;;YES
023300  012737                             MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
023302  000001                          $ENDCT: .WORD   1
023304  023274                                  $EOPCT
023306  105737  002710                       TSTB    EOPLOC          ;&& SEE IF EOP MESSAGES ARE TO PRINT
023312  001420                             BEQ     1$              ;&& BRANCH IF THEY ARE
023314  005737  001312                       TST     $ERTTL          ;&& SEE IF ANY ERRORS THIS PASS
023320  001020                             BNE     2$              ;&& BRANCH IF THERE ARE TO PRINT EOP
023322  105737  002711                       TSTB    EOPLOC+1        ;&& SEE IF ERROR OCCURED IN ANOTHER DEVICE
023326  001012                             BNE     1$              ;&& BRANCH IF ANY TO PRINT EOP
023330  105737  027423                       TSTB    CHARCT          ;&& SEE IF ANY CHARACTER WAS INPUTED
023334  001447                             BEQ     $GET42          ;&& BRANCH IF NOT
023336  122737  000007  027423               CMPB    #7,CHARCT       ;&& SEE IF OTHER THAT (^G) TYPED, REQUESTING EOP
023344  001443                             BEQ     $GET42          ;&& BRANCH IF A (^G) - THIS NOT MEANT FOR EOP ROUTINE
023346  105037  027423                       CLRB    CHARCT          ;&& CLEAR THE LOCATION
023352  000405                             BR      3$              ;&& GET OVER $ERTTL TEST AND <CRLF> PRINT
023354  005737  001312                  1$:  TST     $ERTTL          ;&& SEE IF AN EXTRA <CRLF> NEEDS PRINTING
023360  001402                             BEQ     3$              ;&& BRANCH IF NOT
023362  104401  001405                  2$:  TYPE    ,$CRLF          ;&& TYPE A <CRLF> TO SPACE EOP FROM ERROR
023366  105037  002711                  3$:  CLRB    EOPLOC+1        ;&& CLEAR THE UPPER BYTE OF EOPLOC
023372  104401  023503                       TYPE    ,$ENDMG         ;;TYPE "END PASS #"
023376  013746  001420                       MOV     $PASS+2,-(SP)   ;&& MOVE OVERFLOW TO STACK FOR PRINTING
023402  013746  001416                       MOV     $PASS,-(SP)     ;;MOVE $PASS TO THE STACK FOR PRINTING
023406  104406                             TYPDE                   ;&& TYPE THE NUMBER IN EXTENDED DECIMAL
023410  104401  023500                       TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
023414  005737  001312                       TST     $ERTTL          ;&& SEE IF ANY ERRORS THIS PASS
023420  001413                             BEQ     4$              ;&& BRANCH AROUND NUMBER OF ERRORS MESSAGE IF NONE
023422  022737  000001  002416               CMP     #1,QTYBRD       ;&& SEE IF ONLY 1 BOARD IS BEING TESTED
023430  001007                             BNE     4$              ;&& BRANCH IF NOT - 'TOTAL ERRORS' IS MEANINGLESS
023432  104401  027424                       TYPE    ,TESLR          ;&& TYPE:  '  TOTAL ERRORS SINCE LAST REPORT '
023436  013746  001312                       MOV     $ERTTL,-(SP)    ;&& PUT $ERTTL ON STACK FOR PRINTING
023442  104405                             TYPDS                   ;&& TYPE THE NUMBER OF ERRORS IN DECIMAL
023444  004737  003126                       JSR     PC,ERCAPT       ;&& GO CAPTURE THE DEVICE, PASS & # OF ERRORS
023450  104401  001405                  4$:  TYPE    ,$CRLF          ;&& TYPE A <CRLF>
023454  013700  000042                  $GET42: MOV  @#42,R0         ;;GET MONITOR ADDRESS
023460  001405                             BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
023462  000005                             RESET                   ;;CLEAR THE WORLD
023464  004710                          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
023466  000240                             NOP                     ;;SAVE ROOM
023470  000240                             NOP                     ;;FOR
023472  000240                             NOP                     ;;ACT11
023474                                  $DOAGN:
```

```
                023474  000137                          JMP     @(PC)+          ;;RETURN
                023476  023516                  $RTNAD: .WORD   GOAGIN
                023500     377     377     000  $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
                023503     105     116     104  $ENDMG: .ASCIZ  /END PASS #/
     2805       023516  005037  001422          GOAGIN: CLR     $DEVCT          ;CLEAR DEVICE COUNT
     2806       023522  022737  000001  002416          CMP     #1,QTYBRD       ;IS THERE ONLY ONE DEVICE UNDER TEST?
     2807       023530  001002                          BNE     RSTRT           ;BR, IF NOT
     2808       023532  000137  011206                  JMP     REINIT          ;GO DO ANOTHER PASS
     2809       023536  005037  001424          RSTRT:  CLR     $UNIT           ;CLEAR UNIT NUMBER
     2810       023542  000137  011006                  JMP     BEGIN1          ;GO BEGIN TEST OF NEXT DEVICE
```

```
2811                                    .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
                                        ;*
023546  105737  001357          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
023552  100002                          BPL     1$              ;;BR IF YES
023554  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
023556  000430                          BR      3$              ;;LEAVE
023560  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
023562  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
023566  122737  000001  001432          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
023574  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
023576  132737  000100  001433          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
023604  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
023606  010037  023616                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
023612  004737  030250                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
023616  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
023620  132737  000040  001433  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
023626  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
023630  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
023632  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
023634  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
023636  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
023640  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
023644  000002                          RTI                     ;;RETURN
023646  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
023652  001430                          BEQ     8$
023654  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
023660  001006                          BNE     5$
023662  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
023664  104401                          TYPE                    ;;TYPE A CR AND LF
023666  001405                          $CRLF
023670  105037  024110                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
023674  000755                          BR      2$              ;;GET NEXT CHARACTER
023676  004737  023760          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
023702  123726  001356          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
023706  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
023710  013746  001354                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
023714  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
023720  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
023722  004737  023760                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
023726  105337  024110                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
023732  000770                          BR      7$              ;;LOOP
                                        ;HORIZONTAL TAB PROCESSOR
023734  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
```

```
023740  004737  023760          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
023744  132737  000007  024110          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
023752  001372                          BNE     9$              ;;TAB STOP
023754  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
023756  000724                          BR      2$              ;;GET NEXT CHARACTER
023760  105777  155364      $TYPEC:     TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
023764  100375                          BPL     $TYPEC
023766  116677  000002  155356          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
023774  105777  155344                  TSTB    @$TKS           ;;SEE IF KEYBOARD IS TALKING.
024000  100027                          BPL     2$              ;;BRANCH IF IT ISN'T.
024002  117737  155340  027423          MOVB    @$TKB,CHARCT    ;&& PUT CHARACTER IN CHARCT
024010  142737  000200  027423          BICB    #200,CHARCT     ;&& BIT CLEAR PARITY BIT.
024016  122737  000023  027423          CMPB    #23,CHARCT      ;&& SEE IF THIS IS A ^S.
024024  001015                          BNE     2$              ;;BRANCH TO CONTINUE IF IT ISN'T.
024026  105777  155312      3$:         TSTB    @$TKS           ;;WAIT FOR ANOTHER INPUT.
024032  100375                          BPL     3$              ;;BRANCH BACK IF NOT READY.
024034  117737  155306  027423          MOVB    @$TKB,CHARCT    ;&& PUT CHARACTER IN CHARCT
024042  142737  000200  027423          BICB    #200,CHARCT     ;&& BIT CLEAR PARITY BIT.
024050  122737  000021  027423          CMPB    #21,CHARCT      ;&& SEE IF THIS IS A ^Q.
024056  001363                          BNE     3$              ;;BRANCH BACK FOR MORE WAIT IF NOT.
024060  122766  000015  000002  2$:     CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
024066  001003                          BNE     1$              ;;BRANCH IF NO
024070  105037  024110                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
024074  000406                          BR      $TYPEX          ;;EXIT
024076  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
024104  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
024106  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
024110  000000      $CHARCNT:.WORD      0                       ;;CHARACTER COUNT STORAGE
024112  000207      $TYPEX: RTS         PC
```

```
2812                                    .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
                                        ;**********************************************************************
                                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                        ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                        ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                        ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                        ;*      .BYTE   M               ;;M=1 OR 0
                                        ;*                                      ;;1=TYPE LEADING ZEROS
                                        ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                        ;*
                                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                        ;*$TYPOS OR $TYPOC
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT
024114  017646  000000        $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
024120  116637  000001  024337        MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
024126  112637  024341                MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
024132  062716  000002                ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
024136  000406                        BR      $TYPON
024140  112737  000001  024337  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
024146  112737  000006  024341        MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
024154  112737  000005  024336  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
024162  010346                        MOV     R3,-(SP)        ;;SAVE R3
024164  010446                        MOV     R4,-(SP)        ;;SAVE R4
024166  010546                        MOV     R5,-(SP)        ;;SAVE R5
024170  113704  024341                MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
024174  005404                        NEG     R4
024176  062704  000006                ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
024202  110437  024340                MOVB    R4,$OMODE       ;;SAVE IT FOR USE
024206  113704  024337                MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
024212  016605  000012                MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
024216  005003                        CLR     R3              ;;CLEAR THE OUTPUT WORD
024220  006105                1$:     ROL     R5              ;;ROTATE MSB INTO "C"
024222  000404                        BR      3$              ;;GO DO MSB
024224  006105                2$:     ROL     R5              ;;FORM THIS DIGIT
024226  006105                        ROL     R5
024230  006105                        ROL     R5
024232  010503                        MOV     R5,R3
024234  006103                3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
024236  105337  024340                DECB    $OMODE          ;;TYPE THIS DIGIT?
024242  100016                        BPL     7$              ;;BR IF NO
024244  042703  177770                BIC     #177770,R3      ;;GET RID OF JUNK
024250  001002                        BNE     4$              ;;TEST FOR 0
024252  005704                        TST     R4              ;;SUPPRESS THIS 0?
024254  001403                        BEQ     5$              ;;BR IF YES
024256  005204                4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
024260  052703  000060                BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
024264  052703  000040                BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
```

```
024270  110337  024334                  MOVB    R3,8$           ;;SAVE FOR TYPING
024274  104401  024334                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
024300  105337  024336          7$:     DECB    $OCNT           ;;COUNT BY 1
024304  003347                          BGT     2$              ;;BR IF MORE TO DO
024306  002402                          BLT     6$              ;;BR IF DONE
024310  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
024312  000744                          BR      2$              ;;GO DO THE LAST DIGIT
024314  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
024316  012604                          MOV     (SP)+,R4        ;;RESTORE R4
024320  012603                          MOV     (SP)+,R3        ;;RESTORE R3
024322  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
024330  012616                          MOV     (SP)+,(SP)
024332  000002                          RTI                     ;;RETURN
024334     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
024335     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
024336     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
024337     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
024340  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
                                .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
024342  012737  000001  001364  $TYPDE: MOV     #1,$TMP2        ;&& SHOW ENTRY AT $TYPDE
024350  000402                        BR      $TYPD           ;&& GO TO ROUTINE
024352  005037  001364          $TYPDS: CLR     $TMP2           ;&& SHOW ENTRY AT $TYPDS
024356                          $TYPD:
024356  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
024360  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
024362  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
024364  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
024366  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
024370  012746  020200                MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
024374  016605  000020                MOV     20(SP),R5       ;;GET THE INPUT NUMBER
024400  100004                        BPL     1$              ;;BR IF INPUT IS POS.
024402  005405                        NEG     R5              ;;MAKE THE BINARY NUMBER POS.
024404  112766  000055  000001        MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
024412  005000                1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
024414  012703  024656                MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
024420  112723  000040                MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
024424  005002                2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
024426  016001  024646                MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
024432  160105                3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
024434  002402                        BLT     4$              ;;BR IF DONE
024436  005202                        INC     R2              ;;INCREASE THE BCD DIGIT BY 1
024440  000774                        BR      3$
024442  060105                4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
024444  005702                        TST     R2              ;;CHECK IF BCD DIGIT=0
024446  001002                        BNE     5$              ;;FALL THROUGH IF 0
024450  105716                        TSTB    (SP)            ;;STILL DOING LEADING 0'S?
024452  100407                        BMI     7$              ;;BR IF YES
024454  106316                5$:     ASLB    (SP)            ;;MSD?
024456  103003                        BCC     6$              ;;BR IF NO
024460  116663  000001  177777        MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
024466  052702  000060          6$:   BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
024472  052702  000040          7$:   BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
024476  110223                        MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
024500  005720                        TST     (R0)+           ;;JUST INCREMENTING
024502  020027  000010                CMP     R0,#10          ;;CHECK THE TABLE INDEX
024506  002746                        BLT     2$              ;;GO DO THE NEXT DIGIT
```

```
024510  003002                          BGT     8$              ;;GO TO EXIT
024512  010502                          MOV     R5,R2           ;;GET THE LSD
024514  000764                          BR      6$              ;;GO CHANGE TO ASCII
024516  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
024520  100003                          BPL     9$              ;;BR IF NO
024522  116663  177777  177776        MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
024530  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
024532  005737  001364                  TST     $TMP2           ;&& WAS ENTRY AT $TYPDS?
024536  001405                          BEQ     10$             ;&& BRANCH IF SO
024540  005766  000020                  TST     20(SP)          ;&& TEST THE OVERFLOW LOCATION
024544  001402                          BEQ     10$             ;&& BRANCH IF NON-ZERO
024546  004737  024670                  JSR     PC,EXPAND       ;&& GO EXPAND THE DECIMAL NUMBER
024552                          10$:
024552  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
024554  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
024556  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
024560  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
024562  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
024564  104401  024656                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
024570  005737  001364                  TST     $TMP2           ;&& SEE IF ENTRY WAS AT $TYPDS
024574  001417                          BEQ     15$             ;&& BRANCH IF SO
024576  016666  000002  000006          MOV     2(SP),6(SP)     ;&& ADJUST THE STACK
024604  012666  000002                  MOV     (SP)+,2(SP)
024610  005726                          TST     (SP)+
024612  105037  024665                  CLRB    $DBLK+7         ;&& REPLACE ORIGINAL TERMINATOR
024616  112737  000040  024664          MOVB    #' ,$DBLK+6     ;&& REPLACE ORIGINAL SPACE CHARACTER
024624  112737  000040  024656          MOVB    #' ,$DBLK       ;&& REPLACE ORIGINAL SPACE CHARACTER
024632  000002                          RTI                     ;&& RETURN TO USER
024634  016666  000002  000004  15$:     MOV     2(SP).4(SP)     ;;ADJUST THE STACK
024642  012616                          MOV     (SP)+,(SP)
024644  000002                          RTI                     ;;RETURN TO USER
024646  023420  001750  000144  $DTBL:   .WORD   10000.,1000.,100.,10.
024656                          $DBLK:   .BLKW   5
```

