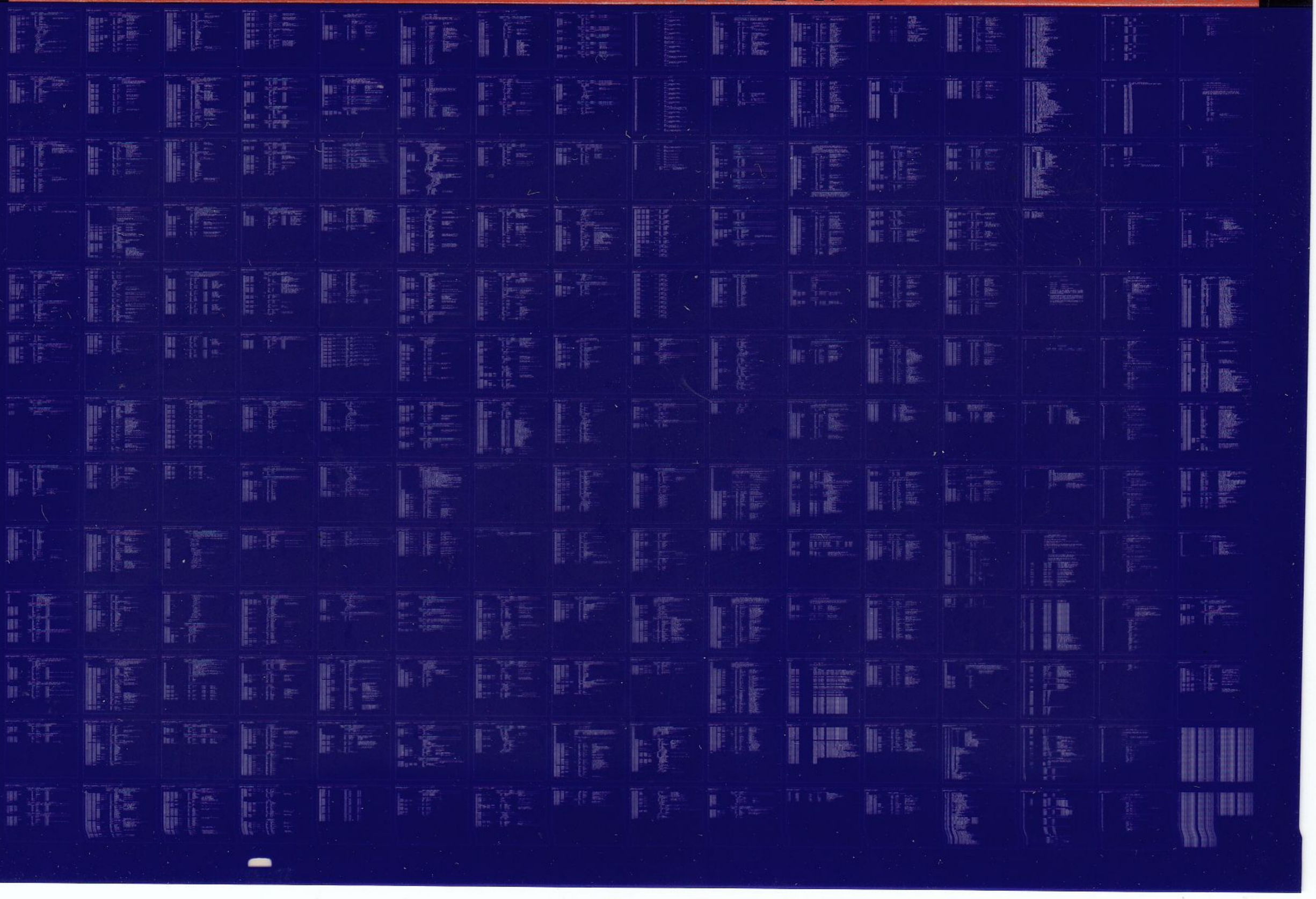
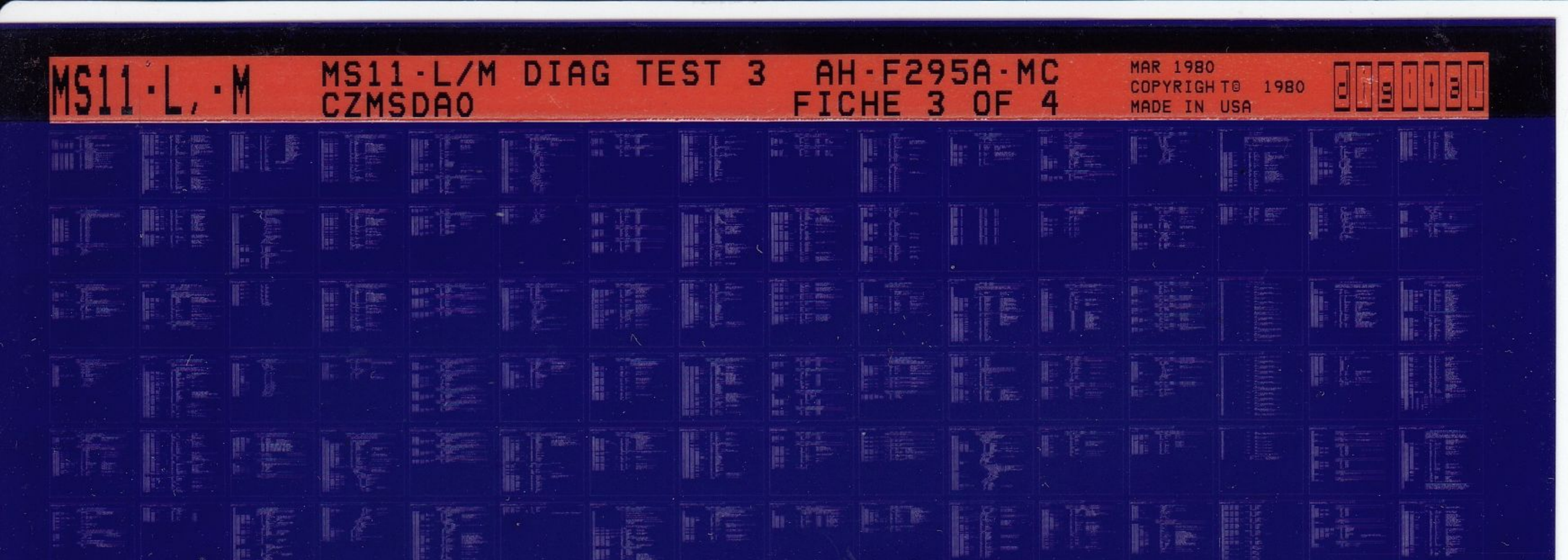
MS11-L,-M MS11-L/M DIAG TEST 3 AH-F295A-MC
CZMSDAO FICHE 1 OF 4

DEC-11-HMSDA-A-D

MS11-L,-M MS11-L/M DIAG TEST 3 AH-F295A-MC
CZMSDA0 FICHE 2 OF 4



MS11-L,-M MS11-L/M DIAG TEST 