```
                                        .SBTTL  SUBROUTINE TO EXPAND DECIMAL TO LARGER THAN 32767
                                    ;*****************************************************************
024670  012702  024664          EXPAND: MOV     #$DBLK+6,R2     ;&& MOVE LOCATION OF LCD+1 TO R2
024674  012703  024670                  MOV     #$DBLK+12,R3    ;&& MOVE NEW LOCATION OF LCD+2 TO R3
024700  105043                          CLRB    -(R3)           ;&& MAKE SURE TERMINATOR IS THERE
024702  114243                          MOVB    -(R2),-(R3)     ;&& MOVE THE 5 ASCII'S TO THEIR NEW LOCATIONS
024704  114243                          MOVB    -(R2),-(R3)     ;&&
024706  114243                          MOVB    -(R2),-(R3)     ;&&
024710  114243                          MOVB    -(R2),-(R3)     ;&&
024712  114243                          MOVB    -(R2),-(R3)     ;&&
024714  112743  000040                  MOVB    #' ,-(R3)       ;&& MOVE 4 SPACE CHARACTERS TO THE 4 NEW LOCATIONS
024720  112743  000040                  MOVB    #' ,-(R3)       ;&&
024724  112743  000040                  MOVB    #' ,-(R3)       ;&&
024730  112743  000040                  MOVB    #' ,-(R3)       ;&&
024734  013746  001360                  MOV     $TMP0,-(SP)     ;&& SAVE $TMP0
024740  005037  001360                  CLR     $TMP0           ;&& CLEAR LOCATION TO USE AS ACCUMULATOR
024744  013746  001362                  MOV     $TMP1,-(SP)     ;&& SAVE $TMP1
024750  005037  001362                  CLR     $TMP1           ;&& CLEAR LOCATION TO USE AS 2ND ACCUMULATOR
024754  012701  024665                  MOV     #$DBLK+7,R1     ;&& MOVE ADDRESS OF LCD TO R2
024760  012702  025134                  MOV     #$NUMS+10,R2    ;&& MOVE ADDRESS+10 OF WORD STREAM OF 8^6 TO R2
024764  016600  000026          1$:     MOV     26(SP),R0       ;&& MOVE OVERFLOW LOCATION CONTENTS TO R0
024770  111137  001360                  MOVB    (R1),$TMP0      ;&& MOVE ASCII TO THE TEMPORARY LOCATION
024774  052737  000060  001360          BIS     #'0,$TMP0       ;&& MAKE LOCATION AN ASCII IF NOT ALREADY
025002  061237  001360          2$:     ADD     (R2),$TMP0      ;&& ADD THE NUMBER TO THE ASCII
025006  022737  000071  001360          CMP     #'9,$TMP0       ;&& HAVE WE SURPASSED ASCII '9'?
025014  002025                          BGE     4$              ;&& BRANCH IF NOT
025016  162737  000012  001360          SUB     #10.,$TMP0      ;&& SUBTRACT 10 FROM THE ASCII AND
025024  010103                          MOV     R1,R3           ;&& MOVE PRESENT ASCII ADDRESS TO R3
025026  005303                  3$:     DEC     R3              ;&& DECREMENT TO NEXT SIGNIFICANT ASCII DIGIT
025030  105213                          INCB    (R3)            ;&& ADD THE CARRY TO THE ASCII
025032  152713  000060                  BISB    #'0,(R3)        ;&& SET BITS TO MAKE IT A NUMBER ASCII
025036  122713  000071                  CMPB    #'9,(R3)        ;&& SEE IF CARRY NEEDS TO BE TRANSFERED
025042  002012                          BGE     4$              ;&& BRANCH IF NOT
025044  111337  001362                  MOVB    (R3),$TMP1      ;&& MOVE BYTE TO LOCATION $TMP1
025050  162737  000012  001362          SUB     #10.,$TMP1      ;&& SUBTRACT 10 DECIMAL AND
025056  113713  001362                  MOVB    $TMP1,(R3)      ;&& MOVE IT BACK
025062  022703  024656                  CMP     #$DBLK,R3       ;&& SEE IF ALL POSITIONS HAVE BEEN DONE
025066  001357                          BNE     3$              ;&& BRANCH BACK IF NOT
025070  005300                  4$:     DEC     R0              ;&& DECREMENT THE OVERFLOW LOCATION R0
025072  001343                          BNE     2$              ;&& BRANCH IF NOT ALL ADDED
025074  113711  001360                  MOVB    $TMP0,(R1)      ;&& MOVE NEW NUMBER TO LOCATION
025100  005301                          DEC     R1              ;&& POINT R1 TO THE NEXT ASCII LOCATION
025102  005742                          TST     -(R2)           ;&& POINT R2 TO NEXT DIGIT TO ADD
025104  022701  024661                  CMP     #$DBLK+3,R1     ;&& SEE IF ALL DIGITS HAVE BEEN ADDED
025110  001325                          BNE     1$              ;&& BRANCH IF NOT DONE
025112  012637  001362                  MOV     (SP)+,$TMP1     ;&& RESTORE $TMP1
025116  012637  001360                  MOV     (SP)+,$TMP0     ;&& RESTORE $TMP0
025122  000207                          RTS     PC              ;&& RETURN
025124  000003  000002  000007  $NUMS:  .WORD   3,2,7,6,8.
```

```
2814                                    .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.
025136  022737  000176  001340  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
025144  001112                          BNE     15$             ;;BRANCH IF NO
025146  105737  027423                  TSTB    CHARCT          ;&& SEE IF CHARACTER WAS INPUTED DURING PRINT
025152  001405                          BEQ     1$              ;&& BRANCH IF NOT
025154  113746  027423                  MOVB    CHARCT,-(SP)    ;&& MOVE CHARACTER TO THE STACK
025160  105037  027423                  CLRB    CHARCT          ;&& CLEAR THE CHARACTER FROM CHARCT
025164  000405                          BR      2$              ;&& GO CHECK IT OUT
025166  105777  154152          1$:     TSTB    @$TKS           ;;CHAR THERE?
025172  100077                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
025174  117746  154146                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
025200  042716  177600          2$:     BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
025204  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
025210  001404                          BEQ     3$              ;&& YES, BRANCH AROUND RETURN TO USER SETUP
025212  116637  177776  027423          MOVB    -2(SP),CHARCT   ;&& MOVE CHARACTER BACK TO CHARCT
025220  000464                          BR      15$             ;&& RETURN TO USER
025222  123727  001334  000001  3$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
025230  001460                          BEQ     15$             ;;BRANCH IF YES
025232  104401  025716                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
025236  104401  025723          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
025242  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
025246  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
025250  104401  025734                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
025254  105037  027423          19$:    CLRB    CHARCT          ;&& CLEAR ANY CHARACTER THAT WAS INPUTED DURING PRINT
025260  005046                          CLR     -(SP)           ;;CLEAR COUNTER
025262  005046                          CLR     -(SP)           ;;THE NEW SWR
025264  105777  154054          7$:     TSTB    @$TKS           ;;CHAR THERE?
025270  100375                          BPL     7$              ;;IF NOT TRY AGAIN
025272  117746  154050                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
025276  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
025302  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
025306  001005                          BNE     10$             ;;BRANCH IF NOT
025310  104401  025711                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
025314  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
025320  000746                          BR      $GTSWR          ;;LET'S TRY IT AGAIN
025322  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
025326  001022                          BNE     16$             ;;BRANCH IF NO
025330  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
025334  001403                          BEQ     11$             ;;BRANCH IF YES
025336  016677  000002  153774          MOV     2(SP),@SWR      ;;SAVE NEW SWR
025344  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
025350  104401  001405          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
025354  123727  001335  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
025362  001003                          BNE     15$             ;;BRANCH IF NOT
025364  012777  000100  153752          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
025372  000002          15$:            RTI                     ;;RETURN
025374  004737  023760          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
025400  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
025404  002420                          BLT     18$             ;;BRANCH IF YES
025406  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
```

```
025412  003015                      BGT     18$             ;;BRANCH IF YES
025414  042726  000060              BIC     #60,(SP)+       ;;STRIP-OFF ASCII
025420  005766  000002              TST     2(SP)           ;;IS THIS THE FIRST CHAR
025424  001403                      BEQ     17$             ;;BRANCH IF YES
025426  006316                      ASL     (SP)            ;;NO, SHIFT PRESENT
025430  006316                      ASL     (SP)            ;;   CHAR OVER TO MAKE
025432  006316                      ASL     (SP)            ;;   ROOM FOR NEW ONE.
025434  005266  000002      17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
025440  056616  177776              BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
025444  000707                      BR      7$              ;;GET THE NEXT ONE
025446  104401  001404      18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
025452  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
                                ;
025454  011646                 $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
025456  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
025464  105777  153654          1$:     TSTB    @$TKS           ;;WAIT FOR
025470  100375                          BPL     1$              ;;A CHARACTER
025472  117766  153650  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
025500  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
025506  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
025514  001013                          BNE     3$              ;;BRANCH IF NO
025516  105777  153622          2$.     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
025522  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
025524  117746  153616                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
025530  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
025534  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
025540  001366                          BNE     2$              ;;IF NOT DISCARD IT
025542  000750                          BR      1$              ;;YES, RESUME
025544  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
025552  002407                          BLT     4$              ;;BRANCH IF YES
025554  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
025562  003003                          BGT     4$              ;;BRANCH IF YES
025564  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
025572  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
025574  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
025576  012703  025702          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
025602  022703  025711          2$:     CMP     #$TTYIN+7.,R3   ;;BUFFER FULL?
025606  101405                          BLOS    4$              ;;BR IF YES
025610  104411                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
025612  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
025614  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
025620  001003                          BNE     3$              ;;SKIP IF NOT
025622  104401  001404          4$:     TYPE    ,$QUES          ;;TYPE A '?'
025626  000763                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
025630  111337  025700          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
025634  104401  025700                  TYPE    ,9$
025640  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
025644  001356                          BNE     2$              ;;LOOP IF NOT RETURN
025646  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
025652  104401  001406                  TYPE    ,$LF            ;;TYPE A LINE FEED
025656  012603                          MOV     (SP)+,R3        ;;RESTORE R3
025660  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
025662  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
025670  012766  025702  000004          MOV     #$TTYIN,4(SP)
025676  000002                          RTI                     ;;RETURN
025700     000              9$:     .BYTE   0                   ;;STORAGE FOR ASCII CHAR. TO TYPE
025701     000                      .BYTE   0                   ;;TERMINATOR
025702                      $TTYIN: .BLKB   7.                  ;;RESERVE 7. BYTES FOR TTY INPUT
025711     136     125     015  $CNTLU: .ASCIZ  /^U/<15><12>     ;;CONTROL "U"
025716     136     107     015  $CNTLG: .ASCIZ  /^G/<15><12>     ;;CONTROL "G"
025723     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
025734     040     040     116  $MNEW:  .ASCIZ  /  NEW = /
                                .EVEN
```

```
2815                                    .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
                                        ;
025746  011646                  $RDDEC: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR
025750  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE INPUT NUMBER
025756  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
025760  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
025762  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
025764  104412                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
025766  012600                          MOV     (SP)+,R0        ;;ADDRESS OF 1ST CHAR.
025770  010037  026114                  MOV     R0,6$           ;;SAVE INCASE OF BAD INPUT
025774  005046                          CLR     -(SP)           ;;CLEAR DATA WORD
025776  005002                          CLR     R2              ;;SIGN SET POSITIVE
026000  122710  000055                  CMPB    #'-,(R0)        ;;SEE IF A MINUS SIGN WAS TYPED
026004  001001                          BNE     2$              ;;BR IF NO MINUS SIGN
026006  112002                          MOVB    (R0)+,R2        ;;SAVE FOR LATER USE
026010  112001                  2$:     MOVB    (R0)+,R1        ;;PICKUP THIS CHARACTER
026012  001424                          BEQ     3$              ;;GET OUT IF ZERO
026014  122701  000060                  CMPB    #'0,R1          ;;MAKE SURE THIS CHARACTER
026020  003032                          BGT     5$              ;;IS A DIGIT BETWEEN 0 & 9
026022  122701  000071                  CMPB    #'9,R1
026026  002427                          BLT     5$
026030  032716  170000                  BIT     #^C7777,(SP)    ;;DON'T LET NUMBER GET TO BIG
026034  001024                          BNE     5$              ;;BR IF NUMBER WOULD OVERFLOW
026036  006316                          ASL     (SP)            ;;*2
026040  011646                          MOV     (SP),-(SP)      ;;SAVE FOR LATER
026042  006316                          ASL     (SP)            ;;*4
026044  006316                          ASL     (SP)            ;;*8.
026046  062616                          ADD     (SP)+,(SP)      ;;*10.
026050  102416                          BVS     5$              ;;OVERFLOW ISN'T ALLOWED
026052  162701  000060                  SUB     #'0,R1          ;;STRIP AWAY THE ASCII JUNK
026056  060116                          ADD     R1,(SP)         ;;ADD IN THIS DIGIT
026060  102412                          BVS     5$              ;;OVERFLOW ISN'T ALLOWED
026062  000752                          BR      2$              ;;LOOP
026064  005702                  3$:     TST     R2              ;;CHECK IF NUMBER IS NEG
026066  001401                          BEQ     4$              ;;BR IF NO
026070  005416                          NEG     (SP)            ;;YES--NEGATE THE NUMBER
026072  012666  000012          4$:     MOV     (SP)+,12(SP)    ;;SAVE THE RESULT
026076  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
026100  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
026102  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
026104  000002                          RTI                     ;;RETURN
026106  005726                  5$:     TST     (SP)+           ;;CLEAN PARTIAL NUMBER FROM STACK
026110  105010                          CLRB    (R0)            ;;SET A TERMINATOR
026112  104401                          TYPE                    ;;TYPE THE INPUT UP TO BAD CHAR.
026114  000000                  6$:     .WORD   0               ;;POINTER GOES HERE
026116  104401  001404                  TYPE    ,$QUES          ;;"?" "CR" &"LF"
026122  000720                          BR      1$              ;;TRY AGAIN
```

```
2816                                    .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
026124  011646                 $RDOCT: MOV     (SP),-(SP)              ;;PROVIDE SPACE FOR THE
026126  016666  000004  000002          MOV     4(SP),2(SP)             ;;INPUT NUMBER
026134  010046                         MOV     R0,-(SP)                ;;PUSH R0 ON STACK
026136  010146                         MOV     R1,-(SP)                ;;PUSH R1 ON STACK
026140  010246                         MOV     R2,-(SP)                ;;PUSH R2 ON STACK
026142  104412                 1$:     RDLIN                           ;;READ AN ASCIZ LINE
026144  012600                         MOV     (SP)+,R0                ;;GET ADDRESS OF 1ST CHARACTER
026146  010037  026252                 MOV     R0,5$                   ;;AND SAVE IT
026152  005001                         CLR     R1                      ;;CLEAR DATA WORD
026154  005002                         CLR     R2
026156  112046                 2$:     MOVB    (R0)+,-(SP)             ;;PICKUP THIS CHARACTER
026160  001420                         BEQ     3$                      ;;IF ZERO GET OUT
026162  122716  000060                 CMPB    #'0,(SP)                ;;MAKE SURE THIS CHARACTER
026166  003026                         BGT     4$                      ;;IS AN OCTAL DIGIT
026170  122716  000067                 CMPB    #'7,(SP)
026174  002423                         BLT     4$
026176  006301                         ASL     R1                      ;;*2
026200  006102                         ROL     R2
026202  006301                         ASL     R1                      ;;*4
026204  006102                         ROL     R2
026206  006301                         ASL     R1                      ;;*8
026210  006102                         ROL     R2
026212  042716  177770                 BIC     #^C7,(SP)               ;;STRIP THE ASCII JUNK
026216  062601                         ADD     (SP)+,R1                ;;ADD IN THIS DIGIT
026220  000756                         BR      2$                      ;;LOOP
026222  005726                 3$:     TST     (SP)+                   ;;CLEAN TERMINATOR FROM STACK
026224  010166  000012                 MOV     R1,12(SP)               ;;SAVE THE RESULT
026230  010237  026262                 MOV     R2,$HIOCT
026234  012602                         MOV     (SP)+,R2                ;;POP STACK INTO R2
026236  012601                         MOV     (SP)+,R1                ;;POP STACK INTO R1
026240  012600                         MOV     (SP)+,R0                ;;POP STACK INTO R0
026242  000002                         RTI                             ;;RETURN
026244  005726                 4$:     TST     (SP)+                   ;;CLEAN PARTIAL FROM STACK
026246  105010                         CLRB    (R0)                    ;;SET A TERMINATOR
026250  104401                         TYPE                            ;;TYPE UP THRU THE BAD CHAR.
026252  000000                 5$:     .WORD   0
026254  104401  001404                 TYPE    ,$QUES                  ;;"?" "CR" & "LF"
026260  000730                         BR      1$                      ;;TRY AGAIN
026262  000000                 $HIOCT: .WORD   0                       ;;HIGH ORDER BITS GO HERE
```

```
2817                                    .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.
026264  010046                          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
026266  016600  000002                          MOV     2(SP),R0        ;;GET TRAP ADDRESS
026272  005740                          $TRAP:  TST     -(R0)           ;;BACKUP BY 2
026274  111000                          $TRAP:  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
026276  006300                          $TRAP:  ASL     R0              ;;POSITION FOR INDEXING
026300  016000  026320                          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
026304  000200                          $TRAP:  RTS     R0              ;;GO TO ROUTINE
                                        ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
026306  011646                          $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
026310  016666  000004  000002                  MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
026316  000002                          $TRAP:  RTI                     ;;RESTORE THE PSW
                                        .SBTTL  TRAP TABLE
                                        ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                        ;*BY THE "TRAP" INSTRUCTION.
                                        ;       ROUTINE
                                        ;       -------
026320  026306                          $TRPAD: .WORD   $TRAP2
026322  023546                                  $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
026324  024140                                  $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
026326  024114                                  $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
026330  024154                                  $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
026332  024352                                  $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
026334  024342                                  $TYPDE  ;;CALL=TYPDE    TRAP+6(104406)  TYPE DECIMAL NUMBER GREATER THAN 32767
026336  025236                                  $GTSWR  ;;CALL=GTSWR    TRAP+7(104407)  GET SOFT-SWR SETTING
026340  025136                                  $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
026342  025454                                  $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
026344  025574                                  $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
026346  026124                                  $RDOCT  ;;CALL=RDOCT    TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
026350  025746                                  $RDDEC  ;;CALL=RDDEC    TRAP+14(104414) READ A DECIMAL NUMBER FROM TTY
```

```
2818                                    .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
026352  032777  001000  152760  $SCOPE: BIT     #BIT09,@SWR         ;;LOOP ON ERROR?
026360  001406                          BEQ     1$                  ;;BR IF NO
026362  005737  001312                  TST     $ERTTL              ;;SEE IF THERE WERE ANY ERRORS YET
026366  001403                          BEQ     1$                  ;;BR IF NOT YET
026370  013716  001310                  MOV     $LPERR,(SP)         ;;FUDGE RETURN
026374  000002                          RTI                         ;;RETURN
026376  032777  040000  152734  1$:     BIT     #BIT14,@SWR         ;;LOOP ON TEST?
026404  001403                          BEQ     2$                  ;;BR IF NO
026406  013716  001306                  MOV     $LPADR,(SP)         ;;FUDGE RETURN
026412  000002                          RTI                         ;;RETURN
026414  105737  002710          2$:     TSTB    EOPLOC              ;&& TEST TO SEE IF EOP'S ARE TO PRINT
026420  001043                          BNE     6$                  ;&& BRANCH IF NOT
026422  105737  027423                  TSTB    CHARCT              ;&& SEE IF A CHARACTER WAS INPUTED IN A PRINT ROUTINE
026426  001406                          BEQ     3$                  ;&& BRANCH TO CHECK KEYBOARD IF NOT
026430  113737  027423  002710          MOVB    CHARCT,EOPLOC       ;&& MOVE THE CHARACTER TO EOPLOC
026436  105037  027423                  CLRB    CHARCT              ;&& CLEAR THE LOCATION
026442  000411                          BR      4$                  ;&& BRANCH TO CHECK THE CHARACTER
026444  105777  152674          3$:     TSTB    @$TKS               ;&& SEE IF EOP REQUESTED
026450  100055                          BPL     8$                  ;&& BRANCH IF NOT
026452  117737  152670  002710          MOVB    @$TKB,EOPLOC        ;&& GET CHARACTER
026460  142737  000200  002710          BICB    #200,EOPLOC         ;&& CLEAR PARITY BIT
026466  122737  000030  002710  4$:     CMPB    #30,EOPLOC          ;&& SEE IF A (^X)
026474  001406                          BEQ     5$                  ;&& BRANCH IF IT IS
026476  113737  002710  027423          MOVB    EOPLOC,CHARCT       ;&& MOVE CHARACTER BACK TO CHARCT FOR POSSIBLE FUTURE USE
026504  105037  002710                  CLRB    EOPLOC              ;&& CLEAR THE LOCATION
026510  000435                          BR      8$                  ;&& BRANCH TO START SCOPE ROUTINE
026512  104401  027042          5$:     TYPE    ,HAKTPM             ;&& TYPE: 'HIT ANY KEY TO OBTAIN A PROGRESS REPORT,'
                                                                    ;&&       'ENTER (^X) TO RESUME EOP'S AND EOD'S'
                                                                    ;&&       'ENTER THE KEY REPEATEDLY, AS RESETS DONE IN THE D
                                                                    ;&&       'MAY CLEAR THE CHARACTER BEFORE THE TESTS FOR THE
026516  105337  002710                  DECB    EOPLOC              ;&& MAKE ^X ANOTHER CHARACTER AND
026522  105037  027423                  CLRB    CHARCT              ;&& CLEAR ANY CHARACTER THAT MAY BE THERE
026526  000426                          BR      8$                  ;&& BRANCH TO START SCOPE ROUTINE
026530  105737  027423          6$:     TSTB    CHARCT              ;&& SEE IF A CHARACTER WAS INPUTED IN THE TYPE ROUTINE
026534  001011                          BNE     7$                  ;&& GO TEST THE CHARACTER IF SO
026536  105777  152602                  TSTB    @$TKS               ;&& SEE IF CHARACTER WAITING
026542  100020                          BPL     8$                  ;&& BRANCH IF NOT
026544  117737  152576  027423          MOVB    @$TKB,CHARCT        ;&& MOVE THE CHARACTER FOR TESTING
026552  142737  000200  027423          BICB    #200,CHARCT         ;&& CLEAR THE PARITY BIT
026560  122737  000030  027423  7$:     CMPB    #30,CHARCT          ;&& SEE IF ANOTHER (^X) WAS INPUTED
026566  001006                          BNE     8$                  ;&& BRANCH IF NOT
026570  105037  002710                  CLRB    EOPLOC              ;&& CLEAR EOPLOC TO RESUME EOP MESSAGES
026574  104401  027352                  TYPE    ,EOPRSM             ;&& TYPE: 'EOP'S AND EOD'S WILL RESUME PRINTING'
026600  105037  027423                  CLRB    CHARCT              ;&& MAKE CHARACTER SOMETHING ELSE
026604  105737  027423          8$:     TSTB    CHARCT              ;&& SEE IF A CHARACTER IS WAITING
026610  001410                          BEQ     9$                  ;&& BRANCH AROUND (^Y) TEST IF NOT
```

```
026612  122737  000031  027423          CMPB    #31,CHARCT      ;&& IS THIS A (^Y) (REQUEST FOR RUN SUMMARY)
026620  001004                          BNE     9$              ;&& BRANCH IF NOT
026622  004737  003170                  JSR     PC,PTCAPT       ;&& GO TO SUBROUTINE TO PRINT SUMMARY
026626  105037  027423                  CLRB    CHARCT          ;&& CLEAR THE CHARCT LOCATION SO SUMMARY NOT REPEATED
026632  104410                  9$:     CKSWR                   ;;TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER
                                ;#####START OF CODE FOR THE XOR TESTER#####
026634  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
026636  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
026642  012737  026662  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
026650  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
026654  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
026660  000444                          BR      $SVLAD          ;;GO TO THE NEXT TEST
026662  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
026664  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
026670  000432                          BR      7$              ;;LOOP ON THE PRESENT TEST
026672                          6$:;#####END OF CODE FOR THE XOR TESTER#####
026672  032777  000400  152440          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
026700  001432                          BEQ     4$              ;;BR IF NO
026702  005046                          CLR     -(SP)           ;;CLEAR A TEMP. LOCATION
026704  117716  152430                  MOVB    @SWR,(SP)       ;;PICKUP THE DESIRED TEST NUMBER
026710  042716  000200                  BIC     #$SWRMK,(SP)    ;;MASK OUT UNDESIRED BITS
026714  001416                          BEQ     8$              ;;BRANCH IF BAD TEST NUMBER IN SWR
026716  022716  000037                  CMP     #37,(SP)        ;;CHECK THE NUMBER IN THE SWR
026722  002413                          BLT     8$              ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
026724  011637  001302                  MOV     (SP),$TSTNM     ;;UPDATE THE TEST NUMBER IN $TSTNM
026730  011637  001414                  MOV     (SP),$TESTN     ;&& UPDATE THE TEST NUMBER IN $TESTN
026734  005316                          DEC     (SP)            ;;BACKUP BY ONE
026736  006316                          ASL     (SP)            ;;SCALE THE TEST NUMBER AS AN INDEX
026740  062716  027466                  ADD     #$SW08TBL,(SP)  ;;FORM THE ADDRESS OF TEST POINTER
026744  013637  001306                  MOV     @(SP)+,$LPADR   ;;SET LOOP ADDRESS TO DESIRED TEST
026750  000426                          BR      $OVER           ;;GO LOOP ON THE TEST
026752  005726                  8$:     TST     (SP)+           ;;CLEAN THE BAD TEST NUMBER OFF OF THE STACK
026754                          2$:     ;TSTB   $ERFLG          ;;HAS AN ERROR OCCURRED? ;&& ELIMINATED FOR CZDRLB
026754  001406                          BEQ     $SVLAD          ;;BR IF NO
026756  013737  001310  001306  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
026764  000420                          BR      $OVER
026766  105037  001303          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
026772  105237  001302          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
026776  113737  001302  001414          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
027004  011637  001306                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
027010  011637  001310                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
027014  005037  001376                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
027020  112737  000001  001315          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
027026  013777  001302  152306  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
027034  013716  001306                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
027040  000002                          RTI                     ;;RETURN
027042     136     130     200  HAK'PM: .ASCII  /^X/<CRLF>/HIT ANY KEY TO OBTAIN A PROGRESS REPORT,/<CRLF> ;&&
027116     105     116     124          .ASCII  /ENTER (^X) TO RESUME EOP'S AND EOD'S/<CRLF>                ;&&
027163     105     116     124          .ASCII  /ENTER THE KEY REPEATEDLY, AS RESETS DONE IN THE DIAGNOSTIC/<CRLF> ;&&
027256     115     101     131          .ASCIZ  /MAY CLEAR THE CHARACTER BEFORE THE TESTS FOR THE CHARACTER/<CRLF> ;&&
027352     136     130     200  EOPRSM: .ASCIZ  /^X/<CRLF>/EOP'S AND EOD'S WILL RESUME PRINTING/<CRLF>     ;&&
027423     000                  CHARCT: .BYTE   0               ;&& LOCATION TO HOLD INPUTED CHARACTER
027424     040     040     124  TESLR:  .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /     ;&& ERROR MESSAGE FOR $EOP
                                        .EVEN
```

```
027466                          $SW08TBL:
027466  011246                          .WORD   TST1+2          ;STARTING ADDRESS+2 OF TEST 1
027470  011416                          .WORD   TST2+2          ;STARTING ADDRESS+2 OF TEST 2
027472  011520                          .WORD   TST3+2          ;STARTING ADDRESS+2 OF TEST 3
027474  012032                          .WORD   TST4+2          ;STARTING ADDRESS+2 OF TEST 4
027476  012574                          .WORD   TST5+2          ;STARTING ADDRESS+2 OF TEST 5
027500  012742                          .WORD   TST6+2          ;STARTING ADDRESS+2 OF TEST 6
027502  013256                          .WORD   TST7+2          ;STARTING ADDRESS+2 OF TEST 7
027504  013406                          .WORD   TST10+2         ;STARTING ADDRESS+2 OF TEST 10
027506  013614                          .WORD   TST11+2         ;STARTING ADDRESS+2 OF TEST 11
027510  013752                          .WORD   TST12+2         ;STARTING ADDRESS+2 OF TEST 12
027512  014110                          .WORD   TST13+2         ;STARTING ADDRESS+2 OF TEST 13
027514  014246                          .WORD   TST14+2         ;STARTING ADDRESS+2 OF TEST 14
027516  014410                          .WORD   TST15+2         ;STARTING ADDRESS+2 OF TEST 15
027520  014614                          .WORD   TST16+2         ;STARTING ADDRESS+2 OF TEST 16
027522  015024                          .WORD   TST17+2         ;STARTING ADDRESS+2 OF TEST 17
027524  015374                          .WORD   TST20+2         ;STARTING ADDRESS+2 OF TEST 20
027526  015752                          .WORD   TST21+2         ;STARTING ADDRESS+2 OF TEST 21
027530  016240                          .WORD   TST22+2         ;STARTING ADDRESS+2 OF TEST 22
027532  016510                          .WORD   TST23+2         ;STARTING ADDRESS+2 OF TEST 23
027534  017220                          .WORD   TST24+2         ;STARTING ADDRESS+2 OF TEST 24
027536  017474                          .WORD   TST25+2         ;STARTING ADDRESS+2 OF TEST 25
027540  017736                          .WORD   TST26+2         ;STARTING ADDRESS+2 OF TEST 26
027542  020166                          .WORD   TST27+2         ;STARTING ADDRESS+2 OF TEST 27
027544  020356                          .WORD   TST30+2         ;STARTING ADDRESS+2 OF TEST 30
027546  020616                          .WORD   TST31+2         ;STARTING ADDRESS+2 OF TEST 31
027550  020720                          .WORD   TST32+2         ;STARTING ADDRESS+2 OF TEST 32
027552  021252                          .WORD   TST33+2         ;STARTING ADDRESS+2 OF TEST 33
027554  021530                          .WORD   TST34+2         ;STARTING ADDRESS+2 OF TEST 34
027556  022064                          .WORD   TST35+2         ;STARTING ADDRESS+2 OF TEST 35
027560  022334                          .WORD   TST36+2         ;STARTING ADDRESS+2 OF TEST 36
027562  022622                          .WORD   TST37+2         ;STARTING ADDRESS+2 OF TEST 37
```

```
 2819                                   .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
     027564                             $ERROR:
     027564  104410                             CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
     027566  105237  001303            7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
     027572  001775                             BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
     027574  013777  001302  151540            MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
     027602  032777  002000  151530            BIT     #BIT10,@SWR     ;;BELL ON ERROR?
     027610  001402                             BEQ     1$              ;;NO - SKIP
     027612  104401  001400                     TYPE    ,$BELL          ;;RING BELL
     027616  005237  001312            1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
     027622  005237  002716                     INC     ERRCNT          ;&& INCREMENT THE ERROR COUNT
     027626  011637  001316                     MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
     027632  162737  000002  001316            SUB     #2,$ERRPC
     027640  117737  151452  001314            MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
     027646  032777  020000  151464            BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
     027654  001002                             BNE     20$             ;;SKIP TYPEOUTS
     027656  004737  027764                     JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
     027662                             20$:
     027662  122737  000001  001432            CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
     027670  001007                             BNE     2$              ;;NO,SKIP APT ERROR REPORT
     027672  113737  001314  027704            MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
     027700  004737  030260                     JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
     027704     000             21$:    .BYTE   0
     027705     000                     .BYTE   0
     027706  000777             22$:    BR      22$             ;;APT ERROR LOOP
     027710  005777  151424            2$:     TST     @SWR            ;;HALT ON ERROR
     027714  100002                             BPL     3$              ;;SKIP IF CONTINUE
     027716  000000                             HALT                    ;;HALT ON ERROR!
     027720  104410                             CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
     027722  005737  001376            3$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
     027726  001402                             BEQ     4$              ;;BR IF NONE
     027730  013716  001376                     MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
     027734  032777  001000  151376    4$:     BIT     #BIT9,@SWR      ;&& SEE IF WE ARE TO LOOP ON ERROR
     027742  001402                             BEQ     5$              ;&& BRANCH OUT IF NOT
     027744  013716  001310                     MOV     $LPERR,(SP)     ;&& FUDGE RETURN
     027750                             5$:
     027750  022737  023464  000042            CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
     027756  001001                             BNE     6$              ;;BRANCH IF NO
     027760  000000                             HALT                    ;;YES
     027762                             6$:
     027762  000002                             RTI                     ;;RETURN
```

```
 2820                                   .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.
     027764                             $ERRTYP:
     027764  010046                            MOV     R0,-(SP)        ;;SAVE R0
     027766  005000                            CLR     R0              ;;CLEAR R0 TO RECEIVE ITEM INDEX
     027770  113700  001314                    MOVB    @#$ITEMB,R0     ;;PICKUP THE ITEM INDEX
     027774  001004                            BNE     1$              ;;IF ITEM NUMBER IS ZERO, TYPE THE PC OF THE ERROR
     027776  013746  001316                    MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                        ;;ERROR ADDRESS
     030002  104402                            TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
     030004  000513                            BR      14$             ;;GET OUT
     030006  010037  001360          1$:       MOV     R0,$TMP0        ;;@@ MOVE R0 TO $TMP0 FOR 200 TEST
     030012  042700  000200                    BIC     #200,R0         ;;@@ CLEAR BIT 7 IF PRESENT
     030016  005300                            DEC     R0              ;;@@ MAKE POINTER AN INDEX
     030020  006300                            ASL     R0              ;;@@ SHIFT TO MULTIPLY BY 10 (OCTAL)
     030022  006300                            ASL     R0              ;;@@
     030024  006300                            ASL     R0              ;;@@
     030026  105737  001360                    TSTB    $TMP0           ;;@@ SEE IF ITEM NUMBER IS OVER 200
     030032  100041                            BPL     4$              ;;@@ BRANCH IF ITEM NUMBER IS LESS THAN 200
     030034  023727  002716  000020            CMP     ERRCNT,#20      ;;@@ SEE IF 20 (OCTAL) ERRORS HAVE PRINTED
     030042  002404                            BLT     2$              ;;@@ BRANCH TO PRINT THE ERROR IF LESS
     030044  003073                            BGT     14$             ;;@@ BRANCH TO RETURN IF GREATER - NO MORE DATA IS TO PRINT
     030046  104401  035377                    TYPE    ,NOMORE         ;;@@ TYPE MESSAGE ANNOUNCING NO MORE PRINTING OF ERRORS
     030052  000470                            BR      14$             ;;@@ BRANCH TO RETURN
     030054  022737  000001  002716  2$:       CMP     #1,ERRCNT       ;;@@ SEE IF THIS IS THE FIRST ERROR
     030062  001415                            BEQ     3$              ;;@@ BRANCH IF IT WAS AND GO TYPE ERROR MESSAGE
     030064  123737  001360  030240            CMPB    $TMP0,MEPITM    ;;@@ SEE IF ITEM MATCHES LAST MULTIPLE ERROR
     030072  001011                            BNE     3$              ;;@@ BRANCH IF NOT - NEW HEADER NEEDED
     030074  032777  000200  151236            BIT     #BIT7,@SWR      ;;@@ SEE IF SWITCH REGISTER BIT 7 IS SET
     030102  001054                            BNE     14$             ;;@@ BRANCH TO RETURN IF SWITCH SET
     030104  042700  177400                    BIC     #177400,R0      ;;@@ CLEAR UPPER BYTE OF R0 EXPOSING ITEM BYTE
     030110  062700  002302                    ADD     #ER200+4,R0     ;;@@ POINT TO DATA TABLE ENTRY
     030114  000434                            BR      9$              ;;@@ BRANCH TO PRINT DATA
     030116  113737  001360  030240  3$:       MOVB    $TMP0,MEPITM    ;;@@ MOVE ITEM NUMBER TO MEPITM FOR POSSIBLE FUTURE USE
     030124  042700  177000                    BIC     #177000,R0      ;;@@ CLEAR UPPER BYTE OF R0
     030130  062700  000540                    ADD     #ER200-$ERRTB,R0;;@@ ADD 200 BASE POINTER TO R0 AND
     030134  000402                            BR      5$              ;;@@ BRANCH AROUND ERRCNT CLEAR
     030136  005037  002716          4$:       CLR     ERRCNT          ;;@@ CLEAR ERRCNT SO MULTIPLE ERRORS GET NEW HEADER
     030142  104401  001405          5$:       TYPE    ,$CRLF          ;;@@ TYPE  <CRLF>
     030146  062700  001536                    ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
     030152  012037  030162                    MOV     (R0)+,6$        ;;PICKUP "ERROR MESSAGE" POINTER
     030156  001404                            BEQ     7$              ;;SKIP TYPEOUT IF NO POINTER
     030160  104401                            TYPE                    ;;TYPE THE "ERROR MESSAGE"
     030162  000000          6$:       .WORD   0                       ;;"ERROR MESSAGE" POINTER GOES HERE
     030164  104401  001405                    TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
     030170  012037  030200          7$:       MOV     (R0)+,8$        ;;PICKUP "DATA HEADER" POINTER
     030174  001404                            BEQ     9$              ;;SKIP TYPEOUT IF 0
     030176  104401                            TYPE                    ;;TYPE THE "DATA HEADER"
     030200  000000          8$:       .WORD   0                       ;;"DATA HEADER" POINTER GOES HERE
     030202  104401  001405                    TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
     030206  011000          9$:       MOV     (R0),R0                 ;;PICKUP "DATA TABLE" POINTER
     030210  001407                            BEQ     13$             ;;GO AROUND ROUTINE TO TYPE THE DATA IF NONE
     030212  013046          10$:      MOV     @(R0)+,-(SP)            ;;PUT OCTAL DATA ON STACK FOR TYPING
     030214  104402                            TYPOC                   ;;TYPE AN OCTAL NUMBER
```

```
030216  005710                          TST     (R0)            ;;IS THERE ANOTHER NUMBER?
030220  001403                          BEQ     13$             ;;BR IF NO
030222  104401  033764                  TYPE    ,SPACES         ::TYPE TWO(2) SPACES
030226  000771                          BR      10$               ;;0 BACK TO PRINT THE OCTAL NUMBER
030230  104401  001405          13$:    TYPE    ,$CRLF          ;;''CARRIAGE RETURN'' & ''LINE FEED''
030234  012600                  14$:    MOV     (SP)+,R0        ;;RESTORE P0
030236  000207                          RTS     PC              ;;RETURN
030240  000000                  MEPITM: .WORD   0               ;&& LOCATION TO STORE 200+ ERROR ITEM NUMBER
```

```
2821                                    .SBTTL  APT COMMUNICATIONS ROUTINE
                                        ;*****************************************************************
030242  112737  000001  030506  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
030250  112737  000001  030504  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
030256  000403                        BR      $ATYC
030260  112737  000001  030506  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
030266                          $ATYC:
030266  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
030270  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
030272  105737  030504                TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
030276  001450                        BEQ     5$              ;;IF NOT:  BR
030300  122737  000001  001432        CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
030306  001031                        BNE     3$              ;;IF NOT:  BR
030310  132737  000100  001433        BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
030316  001425                        BEQ     3$              ;;IF NOT:  BR
030320  017600  000004                MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
030324  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
030332  005737  001410          1$:   TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
030336  001375                        BNE     1$              ;;IF NOT:  WAIT
030340  010037  001426                MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
030344  105720                  2$:   TSTB    (R0)+           ;;FIND END OF MESSAGE
030346  001376                        BNE     2$
030350  163700  001426                SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
030354  006200                        ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
030356  010037  001430                MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
030362  012737  000004  001410        MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
030370  000413                        BR      5$
030372  017637  000004  030416  3$:   MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
030400  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
030406  013746  177776                MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
030412  004737  023546                JSR     PC,$TYPE        ;;CALL TYPE MACRO
030416  000000                  4$:   .WORD   0
030420                          5$:
030420  105737  030506          10$:  TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
030424  001416                        BEQ     12$             ;;IF NOT:  BR
030426  005737  001432                TST     $ENV            ;;RUNNING UNDER APT?
030432  001413                        BEQ     12$             ;;IF NOT:  BR
030434  005737  001410          11$:  TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
030440  001375                        BNE     11$             ;;IF NOT:  WAIT
030442  017637  000004  001412        MOV     @4(SP),$FATAL   ;;GET ERROR #
030450  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
030456  005237  001410                INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
030462  105037  030506          12$:  CLRB    $FFLG           ;;CLEAR FATAL FLAG
030466  105037  030505                CLRB    $LFLG           ;;CLEAR LOG FLAG
030472  105037  030504                CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
030476  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
030500  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
030502  000207                        RTS     PC              ;;RETURN
030504     000                $MFLG:  .BYTE   0               ;;MESSG. FLAG
030505     000                $LFLG:  .BYTE   0               ;;LOG FLAG
030506     000                $FFLG:  .BYTE   0               ;;FATAL FLAG
                                        .EVEN
            000200                  APTSIZE=200
            000001                  APTENV=001
            000100                  APTSPOOL=100
            000040                  APTCSUP=040
2822                                    .SBTTL  POWER DOWN AND UP ROUTINE
```

```
                                    ;*****************************************************************
                                    ;POWER DOWN AND UP ROUTINE
030510  012737  030526  000024  $PWRDN: MOV     #$PWRUP,PWRVEC  ;&& SET UP VECTOR TO RETURN TO THE HALT BELOW
030516  012737  000340  000026          MOV     #LEVEL7,PWRVEC+2;&& RETURN PRIORITY TO 7
030524  000000                        HALT                    ;;HALT PROCESSOR
030526  012737  030510  000024  $PWRUP: MOV     #$PWRDN,PWRVEC  ;&& RESET PWRVEC TO PWRDN ROUTINE AND
030534  012737  000340  000026          MOV     #LEVEL7,PWRVEC+2;&& PRIORITY TO 7
030542  012706  001300                  MOV     #STACK,SP       ;&& REINITIALIZE THE STACK,
030546  012746  000340                  MOV     #LEVEL7,-(SP)   ;&& SET UP RETURN PRIORITY TO 7 AND
030552  012746  000210                  MOV     #STAGIN,-(SP)   ;&& MOVE STAGIN ADDRESS TO STACK AND
030556  005037  001360                  CLR     $TMP0           ;&& CLEAR WAIT LOOP COUNTER
030562  005237  001360          1$:     INC     $TMP0           ;&& GIVE TTY TIME TO RECOVER FROM POWER FAILURE
030566  001375                  BNE     1$                      ;&& BRANCH BACK UNTIL ZERO AGAIN
030570  104401  030576                  TYPE    ,$POWER         ;&& TYPE THE POWER FAILURE MESSAGE ASCIZED BELOW
030574  000002                  RTI                             ;&& RETURN TO PROGRAM
030576     200     120     117  $POWER: .ASCIZ  <CRLF>/POWER FAILURE - RESTARTING PROGRAM/<CRLF>
                                        .EVEN
```

```
                                        .SBTTL  MULTIPLE BOARD DIALOGUE ROUTINE
                                ;*****************************************************************
                                ;>>>>>>>MULTIPLE BOARD DIALOGUE ROUTINE<<<<<<<
                                ;*****************************************************************
030644  012706  001300          MBD:    MOV     #STACK,SP       ;INITIALIZE THE STACK
030650  004737  005660                  JSR     PC,SETTUP       ;GO INITIALIZE THE COMMON TAGS
030654  104401  035245                  TYPE    ,MBDIAL         ;TYPE:  'MULTIPLE BOARD DIALOGUE'
030660  105037  027423          PROMPT: CLRB    CHARCT          ;CLEAR LOCATION FOR POSSIBLE INPUT DURING PRINT
030664  104401  035300                  TYPE    ,ECELR          ;TYPE:  'ENTER COMMAND ([E]DIT, [L]IST, [B]URST CALIBRATION,
030670  012703  000001                  MOV     #1,R3           ;EXPECT 1 CHARACTER
030674  004737  002722                  JSR     PC,READ         ;GO READ 1 CHARACTER
030700  022737  000114  002666          CMP     #'L,ANSWER      ;LIST PRESENT TABLE?
030706  001073                          BNE     1$              ;BRANCH IF NO
030710  104401  035054                  TYPE    ,HEADER         ;TYPE:  '# OF    START                                  2-N
                                                                ;       'BOARDS  REGADR  VECADR   W-B     P-LEV   CYCLE    T
030714  013737  001476  002720          MOV     $DDW0,DDW       ;SET UP THE DEVICE DESCRIPTOR WORD FOR PRINTING
030722  013746  002416                  MOV     QTYBRD,-(SP)    ;MOVE NUMBER OF DEVICES TO STACK FOR TYPING
030726  104405                          TYPDS                   ;TYPE THE NUMBER OF DEVICES
030730  104401  033767                  TYPE    ,SPACE3         ;TYPE 3 SPACE CHARACTERS
030734  013746  002420                  MOV     REGADR,-(SP)    ;MOVE THE DEVICE REGISTER ADDRESS TO THE STACK
030740  104402                          TYPOC                   ;TYPE THE DEVICE REGISTER ADDRESS
030742  104401  033773                  TYPE    ,SPACE6         ;TYPE 6 SPACE CHARACTERS
030746  013746  002460                  MOV     VECADR,-(SP)    ;MOVE THE DEVICE VECTOR ADDRESS TO THE STACK
030752  104403                          TYPOS                   ;TYPE THE DEVICE VECTOR ADDRESS
030754     003     000                  .BYTE   3,0             ;TYPE 3 CHARACTERS, LEADING ZEROS SUPPRESSED
030756  104401  034002                  TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
030762  032737  000001  002720          BIT     #BIT0,DDW       ;SEE WHICH W/B STATE FOR BOARDS
030770  001403                          BEQ     10$             ;GO PRINT W STATE IF W
030772  104401  034017                  TYPE    ,B              ;TYPE A 'B'
030776  000402                          BR      11$             ;GO TO NEXT CHECK
031000  104401  034021          10$:    TYPE    ,W              ;TYPE A 'W'
031004  104401  034002          11$:    TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
031010  004737  005630                  JSR     PC,PNTPRI       ;PRINT DEVICE PRIORITY
031014  104401  034002                  TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
031020  032737  000002  002720          BIT     #BIT1,DDW       ;SEE WHICH 2/N STATE FOR BOARDS
031026  001003                          BNE     12$             ;GO PRINT N STATE IF N
031030  104401  034032                  TYPE    ,TWO            ;TYPE A '2'
031034  000402                          BR      13$             ;GO TO NEXT CHECK
031036  104401  034034          12$:    TYPE    ,N              ;TYPE AN 'N'
031042  104401  034002          13$:    TYPE    ,SPACE7         ;TYPE 7 SPACE CHARACTERS
031046  032737  000004  002720          BIT     #BIT2,DDW       ;SEE WHICH CABLE STATE FOR BOARDS
031054  001403                          BEQ     14$             ;GO PRINT NO CABLE IF NONE
031056  104401  034026                  TYPE    ,YES            ;TYPE 'YES'
031062  000402                          BR      15$             ;BRANCH TO CONTINUE
031064  104401  034023          14$:    TYPE    ,NO             ;TYPE 'NO'
031070  104401  034047          15$:    TYPE    ,CRLF2          ;TYPE 2 <CRLF>'S
031074  000671                          BR      PROMPT          ;BRANCH TO PROMPT ANOTHER COMMAND
031076  022737  000122  002666  1$:     CMP     #'R,ANSWER      ;RUN PROGRAM?
031104  001020                          BNE     4$              ;BRANCH IF NOT
031106  005737  002420                  TST     REGADR          ;SEE IF REGADR HAS BEEN LOADED
031112  001003                          BNE     3$              ;BRANCH TO CHECK VECADR IF SO
031114  104401  033504          2$:     TYPE    ,MUSTED         ;TYPE:  'DEVICE ADDRESS AND/OR VECTOR TABLE NOT SET UP - MUS
031120  000657                          BR      PROMPT          ;BRANCH BACK FOR PROMPT MESSAGE
031122  005737  002460          3$:     TST     VECADR          ;SEE IF VECADR HAS BEEN LOADED
031126  001772                          BEQ     2$              ;BRANCH BACK TO PRINT ERROR MESSAGE IF NOT
031130  004737  003360                  JSR     PC,FIXTBL       ;FILL TABLE
031134  012737  177777  002672          MOV     #-1,MANSIZE     ;MOVE -1 TO MANSIZE TO INDICATE WE HAVE MANUALLY SIZED
```

```
2881 031142  000137  010272                    JMP     START           ;JUMP TO START
2882 031146  022737  000105  002666  4$:       CMP     #'E,ANSWER      ;EDIT TABLE?
2883 031154  001414                            BEQ     EDIT            ;BRANCH TO EDIT IF SO
2884 031156  022737  000102  002666            CMP     #'B,ANSWER      ;ENTER BURST DATA LATE CALIBRATION?
2885 031164  001235                            BNE     PROMPT          ;BRANCH TO PROMPT IF COMMAND NOT RECOGNIZED
2886 031166  005737  002420                    TST     REGADR          ;SEE IF REGADR HAS BEEN LOADED
2887 031172  001750                            BEQ     2$              ;BRANCH TO ERROR MESSAGE IF NOT
2888 031174  005737  002460                    TST     VECADR          ;SEE IF VECADR HAS BEEN LOADED
2889 031200  001745                            BEQ     2$              ;BRANCH TO ERROR MESSAGE IF NOT
2890 031202  000137  032326                    JMP     BDLCR           ;JUMP TO BURST DATA LATE CALIBRATION ROUTINE
```

```
2891                                        .SBTTL  TABLE EDIT ROUTINE
2892 031206  104401  034147          EDIT:  TYPE    ,NOBUT          ;TYPE:  'NUMBER OF BOARDS UNDER TEST: '
2893 031212  013746  002416                 MOV     QTYBRD,-(SP)    ;PUT QUANTITY OF BOARDS ON STACK FOR PRINTING
2894 031216  104405                        TYPDS                   ;GO PRINT THE QUANTITY OF BOARDS
2895 031220  104401  034012                TYPE    ,SPACEC         ;TYPE:  '  : '
2896 031224  012703  000002                MOV     #2,R3           ;EXPECT MAX OF 2 CHARACTERS
2897 031230  112737  000071  002706        MOVB    #'9,LRGSTC      ;MOVE ASCII '9' TO THE LARGEST CHARACTER LOCATION
2898 031236  004737  002722                JSR     PC,READ         ;GO READ 2 CHARACTERS
2899 031242  022703  000002                CMP     #2,R3           ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
2900 031246  001017                        BNE     2$              ;BRANCH IF NOT
2901 031250  022737  000033  002666        CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
2902 031256  001453                        BEQ     5$              ;BRANCH TO PROMPT JUMP IF SO
2903 031260  022737  000003  002666        CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
2904 031266  001447                        BEQ     5$              ;BRANCH TO PROMPT JUMP IF EXIT REQUESTED
2905 031270  022737  000015  002666        CMP     #CARETN,ANSWER  ;SEE IF A <CR> WAS INPUTED
2906 031276  001412                        BEQ     4$              ;IF <CR> USE EXISTING NUMBER
2907 031300  104401  033023          1$:    TYPE    ,BDNERR         ;TYPE:  'ILLEGAL NUMBER (# OTHER THAN 1-16) OR CHARACTER INP
2908 031304  000740                        BR      EDIT            ;BRANCH BACK FOR NEW INPUT
2909 031306  005704                  2$:   TST     R4              ;CHECK FOR ZERO MODULES INPUT
2910 031310  001773                        BEQ     1$              ;BRANCH TO PRINT ERROR MESSAGE IF SO
2911 031312  022704  000020                CMP     #20,R4          ;SEE IF BOARD NUMBER IS ILLEGAL
2912 031316  100770                        BMI     1$              ;BRANCH TO PRINT ERROR MESSAGE IF SO
2913 031320  010437  002416          3$:   MOV     R4,QTYBRD       ;MOVE INPUTED NUMBER TO QTYBRD
2914 031324  104401  034720          4$:   TYPE    ,SDADRS         ;TYPE:  ' STARTING DEVICE ADDRESS: '
2915 031330  013746  002420                MOV     REGADR,-(SP)    ;MOVE THE PRESENT ADDRESS TO THE STACK FOR PRINTING
2916 031334  104402                        TYPOC                   ;PRINT THE ADDRESS
2917 031336  104401  034012                TYPE    ,SPACEC         ;TYPE:  '  : '
2918 031342  012703  000006                MOV     #6,R3           ;EXPECT MAXIMUM 6 CHARACTERS
2919 031346  112737  000067  002706        MOVB    #'7,LRGSTC      ;MOVE ASCII '7' TO THE LARGEST CHARACTER LOCATION
2920 031354  004737  002722                JSR     PC,READ         ;GO READ 6 CHARACTERS
2921 031360  022703  000006                CMP     #6,R3           ;SEE IF NON-NUMERIC CHARACTER INPUTED
2922 031364  001022                        BNE     8$              ;BRANCH IF NOT
2923 031366  022737  000033  002666        CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
2924 031374  001704                        BEQ     EDIT            ;BRANCH AROUND ESC PRINT AND PREVIOUS PROMPT BRANCH IF SO
2925 031376  022737  000003  002666        CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
2926 031404  001005                        BNE     7$              ;BRANCH AROUND PROMPT JUMP IF NOT
2927 031406  000137  030660          5$:   JMP     PROMPT          ;JUMP TO PROMPT A NEW COMMAND
2928 031412  104401  033146          6$:   TYPE    ,ADRERR         ;TYPE:  'ADDRESS INPUTED IS NOT IN THE RANGE 171000 TO 17700
2929 031416  000742                        BR      4$              ;BRANCH BACK FOR REINPUT
2930 031420  022737  000015  002666  7$:   CMP     #CARETN,ANSWER  ;SEE IF <CR> WAS ONLY CHARACTER INPUTED
2931 031426  001417                        BEQ     10$             ;IF <CR> USE EXISTING REG ADDRESS
2932 031430  000735                        BR      4$              ;BRANCH BACK - INPUT NOT LEGAL
2933 031432  022704  171000          8$:   CMP     #171000,R4      ;IS ANSWER BELOW 171000
2934 031436  101365                        BHI     6$              ;BRANCH TO PRINT ERROR MESSAGE IF IT IS
2935 031440  022704  177000                CMP     #177000,R4      ;IS ANSWER ABOVE 177000
2936 031444  103762                        BLO     6$              ;BRANCH TO PRINT ERROR MESSAGE IF NOT
2937 031446  032704  000007                BIT     #7,R4           ;TEST TO MAKE SURE A '0' IS PRESENT IN LOWEST OCTAL DIGIT
2938 031452  001403                        BEQ     9$              ;BRANCH AROUND ERROR MESSAGE TYPE IF SO
2939 031454  104401  033314                TYPE    ,ADLCHR         ;TYPE:  'ADDRESS INPUTED HAS OTHER THAN 0 FOR LEAST'
2940                                                                ;       'SIGNIFICANT OCTAL DIGIT'
2941 031460  000721                        BR      4$              ;BRANCH BACK FOR REINPUT
2942 031462  010437  002420          9$:   MOV     R4,REGADR       ;INSTALL NEW # IN TABLE
2943 031466  104401  034664          10$:  TYPE    ,SVADRS         ;TYPE:  'STARTING VECTOR ADDRESS:  '
2944 031472  013746  002460                MOV     VECADR,-(SP)    ;MOVE PRESENT VECTOR TO STACK FOR PRINTING
2945 031476  104403                        TYPOS                   ;PRINT THE PRESENT VECTOR ADDRESS
2946 031500     003     000                .BYTE   3,0             ;TYPE 3 CHARACTERS, SUPPRESS LEADING ZEROS
2947 031502  104401  034012                TYPE    ,SPACEC         ;TYPE:  '  : '
```

```
2948 031506  012703  000003                MOV     #3,R3           ;EXPECT ONLY 3 CHARACTERS
2949 031512  004737  002722                JSR     PC,READ         ;GO READ 3 CHARACTERS
2950 031516  022703  000003                CMP     #3,R3           ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
2951 031522  001015                       BNE     11$             ;BRANCH IF NOT
2952 031524  022737  000033  002666        CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
2953 031532  00  74                       BEQ     4$              ;BRANCH TO PREVIOUS PROMPT IF SO
2954 031534  022737  000003  002666        CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
2955 031542  001721                       BEQ     5$              ;BRANCH TO PROMPT JUMP IF SO
2956 031544  022737  000015  002666        CMP     #CARETN,ANSWER  ;SEE IF <CR> WAS INPUTED
2957 031552  001417                       BEQ     15$             ;BRANCH IF NO CHANGE WANTED
2958 031554  000744                       BR      10$             ;BRANCH BACK - INPUT WAS ILLEGAL
2959 031556  022704  000123          11$: CMP     #123,R4         ;SEE IF ANSWER IS BELOW 124
2960 031562  100403                       BMI     13$             ;BRANCH AROUND ERROR MESSAGE IF NOT
2961 031564  104401  033233                TYPE    ,VECERR         ;TYPE:  'VECTOR INPUTED IS NOT IN THE RANGE OF 124 TO 777'
2962 031570  000736                       BR      10$             ;BRANCH BACK FOR REINPUT
2963 031572  032704  000003          13$: BIT     #3,R4           ;MAKE SURE LEAST SIGNIFICANT OCTAL DIGIT IS '0' OR '4'
2964 031576  001403                       BEQ     14$             ;BRANCH OVER ERROR PRINTING IF NOT
2965 031600  104401  033417                TYPE    ,VCLCHR         ;TYPE:  'VECTOR INPUTED SHOULD HAVE ZERO AS LEAST DIGIT'
2966 031604  000730                       BR      10$             ;BRANCH BACK FOR REINPUT
2967 031606  010437  002460          14$: MOV     R4,VECADR       ;INSTALL NEW VECTOR ADDRESS IN TABLE
2968 031612  104401  034613          15$: TYPE    ,DR1WOB         ;TYPE:  'DR11-W OR B (W=0, B=1) CURRENT STATE = '
2969 031616  013737  001476  002720        MOV     $DDWO,DDW       ;MOVE DEVICE DESCRIPTOR WORD TO DDW
2970 031624  012737  000001  002712        MOV     #BIT0,BITTST    ;MOVE BIT STATE TO PRINT TO BITTST
2971 031632  004737  005606                JSR     PC,PSTATE       ;PRINT CURRENT W/B STATE
2972 031636  104401  034012                TYPE    ,SPACEC         ;TYPE:  '  :  '
2973 031642  012703  000001                MOV     #1,R3           ;ONLY INPUT 1 CHARACTER
2974 031646  004737  002722                JSR     PC,READ         ;GO READ 1 CHARACTER
2975 031652  005703                        TST     R3              ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
2976 031654  001415                       BEQ     16$             ;BRANCH AROUND NON-NUMERIC TESTS IF SO
2977 031656  022737  000033  002666        CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
2978 031664  001700                       BEQ     10$             ;BRANCH TO PREVIOUS PROMPT IF SO
2979 031666  022737  000015  002666        CMP     #CARETN,ANSWER  ;SEE IF USER WANTS NO CHANGE
2980 031674  001417                       BEQ     18$             ;BRANCH IF SO
2981 031676  022737  000003  002666        CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
2982 031704  001640                       BEQ     5$              ;BRANCH TO PROMPT JUMP IF SO
2983 031706  000741                       BR      15$             ;BRANCH BACK - INPUT NOT LEGAL
2984 031710  005704                  16$: TST     R4              ;CHECK FOR LEGAL INPUT
2985 031712  001403                       BEQ     17$             ;BRANCH IF OK
2986 031714  022704  000001                CMP     #1,R4           ;CHECK FOR ILLEGAL INPUT
2987 031720  001334                       BNE     15$             ;BRANCH BACK IF ILLEGAL STATE INPUTED
2988 031722  042737  000001  001476  17$: BIC     #BIT0,$DDWO     ;CLEAR THE BIT TO BE ALTERED
2989 031730  050437  001476                BIS     R4,$DDWO        ;PUT USER INPUT INTO $DDWO
2990 031734  104401  034551          18$: TYPE    ,DEVPRI         ;TYPE:  'DEVICE PRIORITY PRESENT LEVEL = '
2991 031740  013737  001476  002720        MOV     $DDWO,DDW       ;MOVE DEVICE DESCRIPTOR WORD TO DDW
2992 031746  004737  005630                JSR     PC,PNTPRI       ;PRINT DEVICE PRIORITY
2993 031752  104401  034012                TYPE    ,SPACEC         ;TYPE:  '  :  '
2994 031756  012703  000001                MOV     #1,R3           ;ONLY INPUT 1 CHARACTER
2995 031762  004737  002722                JSR     PC,READ         ;GO READ 1 CHARACTER
2996 031766  005703                        TST     R3              ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
2997 031770  001415                       BEQ     19$             ;BRANCH AROUND NON-NUMERIC TESTS IF NOT
2998 031772  022737  000033  002666        CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
2999 032000  001704                       BEQ     15$             ;BRANCH TO PREVIOUS PROMPT IF SO
3000 032002  022737  000003  002666        CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
3001 032010  001544                       BEQ     26$             ;BRANCH IF EXIT WANTED
3002 032012  022737  000015  002666        CMP     #CARETN,ANSWER  ;SEE IF <CR> INPUTED FOR NO CHANGE WANTED
3003 032020  001413                       BEQ     20$             ;BRANCH IF NO CHANGE WANTED
3004 032022  000744                       BR      18$             ;BRANCH BACK - INPUT NOT LEGAL
```

```
3005 032024  006304                    19$:   ASL     R4              ;PUT PRIORITY IN PROPER POSITION
3006 032026  006304                           ASL     R4              ;BY SHIFTING TO THE LEFT 5 PLACES
3007 032030  006304                           ASL     R4
3008 032032  006304                           ASL     R4
3009 032034  006304                           ASL     R4
3010 032036  042737  000340  001476          BIC     #LEVEL7,$DDW0   ;CLEAR OLD PRIORITY
3011 032044  050437  001476                   BIS     R4,$DDW0        ;SET PRIORITY INTO DEVICE DESCRIPTOR WORD
3012 032050  104401  034463          20$:     TYPE    ,TORNCB         ;TYPE:  '2 OR N CYCLE BURST (2 CY=0, N CY=1) PRESENT STATE =
3013 032054  013737  001476  002720           MOV     $DDW0,DDW       ;MOVE DEVICE DESCRIPTOR WORD TO DDW
3014 032062  012737  000002  002712           MOV     #BIT1,BITTST    ;MOVE BIT STATE TO PRINT TO BITTST
3015 032070  004737  005606                   JSR     PC,PSTATE       ;PRINT 2/N CYCLE STATE
3016 032074  104401  034012                   TYPE    ,SPACEC         ;TYPE:  '  : '
3017 032100  012703  000001                   MOV     #1,R3           ;ONLY ONE CHARACTER TO INPUT
3018 032104  004737  002722                   JSR     PC,READ         ;READ 1 CHARACTER
3019 032110  005703                           TST     R3              ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
3020 032112  001415                           BEQ     21$             ;BRANCH AROUND NON-NUMERIC TESTS IF NOT
3021 032114  022737  000033  002666           CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
3022 032122  001704                           BEQ     18$             ;BRANCH TO PREVIOUS PROMPT IF SO
3023 032124  022737  000003  002666           CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
3024 032132  001473                           BEQ     26$             ;BRANCH IF USER WANTS TO EXIT
3025 032134  022737  000015  002666           CMP     #CARETN,ANSWER  ;SEE IF USER WANTS NO CHANGE
3026 032142  001414                           BEQ     23$             ;BRANCH IF USER WANTS NO CHANGE
3027 032144  000741                           BR      20$             ;BRANCH BACK - USER INPUT NOT LEGAL
3028 032146  005704                    21$:   TST     R4              ;CHECK FOR LEGAL INPUT
3029 032150  001403                           BEQ     22$             ;BRANCH IF OK
3030 032152  022704  000001                   CMP     #1,R4           ;CHECK FOR ILLEGAL INPUT
3031 032156  001334                           BNE     20$             ;BRANCH BACK IF ILLEGAL STATE INPUTED
3032 032160  006304                    22$:   ASL     R4              ;SHIFT BIT OVER 1 PLACE
3033 032162  042737  000002  001476           BIC     #BIT1,$DDW0     ;CLEAR OLD STATE
3034 032170  050437  001476                   BIS     R4,$DDW0        ;SET THE USERS INPUTED STATE TO $DDW0
3035 032174  104401  034775          23$:     TYPE    ,DOCTS          ;TYPE:  'DO CABLE TESTS (NO=0, YES=1) PRESENT STATE = '
3036 032200  013737  001476  002720           MOV     $DDW0,DDW       ;MOVE DEVICE DESCRIPTOR WORD TO DDW
3037 032206  012737  000004  002712           MOV     #BIT2,BITTST    ;MOVE BIT STATE TO PRINT TO BITTST
3038 032214  004737  005606                   JSR     PC,PSTATE       ;PRINT CABLE STATE
3039 032220  104401  034012                   TYPE    ,SPACEC         ;TYPE:  '  : '
3040 032224  012703  000001                   MOV     #1,R3           ;INPUT ONLY 1 CHARACTER
3041 032230  004737  002722                   JSR     PC,READ         ;GO INPUT 1 CHARACTER
3042 032234  005703                           TST     R3              ;SEE IF NON-NUMERIC WAS THE ONLY INPUT
3043 032236  001415                           BEQ     24$             ;BRANCH AROUND NON-NUMERIC TESTS IF NOT
3044 032240  022737  000033  002666           CMP     #ESC,ANSWER     ;SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
3045 032246  001700                           BEQ     20$             ;BRANCH TO PREVIOUS PROMPT IF SO
3046 032250  022737  000003  002666           CMP     #CNTLC,ANSWER   ;SEE IF USER WANTS TO EXIT (^C)
3047 032256  001421                           BEQ     26$             ;BRANCH IF USER WANTS TO EXIT
3048 032260  022737  000015  002666           CMP     #CARETN,ANSWER  ;SEE IF USER WANTS NO CHANGE
3049 032266  001415                           BEQ     26$             ;BRANCH IF USER WANTS NO CHANGE
3050 032270  000741                           BR      23$             ;BRANCH BACK - USER INPUT NOT LEGAL
3051 032272  005704                    24$:   TST     R4              ;CHECK FOR LEGAL INPUT
3052 032274  001403                           BEQ     25$             ;BRANCH IF OK
3053 032276  022704  000001                   CMP     #1,R4           ;CHECK FOR ILLEGAL INPUT
3054 032302  001334                           BNE     23$             ;BRANCH BACK IF ILLEGAL STATE INPUTED
3055 032304  006304                    25$:   ASL     R4              ;SHIFT INPUTED BIT OVEP 2 PLACES
3056 032306  006304                           ASL     R4
3057 032310  042737  000004  001476           BIC     #BIT2,$DDW0     ;CLEAR BIT TO BE CHANGED
3058 032316  050437  001476                   BIS     R4,$DDW0        ;SET THE USERS INPUTED STATE TO $DDW0
3059 032322  000137  030660          26$:     JMP     PROMPT          ;JUMP TO GET NEW DEVICE NUMBER
```

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

032326  012737  177777  002672  BDLCR:  MOV     #-1,MANSIZE     ;MOVE -1 TO MANSIZE
032334  004737  003360                  JSR     PC,FIXTBL       ;GO FILL TABLE
032340  104401  033624                  TYPE    ,BDLCRM         ;TYPE:  'BURST DATA LATE CALIBRATION'
                                                                ;TYPE:  'ATTACH SCOPE PROBE...'
                                                                ;       'TO CALIBRATE NEXT BOARD, TYPE ANY CHARACTER'
032344  012737  000001  002546          MOV     #BIT0,DEVMSK    ;SET UP BIT MASK TO TEST $DEVM FOR DEVICES
032352  012700  002460                  MOV     #VECADR,R0      ;MOVE VECADR TO R0
032356  012701  002420                  MOV     #REGADR,R1      ;MOVE REGADR TO R1
032362  005037  001424                  CLR     $UNIT           ;CLEAR $UNIT
032366  033737  002546  001470  2$:     BIT     DEVMSK,$DEVM    ;CHECK TO SEE IF DEVICE IS TO BE TESTED
032374  001015                          BNE     5$              ;BRANCH IF SO
032376  005737  002546                  TST     DEVMSK          ;SEE IF BIT 15 IS SET
032402  100004                          BPL     4$              ;BRANCH TO CONTINUE IF NOT SET
032404  104401  033113                  TYPE    ,BCDONE         ;TYPE:  'BURST CALIBRATION COMPLETE'
032410  000137  030660                  JMP     PROMPT          ;JUMP TO PROMPT A NEW COMMAND
032414  022021                  4$:     CMP     (R0)+,(R1)+     ;INCREMENT THE TWO POINTERS
032416  006337  002546                  ASL     DEVMSK          ;UPDATE DEVICE MAP TEST MASK
032422  005237  001424                  INC     $UNIT           ;INCREMENT UNIT NUMBER
032426  000757                          BR      2$              ;GO TEST NEXT BIT OF DEVICE MASK
032430  011137  002524          5$:     MOV     (R1),CSR        ;PUT UUT CSR ADDRESS INTO DEVICE CSR LOCATION
032434  062737  000004  002524          ADD     #4,CSR          ;POINT CSR TO CSR ADDRESS
032442  011037  002530                  MOV     (R0),DRINV      ;PUT UUT VECTOR ADDRESS INTO DEVICE DRINV
032446  104401  033612                  TYPE    ,DEVICE         ;TYPE:  'DEVICE # '
032452  013746  001424                  MOV     $UNIT,-(SP)     ;PUT UNIT NUMBER ON STACK FOR TYPFOUT
032456  104405                          TYPDS                   ;GO TYPE THE UNIT NUMBER IN DECIMAL
032460  104401  034123                  TYPE    ,UCAL           ;TYPE:  ' UNDER CALIBRATION'
032464  004737  004036                  JSR     PC,CLENUP       ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
032470  005077  150030          6$:     CLR     @CSR            ;CLR CYCLE BIT
032474  012737  000077  002662          MOV     #77,TIME        ;MOVE WAIT LOOP COUNTER TO TIME
032502  052777  000400  150014          BIS     #CY,@CSR        ;SET CYCLE BIT
032510  005337  002662          7$:     DEC     TIME            ;SUBTRACT 1 FROM TIME UNTIL ZERO
032514  001375                          BNE     7$              ;BRANCH BACK IF NOT ZERO YET
032516  105777  146622                  TSTB    @$TKS           ;IS A CHARACTER WAITING INDICATING USER WANTS TO GO ON?
032522  100362                          BPL     6$              ;BRANCH IF NOT
032524  017737  150110  002662          MOV     @TKB,TIME       ;WASTE THE CHARACTER, CLEARING THE CHARACTER FLAG
032532  000730                          BR      4$              ;GO ON TO NEXT BOARD
```

```
                                                .SBTTL  ASCII AND ASCIZ MESSAGES AND LOCATIONS
032534      200     123     124  STKIFL: .ASCIZ  <CRLF>/STACK IS FULL - DATA MAY HAVE BEEN LOST/<CRLF><CRLF>
032607      136     131     200  ERCHDR: .ASCII  /^Y/<CRLF>/SUMMATION OF ERRORS SINCE START OR LAST REPORT/
032670      200     200     102          .ASCIZ  <CRLF><CRLF>/BOARD #   PASS #   ERRTTL/<CRLF>
032725      136     131     200  NODATA: .ASCIZ  /^Y/<CRLF>/NO ERROR TOTALS TO REPORT/<CRLF><CRLF>
032764      040     055     040  ETDEV:  .ASCIZ  / - TOTAL ERRORS THIS DEVICE = /
033023      111     114     114  BDNERR: .ASCIZ  /ILLEGAL NUMBER (# OTHER THAN 1-16) OR CHARACTER INPUTED/
033113      102     125     122  BCDONE: .ASCIZ  /BURST CALIBRATION COMPLETE/
033146      101     104     104  ADRERR: .ASCIZ  /ADDRESS INPUTED IS NOT IN THE RANGE 171000 TO 177000/
033233      126     105     103  VECERR: .ASCIZ  /VECTOR INPUTED IS NOT IN THE RANGE OF 124 TO 777/
033314      101     104     104  ADLCHR: .ASCIZ  /ADDRESS INPUTED HAS OTHER THAN 0 FOR LEAST SIGNIFICANT OCTAL DIGIT/
033417      126     105     103  VCLCHR: .ASCIZ  /VECTOR INPUTED SHOULD HAVE ZERO AS LEAST DIGIT/
033476      074     105     123  ESCAPE: .ASCIZ  /<ESC>/
033504      200     104     105  MUSTED: .ASCII  <CRLF>/DEVICE ADDRESS AND-OR VECTOR TABLE NOT SET UP - /
033565      115     125     123          .ASCIZ  /MUST EDIT FIRST/
033605      040     000          LETNCR: .ASCIZ  / /
033607      136     103     000  CNTRLC: .ASCIZ  /^C/
033612      104     105     126  DEVICE: .ASCIZ  /DEVICE # /
033624      200     102     125  BDLCRM: .ASCII  <CRLF>/BURST DATA LATE CALIBRATION/
033660      200     101     124          .ASCII  <CRLF>/ATTACH SCOPE PROBE.../
033706      200     124     117          .ASCIZ  <CRLF>/TO CALIBRATE NEXT BOARD, TYPE ANY CHARACTER/<CRLF>
033764      040     040     000  SPACES: .ASCIZ  /  /
033767      040     040     040  SPACE3: .ASCIZ  /   /
033773      040     040     040  SPACE6: .ASCIZ  /      /
034002      040     040     040  SPACE7: .ASCIZ  /       /
034012      040     040     072  SPACEC: .ASCIZ  /  : /
034017      102     000          B:      .ASCIZ  /B/
034021      127     000          W:      .ASCIZ  /W/
034023      116     117     000  NO:     .ASCIZ  /NO/
034026      131     105     123  YES:    .ASCIZ  /YES/
034032      062     000          TWO:    .ASCIZ  /2/
034034      116     000          N:      .ASCIZ  /N/
034036      102     117     101  BOARD:  .ASCIZ  /BOARD # /
034047      200     200     000  CRLF2:  .ASCIZ  <CRLF><CRLF>
034052      200     101     114  BCFIN:  .ASCIZ  <CRLF>/ALL BOARDS CALIBRATED - BEGINNING TEST/<CRLF>
034123      040     125     116  UCAL:   .ASCIZ  / UNDER CALIBRATION/<CRLF>
034147      200     116     125  NOBUT:  .ASCIZ  <CRLF>/NUMBER OF BOARDS UNDER TEST: /
034206      200     200     104  BRVWPC: .ASCII  <CRLF><CRLF>/DIAGNOSTIC HAS DETERMINED THE FOLLOWING ABOUT THE/<CRLF>
034272      104     122     061          .ASCII  /DR11-W(S) IT HAS FOUND.  USER *MUST* DETERMINE ACCURACY/<CRLF><CRLF>
034363      040     040     040          .ASCIZ  /   BOARD#   REGADR    VECADR     W-B    P-LEV   2-N CY   CABLE/<CRLF>
034463      200     062     040  TORNCB: .ASCIZ  <CRLF>/2 OR N CYCLE BURST (2 CY=0, N CY=1) PRESENT STATE = /
034551      200     104     105  DEVPRI: .ASCIZ  <CRLF>/DEVICE PRIORITY PRESENT LEVEL = /
034613      200     104     122  DR1WOB: .ASCIZ  <CRLF>/DR11-W OR B (W=0, B=1) CURRENT STATE = /
034664      200     123     124  SVADRS: .ASCIZ  <CRLF>/STARTING VECTOR ADDRESS:  /
034720      200     123     124  SDADRS: .ASCIZ  <CRLF>/STARTING DEVICE ADDRESS: /
034753      040     124     105  TSTCOM: .ASCIZ  / TESTING COMPLETE/
034775      200     104     117  DOCTS:  .ASCIZ  <CRLF>/DO CABLE TESTS (NO=0, YES=1) PRESENT STATE = /
035054      200     200     043  HEADER: .ASCII  <CRLF><CRLF>/# OF     START                                    2-N    CABLE/
035150      200     102     117          .ASCIZ  <CRLF>/BOARDS   REGADR   VECADR      W-B    P-LEV   CYCLE   TESTS/<CRLF>
035245      200     200     115  MBDIAL: .ASCIZ  <CRLF><CRLF>/MULTIPLE BOARD DIALOGUE/<CRLF>
035300      200     105     116  ECELR:  .ASCIZ  <CRLF>/ENTER COMMAND ([E]DIT, [L]IST, [B]URST CALIBRATION, [R]UN):  /
035377      124     110     105  NOMORE: .ASCII  /THERE ARE STILL MORE ERRORS, BUT WILL NOT BE PRINTED./<CRLF>
035465      105     122     122          .ASCIZ  /ERRORS WILL STILL BE COUNTED AND PRINTED AT THE EOP./<CRLF>
035553      200     116     117  NODVPR: .ASCII  <CRLF>/NO DEVICES RECOGNIZED - DIAGNOSTIC CANNOT BE RUN/<CRLF>
035635      122     105     123          .ASCIZ  /RESTART AT 204 IF A DEVICE IS PRESENT/<CRLF>
035704      200     050     136  M1:     .ASCII  <CRLF>/(^X) INHIBITS EOP'S, (^Y) FOR ERROR SUMMARY/<CRLF>
035761      125     116     111          .ASCIZ  /UNIBUS HANG?  RESTART AT ADDRESS /
```

```
3158 036023     200     200     103  M1A:    .ASCIZ  <CRLF><CRLF>/CZDRLB0 DR11 GEN NPR INTFC LOGIC TEST/<CRLF>
3159 036074     104     105     126  BARADR: .ASCIZ  /DEVICE ADDRESS - /
3160 036116     054     040     124  TSNUMB: .ASCIZ  /, TEST NUMBER - /
3161 036137     054     040     120  PASNUM: .ASCIZ  /, PASS NUMBER - /
3162                                  .EVEN
3163 036160  000000                  .SAV:   .WORD   0
3164 036162                          .BLKW   400
3165 037162  000000                  BUFF:   .WORD   0
3166 037164  037164                  XINBUF: .
3167 037166                          .BLKW   400
3168 040166  040166                  XCHKBU: .
3169 040170                          .BLKW   400
3170 041170  041172                  CAPNTR: .WORD   CAPSTK           ;LOCATION TO HOLD POINTER FOR CAPTURE STACK
3171 041172                          CAPSTK: .BLKW   600.             ;LOCATIONS TO STORE UP TO 150 DECIMAL PASSES AND THEIR ERROR
3172 043452  000000                  ENDSTK: .WORD   0                ;FLAG SIGNALING END OF THE STACK
```

```
                                                .SBTTL  ERROR MESSAGES
043454  124  105  123  EM1:    .ASCIZ  /TEST SEQUENCING ERROR/
043502  103  101  116  EM2:    .ASCIZ  /CANNOT ACCESS DR11 REGISTER/
043536  104  122  061  EM3:    .ASCIZ  /DR11-B OR W MODE INCORRECT (0=B, 1=W)/
043604  111  116  111  EM4:    .ASCIZ  /INIT FAILED TO CLEAR WCR/
043635  111  116  111  EM5:    .ASCIZ  /INIT FAILED TO CLEAR BAR/
043666  111  116  111  EM6:    .ASCIZ  /INIT FAILED TO CLEAR BDR/
043717  111  116  111  EM7:    .ASCIZ  /INIT FAILED TO CLEAR ALL CSR R-W BITS/
043765  122  105  123  EM10:   .ASCIZ  /RESET FAILED TO CLEAR WCR/
044017  101  124  124  EM11:   .ASCIZ  /ATTEMPT TO SET ALL WCR BITS FAILED/
044062  122  105  123  EM12:   .ASCIZ  /RESET FAILED TO CLEAR BAR/
044114  101  124  124  EM13:   .ASCIZ  /ATTEMPT TO SET ALL BAR BITS TO 1 FAILED/
044164  103  123  122  EM14:   .ASCIZ  /CSR BIT TEST FAILED (FATAL - DIAGNOSTIC NOT CONTINUED)/
044253  103  123  122  EM15:   .ASCIZ  /CSR BIT TEST FAILED/
044277  105  111  122  EM16:   .ASCIZ  /EIR BIT TEST FAILED/
044323  122  105  101  EM17:   .ASCIZ  /READY AND MAINTENANCE ARE NOT THE ONLY BITS SET IN CSR/
044412  101  124  124  EM20:   .ASCIZ  /ATTN AND ERROR FAILED TO SET PROPERLY/
044460  101  124  124  EM21:   .ASCIZ  /ATTN AND ERROR FAILED TO CLEAR PROPERLY/
044530  105  122  122  EM22:   .ASCIZ  /ERROR BIT SHOULD HAVE BEEN CLEAR/
044571  102  111  124  EM23:   .ASCIZ  /BIT PATTERN NOT LOADED PROPERLY IN WCR/
044640  122  105  101  EM24:   .ASCIZ  /READY OF CSR WAS NOT SET/
044671  102  111  124  EM25:   .ASCIZ  /BIT 0 OF THE BAR WAS SET/
044722  102  111  124  EM26:   .ASCIZ  /BIT PATTERN TEST FAILED IN BAR/
044761  127  103  122  EM27:   .ASCIZ  /WCR DATA PATTERN NOT CORRECT/
045016  106  125  116  EM30:   .ASCIZ  /FUNCTION BIT(S) ARE NOT CLEAR/
045054  104  123  124  EM31:   .ASCIZ  /DSTAT A, B OR C ARE NOT AS EXPECTED/
045120  102  104  122  EM32:   .ASCIZ  /BDR IS NOT CLEAR/
045141  101  114  114  EM33:   .ASCIZ  /ALL BDR BITS ARE NOT SET/
045172  102  104  122  EM34:   .ASCIZ  /BDR FAILS TO HOLD A BIT PATTERN/
045232  102  104  122  EM35:   .ASCIZ  /BDR SHOULD NOT HAVE BEEN LOADED WITH NEW PATTERN/
045313  102  104  122  EM36:   .ASCIZ  /BDR PATTERN NOT CORRECT/
045343  122  105  101  EM37:   .ASCIZ  /READY IS NOT THE ONLY BIT SET/
045401  122  105  101  EM40:   .ASCIZ  /READY SHOULD NOT BE SET/
045431  122  105  101  EM41:   .ASCIZ  /READY WAS CLEARED BUT NEVER SET AGAIN/
045477  122  105  101  EM42:   .ASCIZ  /READY CANNOT BE SET BY INIT/
045533  104  122  061  EM43:   .ASCIZ  /DR11 FAILED TO INTERRUPT/
045564  104  122  061  EM44:   .ASCIZ  /DR11 INTERRUPTED, BUT IT SHOULDN'T HAVE/
045634  105  122  122  EM45:   .ASCIZ  /ERROR BIT SHOULD NOT BE CLEAR/
045672  106  125  116  EM46:   .ASCIZ  /FUNCTION BITS DIDN'T INCREMENT IN MAINT MODE/
045747  103  123  122  EM47:   .ASCIZ  /CSR IS WRONG/
045764  124  122  101  EM50:   .ASCIZ  /TRANSFERS SHOULD HAVE BEEN INHIBITED/
046031  104  122  061  EM51:   .ASCIZ  /DR11 SHOULD NOT HAVE INTERRUPTED A SECOND TIME/
046110  105  130  120  EM52:   .ASCIZ  /EXPECTED INTERRUPT DID NOT OCCUR/
046151  127  103  122  EM53:   .ASCIZ  /WCR NOT EQUAL TO ZERO/
046177  102  101  122  EM54:   .ASCIZ  /BAR IS WRONG/
046214  102  101  104  EM55:   .ASCIZ  /BAD DATA IN BDR/
046234  104  101  124  EM56:   .ASCIZ  /DATA NOT TRANSFERED CORRECTLY/
046272  102  125  106  EM57:   .ASCIZ  /BUFFER DATA NOT CORRECT/
046322  124  117  117  EM60:   .ASCIZ  /TOO MANY WORDS WERE TRANSFERED/
046361  125  116  105  EM61:   .ASCIZ  /UNEXPECTED TRAP OR INTERRUPT TO TRAP ADDRESS BELOW/
046444  103  123  122  EM62:   .ASCIZ  /CSR AND-OR WCR AND-OR BAR ARE INCORECT/
046513  104  122  061  EM63:   .ASCIZ  /DR11 INTERRUPTED AT WRONG LEVEL/
046553  062  055  116  EM65:   .ASCIZ  /2-N CYCLE BURST SWITCH IN WRONG POSITION/
046624  115  125  114  EM66:   .ASCIZ  /MULTICYCLE BIT IN THE EIR IS WRONG/
046667  103  123  122  EM202:  .ASCIZ  /CSR PATTERN NOT CORRECT/
046717  102  104  122  EM211:  .ASCIZ  /BDR AND-OR WCR AND-OR BAR ARE INCORECT/
```

```
3229                                            .SBTTL  DATA HEADERS
3230 046766  105  130  120  DH1:    .ASCII  /EXPCTD  RECVD/<CRLF>
3231 047004  124  105  123          .ASCIZ  /TEST #  TEST #/
3232 047023  124  105  123  DH2:    .ASCIZ  /TEST #  ERR PC  ABRTPC  REGISTER/
3233 047064  124  105  123  DH3:    .ASCIZ  /TEST #  ERR PC  EXPMOD  ACTMOD  CSRADR/
3234 047133  124  105  123  DH4:    .ASCIZ  /TEST #  ERR PC  WCRADR  WCRCONTENTS/
3235 047177  124  105  123  DH5:    .ASCIZ  /TEST #  ERR PC  BARADR  BAREXP  BARRCV/
3236 047246  124  105  123  DH6:    .ASCIZ  /TEST #  ERR PC  BDRADR  BDRCONTENTS/
3237 047312  124  105  123  DH7:    .ASCIZ  /TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS/
3238 047366  040  040  040  DH14:   .ASCII  /                BIT(S)/<CRLF>
3239 047415  124  105  123          .ASCIZ  /TEST #  ERR PC  TESTED  CSRADR  CSREXP  CSRCONTENTS/
3240 047501  040  040  040  D416:   .ASCII  /                BIT(S)/<CRLF>
3241 047530  124  105  123          .ASCIZ  /TEST #  ERR PC  TESTED  EIRADR  EIREXP  EIRCONTENTS/
3242 047614  124  105  123  DH17:   .ASCIZ  /TEST #  ERR PC  CSRADR  CSREXP  CSRCONTENTS/
3243 047670  124  105  123  DH23:   .ASCIZ  /TEST #  ERR PC  WCRADR  WCREXP  WCRCONTENTS/
3244 047744  124  105  123  DH26:   .ASCIZ  /TEST #  ERR PC  BARADR  BAREXP  BARCONTENTS/
3245 050020  124  105  123  DH34:   .ASCIZ  /TEST #  ERR PC  BDRADR  BDREXP  BDRCONTENTS/
3246 050074  124  105  123  DH43:   .ASCIZ  /TEST #  ERR PC  CSRADR  CSRCONTENTS/
3247 050140  124  105  123  DH50:   .ASCIZ  /TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  BARADR  BAREXP  BARRCV/
3248 050237  124  105  123  DH56:   .ASCIZ  /TEST #  ERR PC  NPR1AD  NPR1EX  NPR1RC  CSRADR/
3249 050316  040  040  040  DH57:   .ASCII  /                CHECK   CHECK   INPUT   INPUT/<CRLF>
3250 050374  124  105  123          .ASCIZ  /TEST #  ERR PC  BUFADR  BUFDAT  BUFADR  BUFDAT  CSRADR/
3251 050463  040  040  040  DH60:   .ASCII  /                DIDNOT/<CRLF>
3252 050512  124  105  123          .ASCIZ  /TEST #  ERR PC  EXPECT  ADRESS  CSRADR/
3253 050561  124  105  123  DH61:   .ASCIZ  /TEST #  ERR PC  WCRADR  OLDPC   TRAP ADR/
3254 050632  124  105  123  DH62:   .ASCII  /TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  CSREXP  /
3255 050712  103  123  122          .ASCIZ  /CSRRCV  BAREXP  BARRCV/
3256 050741  124  105  123  DH63:   .ASCIZ  /TEST #  ERR PC  EXPLVL  RCVLVL  CSRADR/
3257 051010  124  105  123  DH64:   .ASCIZ  /TEST #  ERR PC  EXPLVL  CSRADR/
3258 051047  124  105  123  DH65:   .ASCIZ  /TEST #  ERR PC  CSRADR  EIREXP  EIRRCV/
3259 051116  124  105  123  DH66:   .ASCIZ  /TEST #  ERR PC  PATERN  CSRADR  CSRRCV/
3260 051165  124  105  123  DH202:  .ASCIZ  /TEST #  ERR PC  CSRADR  PATLDD  CSREXP  CSRRCV/
3261 051244  124  105  123  DH203:  .ASCIZ  /TEST #  ERR PC  CSRADR  PATERN  CSREXP  CSRCONTENTS/
3262 051330  124  105  123  DH207:  .ASCIZ  /TEST #  ERR PC  PATERN  CSRADR  CSRCONTENTS/
3263 051404  124  105  123  DH210:  .ASCIZ  /TEST #  ERR PC  WCRADR  WCRCONTENTS/
3264 051450  124  105  123  DH211:  .ASCIZ  /TEST #  ERR PC  WCRADR  WCREXP  WCRRCV  BDREXP  BDRRCV  BAREXP  BARRCV/
3265                                            .EVEN
```

```
                                                .SBTTL  DATA TABLES
051560  001362  001414  000000  DT1:    .WORD   $TMP1,$TESTN,0
051566  001414  001316  002674  DT2:    .WORD   $TESTN,$ERRPC,OLDPC1,DREG,0
051600  001414  001316  001362  DT3:    .WORD   $TESTN,$ERRPC,$TMP1,BORW,CSR,0
051614  001414  001316  002520  DT4:    .WORD   $TESTN,$ERRPC,WCR,RWCR,0
051626  001414  001316  002522  DT5:    .WORD   $TESTN,$ERRPC,BAR,EBAR,RBAR,0
051642  001414  001316  002526  DT6:    .WORD   $TESTN,$ERRPC,BDR,RBDR,0
051654  001414  001316  002524  DT7:    .WORD   $TESTN,$ERRPC,CSR,ECSR,RCSR,0
051670  001414  001316  002540  DT14:   .WORD   $TESTN,$ERRPC,BUT,CSR,ECSR,RCSR,0
051706  001414  001316  002540  DT16:   .WORD   $TESTN,$ERRPC,BUT,CSR,EEIR,REIR,0
051724  001414  001316  002524  DT17:   .WORD   $TESTN,$ERRPC,CSR,ECSR,RCSR,0
051740  001414  001316  002520  DT23:   .WORD   $TESTN,$ERRPC,WCR,EWCR,RWCR,0
051754  001414  001316  002522  DT26:   .WORD   $TESTN,$ERRPC,BAR,EBAR,RBAR,
051770  001414  001316  002526  DT34:   .WORD   $TESTN,$ERRPC,BDR,EBDR,RBDR,0
052004  001414  001316  002524  DT43:   .WORD   $TESTN,$ERRPC,CSR,RCSR,0
052016  001414  001316  002520  DT50:   .WORD   $TESTN,$ERRPC,WCR,EWCR,RWCR,BAR,EBAR,RBAR,0
052040  001414  001316  002610  DT56:   .WORD   $TESTN,$ERRPC,ANPR1,ENPR1,NPR1,CSR,0
052056  001414  001316  001370  DT57:   .WORD   $TESTN,$ERRPC,$TMP4,$TMP2,$TMP5,$TMP3,CSR,0
052076  001414  001316  001364  DT60:   .WORD   $TESTN,$ERRPC,$TMP2,$TMP3,CSR,0
052112  001414  001316  002520  DT61:   .WORD   $TESTN,$ERRPC,WCR,OLDPC2,BDVECT,0
052126  001414  001316  002520  DT62:   .WORD   $TESTN,$ERRPC,WCR,EWCR,RWCR,ECSR,RCSR,EBAR,RBAR,0
052152  001414  001316  002554  DT63:   .WORD   $TESTN,$ERRPC,DRLEV,LEVEL,CSR,0
052166  001414  001316  001362  DT64:   .WORD   $TESTN,$ERRPC,$TMP1,CSR,0
052200  001414  001316  002524  DT65:   .WORD   $TESTN,$ERRPC,CSR,EEIR,REIR,0
052214  001414  001316  002606  DT66:   .WORD   $TESTN,$ERRPC,ENPR1,CSR,RCSR,0
052230  001414  001316  002524  DT202:  .WORD   $TESTN,$ERRPC,CSR,BUT,ECSR,RCSR,0
052246  001414  001316  002524  DT203:  .WORD   $TESTN,$ERRPC,CSR,$TMP0,ECSR,RCSR,0
052264  001414  001316  001362  DT207:  .WORD   $TESTN,$ERRPC,$TMP1,CSR,RCSR,0
052300  001414  001316  002520  DT210:  .WORD   $TESTN,$ERRPC,WCR,RWCR,0
052312  001414  001316  002520  DT211:  .WORD   $TESTN,$ERRPC,WCR,EWCR,RWCR,EBDR,RBDR,EBAR,RBAR,0
```

```
3296                                            .SBTTL  BUS HANG ROUTINE
3297 052336  104401  036074          UBHANG:   TYPE    ,BARADR           ;TYPE:  'DEVICE ADDRESS - '
3298 052342  013746  002520                    MOV     WCR,-(SP)         ;PUT DEVICE ADDRESS ON STACK
3299 052346  104402                            TYPOC                     ;GO TYPE IT IN OCTAL
3300 052350  104401  036116                    TYPE    ,TSNUMB           ;TYPE:  ', TEST NUMBER - '
3301 052354  013746  001414                    MOV     $TESTN,-(SP)      ;PUT TEST NUMBER ON STACK
3302 052360  104403                            TYPOS                     ;GO TYPE IT IN OCTAL
3303 052362     002     000                    .BYTE   2,0               ;TYPE 2 DIGITS, LEADING ZEROS SUPRESSED
3304 052364  104401  036137                    TYPE    ,PASNUM           ;TYPE:  ', PASS NUMBER - '
3305 052370  013746  001420                    MOV     $PASS+2,-(SP)     ;MOVE OVERFLOW NUMBER TO THE STACK
3306 052374  013746  001416                    MOV     $PASS,-(SP)       ;PUT PASS NUMBER ON STACK
3307 052400  104406                            TYPDE                     ;GO TYPE IT IN EXTENDED DECIMAL
3308 052402  104401  001405                    TYPE    ,$CRLF            ;TYPE A <CRLF>
3309 052406  000000                            HALT                      ;WHOA - YOU GOTTA SERIOUSA PROBLEMA, BUDDY!
3310 052410  000137  010266                    JMP     START1            ;JUMP TO RESTART PROGRAM
3311 052414  000000                  NOCARE:   .WORD   0         ;LOCATION FOR USE WHENEVER CYCLE BIT OF CSR IS USED. THIS
3312                                                              ;SHOULD *ALWAYS* BE THE LAST WORD LOCATION IN THIS DIAGNOSTIC
3313          000001                            .END
```

SYMBOL TABLE

```
ABASE = 172410        AVECT1= 000300        CARETN= 000015        DATCHK  003716        DT1     051560
ACDW1 = 000000        AVECT2= 000000        CAT   = 157777        DATCH1  003736        DT14    051670
ACDW2 = 000000        B       034017        CATCH   005536        DATCH2  004014        DT16    051706
ACPUOP= 000000        BAR     002522        CBIT0 = 177776        DATOCK  004106        DT17    051724
ADDR    002650        BARADR  036074        CBIT1 = 177775        DATOC1  004126        DT2     051566
ADDW0 = 000000        BCDONE  033113        CBIT10= 175777        DATOC2  004170        DT202   052230
ADDW1 = 000000        BCFIN   034052        CBIT11= 173777        DBC   = 003000        DT203   052246
ADDW10= 000000        BDFAIL  002670        CBIT12= 167777        DDISP = 177570        DT207   052264
ADDW11= 000000        BDLCR   032326        CBIT13= 157777        DDW     002720        DT210   052300
ADDW12= 000000        BDLCRM  033624        CBIT14= 137777        DEVADS  004344        DT211   052312
ADDW13= 000000        BDNERR  033023        CBIT15= 077777        DEVICE  033612        DT23    051740
ADDW14= 000000        BDR     002526        CBIT2 = 177773        DEVMSK  002546        DT26    051754
ADDW15= 000000        BDVECT  002544        CBIT3 = 177767        DEVPRI  034551        DT3     051600
ADDW2 = 000000        BEGIN   010466        CBIT4 = 177757        DH1     046766        DT34    051770
ADDW3 = 000000        BEGIN1  011006        CBIT5 = 177737        DH14    047366        DT4     051614
ADDW4 = 000000        BITTST  002712        CBIT6 = 177677        DH16    047501        DT43    052004
ADDW5 = 000000        BIT0  = 000001        CBIT7 = 177577        DH17    047614        DT5     051626
ADDW6 = 000000        BIT00 = 000001        CBIT8 = 177377        DH2     047023        DT50    052016
ADDW7 = 000000        BIT01 = 000002        CBIT9 = 176777        DH202   051165        DT56    052040
ADDW8 = 000000        BIT02 = 000004        CB1513= 057777        DH203   051244        DT57    052056
ADDW9 = 000000        BIT03 = 000010        CCY   = 177377        DH207   051330        DT6     051642
ADEVCT= 000000        BIT04 = 000020        CDAB  = 171777        DH210   051404        DT60    052076
ADEVM = 000000        BIT05 = 000040        CDAC  = 172777        DH211   051450        DT61    052112
ADLCHR  033314        BIT06 = 000100        CDBC  = 174777        DH23    047670        DT62    052126
ADRERR  033146        BIT07 = 000200        CDSA  = 173777        DH26    047744        DT63    052152
AENV  = 000000        BIT08 = 000400        CDSB  = 175777        DH3     047064        DT64    052166
AENVM = 000000        BIT09 = 001000        CDSC  = 176777        DH34    050020        DT65    052200
AFATAL= 000000        BIT1  = 000002        CDST  = 170777        DH4     047133        DT66    052214
AMADR1= 000000        BIT10 = 002000        CEIR  = 077777        DH43    050074        DT7     051654
AMADR2= 000000        BIT11 = 004000        CER   = 077777        DH5     047177        EBAR    002602
AMADR3= 000000        BIT12 = 010000        CFNC  = 177761        DH50    050140        EBDR    002600
AMADR4= 000000        BIT13 = 020000        CF1   = 177775        DH56    050237        ECELR   035300
AMAMS1= 000000        BIT14 = 040000        CF2   = 177773        DH57    050316        ECSR    002574
AMAMS2= 000000        BIT15 = 100000        CF3   = 177767        DH6     047246        EDIT    031206
AMAMS3= 000000        BIT2  = 000004        CGO   = 177776        DH60    050463        EEIR    002576
AMAMS4= 000000        BIT3  = 000010        CHARCT  027423        DH61    050561        EIR   = 100000
AMSGAD= 000000        BIT4  = 000020        CHKBFF  003520        DH62    050632        EMTVEC= 000030
AMSGLG= 000000        BIT5  = 000040        CHKBUF  002622        DH63    050741        EM1     043454
AMSGTY= 000000        BIT6  = 000100        CHKCAB  004060        DH64    051010        EM10    043765
AMTYP1= 000000        BIT7  = 000200        CHK4DR  005104        DH65    051047        EM11    044017
AMTYP2= 000000        BIT8  = 000400        CIE   = 177677        DH66    051116        EM12    044062
AMTYP3= 000000        BIT9  = 001000        CKSWR = 104410        DH7     047312        EM13    044114
AMTYP4= 000000        BOARD   034036        CLENUP  004036        DIOMEM  002616        EM14    044164
ANPR1   002610        BORW    002612        CMA   = 167777        DISPLA  001342        EM15    044253
ANSWER  002666        BPINIT  004374        CNTLC = 000003        DISPRE  000174        EM16    044277
APASS = 000000        BPT   = 000003        CNTRLC  033607        DOCTS   034775        EM17    044323
APRIOR= 000000        BPTINT  004436        CNX   = 137777        DREG    002552        EM2     043502
APTCSU= 000040        BPTVCT  000014        CR    = 000015        DRGET   004452        EM20    044412
APTENV= 000001        BPTVEC= 000014        CRLF  = 000200        DRINV   002530        EM202   046667
APTSIZ= 000200        BRVWPC  034206        CRLF2   034047        DRLEV   002554        EM21    044460
APTSPO= 000100        BRWAIT  002630        CRY   = 177577        DRVS    002532        EM211   046717
ASIZE   005274        BUFF    037162        CSR     002524        DR1WOB  034613        EM22    044530
ASWREG= 000000        BUFLEN  002624        CX6   = 177757        DSA   = 004000        EM23    044571
AT    = 020000        BUSERR= 000004        CX7   = 177737        DSB   = 002000        EM24    044640
ATESTN= 000000        BUT     002540        CY    = 000400        DSC   = 001000        EM25    044671
AUNIT = 000000        CAPNTR  041170        DAB   = 006000        DST   = 007000        EM26    044722
AUSWR = 000000        CAPSTK  041172        DAC   = 005000        DSWR  = 177570        EM27    044761
```

```
EM3     043536          F1     = 000002          NXTTST  002556          STACK = 001300          TST12   013750
EM30    045016          F2     = 000004          N2     = 000400          STAGIN  000210          TST13   014106
EM31    045054          F3     = 000010          OFL     002704          START   010272          TST14   014244
EM32    045120          GO     = 000001          OLDPC1  002674          START1  010266          TST15   014406
EM33    045141          GOAGIN  023516           OLDPC2  002700          STKIFL  032534          TST16   014612
EM34    045172          GTSWR = 104407           OLDPS1  002676          STKLMT= 177774          TST17   015022
EM35    045232          HAKTPM  027042           OLDPS2  002702          SVADRS  034664          TST2    014414
EM36    045313          HEADER  035054           PASCNT  002560          SWR     001340          TST20   015372
EM37    045343          HT     = 000011          PASNUM  036137          SWREG   000176          TST21   015750
EM4     043604          IE     = 000100          PATRNS  006250          SW0    = 000001         TST22   016236
EM40    045401          INBUF   002620           PIRQ   = 177772          SW00   = 000001         TST23   016506
EM41    045431          INBUF1  002660           PIRQVE= 000240           SW01   = 000002         TST24   017216
EM42    045477          INOOUT  020340           PNTPRI  005630          SW02   = 000004         TST25   017472
EM43    045533          INTA    003546           PROMPT  030660          SW03   = 000010         TST26   017734
EM44    045564          IOTVEC= 000020           PR0    = 000000         SW04   = 000020         TST27   020164
EM45    045634          KDPAR2= 172364           PR1    = 000040         SW05   = 000040         TST3    011516
EM46    045672          KDPDR2= 172324           PR2    = 000100         SW06   = 000100         TST30   020354
EM47    045747          KIPAR2= 172344           PR3    = 000140         SW07   = 000200         TST31   020614
EM5     043635          KIPDR2= 172304           PR4    = 000200         SW08   = 000400         TST32   020716
EM50    045764          LENCHK  002626           PR5    = 000240         SW09   = 001000         TST33   021250
EM51    046031          LETNCR  033605           PR6    = 000300         SW1    = 000002         TST34   021526
EM52    046110          LEVEL   002542           PR7    = 000340         SW10   = 002000         TST35   022062
EM53    046151          LEVELS  005264           PS     = 177776         SW11   = 004000         TST36   022332
EM54    046177          LEVEL3= 000140           PSTATE  005606          SW12   = 010000         TST37   022620
EM55    046214          LEVEL4= 000200           PSW    = 177776         SW13   = 020000         TST4    012030
EM56    046234          LEVEL5= 000240           PTCAPT  003170          SW14   = 040000         TST40  = 023074
EM57    046272          LEVEL6= 000300           PWRVEC= 000024           SW15   = 100000         TST5    012572
EM6     043666          LEVEL7= 000340           QTYBRD  002416          SW2    = 000004         TST6    012740
EM60    046322          LF     = 000012          RA     = 002420         SW3    = 000010         TST7    013254
EM61    046361          LODBUF  003472           RBAR    002570          SW4    = 000020         TWO     034032
EM62    046444          LOOP    002664           RBDR    002566          SW5    = 000040         TYPCNF  004630
EM63    046513          LRGSTC  002706           RCSR    002562          SW6    = 000100         TYPDE  = 104406
EM65    046553          MA     = 010000          RDCHR = 104411          SW7    = 000200         TYPDS  = 104405
EM66    046624          MAICLR  006266           RDDEC = 104414          SW8    = 000400         TYPE   = 104401
EM7     043717          MAISET  007266           RDLIN = 104412          SW9    = 001000         TYPOC  = 104402
ENDEV   023074          MANSIZ  002672           RDOCT = 104413          TABINX  002550          TYPON  = 104404
ENDSTK  043452          MBD     030644           RDYCHK  002634          TBITVE= 000014          TYPOS  = 104403
ENPR1   002606          MBDIAL  035245           READ    002722          TESLR   027424          UBHANG  052336
EOPLOC  002710          MEMGMT  002714           REGADR  002420          TIME    002662          UCAL    034123
EOPRSM  027352          MEPITM  030240           REINIT  011206          TKB     002640          VA     = 002460
ER     = 100000         MESSAG  002652           REIR    002564          TKS     002636          VCLCHR  033417
ERCAPT  003126          MMPS   = 000252          RESVEC= 000010           TKVEC  = 000060         VCTADS  005502
ERCHDR  032607          MMR0   = 177572          RSTRT   023536          TMOPSW= 000006          VECADR  002460
ERRCHK  004254          MMVECT= 000250           RWCR    002572          TORNCB  034463          VECERR  033233
ERRCNT  002716          MSG     002646           RY     = 000200         TOVECT= 000004          W       034021
ERROR  = 104000         MUSTED  033504           R6     =%000006         TPB     002644          WCLEN   002632
ERRVEC= 000004          M1      035704           R7     =%000007         TPS     002642          WCR     002520
ER200   002276          M1A     036023           SCOPE  = 000004         TPVEC  = 000064         XCHKBU  040166
ESC    = 000033         N       034034           SDADRS  034720          TRAPVE= 000034          XINBUF  037164
ESCAPE  033476          NO      034023           SDRINV  002534          TRTVEC= 000014          X6     = 000020
ETDEV   032764          NOBUT   034147           SDRVS   002536          TSNUMB  036116          X7     = 000040
EWCR    002604          NOCARE  052414           SETTUP  005660          TSTCOM  034753          YES     034026
EXPAND  024670          NODATA  032725           SPACEC  034012          TSTDEV  011020          $APTHD  001000
FIXTBL  003360          NODVPR  035553           SPACES  033764          TSTMM   006134          $ATYC   030266
FLAG    002654          NOMORE  035377           SPACE3  033767          TST1    011244          $ATY1   030242
FNC    = 000016         NPR1    002614           SPACE6  033773          TST10   013404          $ATY3   030250
FNCCNT  002656          NX     = 040000          SPACE7  034002          TST11   013612          $ATY4   030260
```

SYMBOL TABLE

```
$AUTOB  001334        $DEVCT  001422        $HIOCT  026262        $OMODE  024340        $TMP6   001374
$BASE   001466        $DEVM   001470        $ICNT   001304        $OVER   027026        $TN   = 000040
$BDADR  001322        $DOAGN  023474        $INTAG  001335        $PASS   001416        $TPB    001352
$BDDAT  001326        $DTBL   024646        $ITEMB  001314        $PASTM  001006        $TPFLG  001357
$BELL   001400        $ENDAD  023464        $LF     001406        $POWER  030576        $TPS    001350
$CDW1   001472        $ENDCT  023302        $LFLG   030505        $PWRDN  030510        $TRAP   026264
$CDW2   001474        $ENDMG  023503        $LPADR  001306        $PWRUP  030526        $TRAP2  026306
$CHARC  024110        $ENULL  023500        $LPERR  001310        $QUES   001404        $TRP  = 000015
$CKSWR  025136        $ENV    001432        $MADR1  001444        $RDCHR  025454        $TRPAD  026320
$CMTAG  001300        $ENVM   001433        $MADR2  001450        $RDDEC  025746        $TSTM   001004
$CM3  = 000000        $EOP    023246        $MADR3  001454        $RDLIN  025574        $TSTNM  001302
$CM4  = 000007        $EOPCT  023274        $MADR4  001460        $RDOCT  026124        $TTYIN  025702
$CNTLG  025716        $ERFLG  001303        $MAIL   001410        $RDSZ = 000007        $TYPD   024356
$CNTLU  025711        $ERMAX  001315        $MAMS1  001442        $RTNAD  023476        $TYPDE  024342
$CPUOP  001440        $ERROR  027564        $MAMS2  001446        $SCOPE  026352        $TYPDS  024352
$CRLF   001405        $ERRPC  001316        $MAMS3  001452        $SETUP= 000137        $TYPE   023546
$DBLK   024656        $ERRTB  001536        $MAMS4  001456        $STUP = 177777        $TYPEC  023760
$DDW0   001476        $ERRTY  027764        $MBADR  001002        $SVLAD  026772        $TYPEX  024112
$DDW1   001500        $ERTTL  001312        $MFLG   030504        $SVPC = 001000        $TYPOC  024140
$DDW10  001522        $ESCAP  001376        $MNEW   025734        $SWR  = 163400        $TYPON  024154
$DDW11  001524        $ETABL  001432        $MSGAD  001426        $SWREG  001434        $TYPOS  024114
$DDW12  001526        $ETEND  001536        $MSGLG  001430        $SWRMK= 000200        $UNIT   001424
$DDW13  001530        $FATAL  001412        $MSGTY  001410        $SW08T  027466        $UNITM  001010
$DDW14  001532        $FFLG   030506        $MSWR   025723        $TESTN  001414        $USWR   001436
$DDW15  001534        $FILLC  001356        $MTYP1  001443        $TKB    001346        $VECT1  001462
$DDW2   001502        $FILLS  001355        $MTYP2  001447        $TKS    001344        $VECT2  001464
$DDW3   001504        $GDADR  001320        $MTYP3  001453        $TMP0   001360        $XTSTR  026634
$DDW4   001506        $GDDAT  001324        $MTYP4  001457        $TMP1   001362        $$GET4= 000000
$DDW5   001510        $GET42  023454        $NULL   001354        $TMP2   001364        $$SW08= 000040
$DDW6   001512        $GTSWR  025236        $NUMS   025124        $TMP3   001366        $OFILL  024337
$DDW7   001514        $HD   = 000000        $NWTST= 000001        $TMP4   001370        .SAV    036160
$DDW8   001516        $HIBTS  001000        $OCNT   024336        $TMP5   001372        .$X   = 001000
$DDW9   001520

. ABS.  052416      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  55704 WORDS  ( 218 PAGES)
DYNAMIC MEMORY:  20346 WORDS  ( 78 PAGES)
ELAPSED TIME:  00:11:55
CZDRLB.BIN,CZDRLB.SEQ/-SP=CZDRLB.MLB/ML,CZDRLB.P11
```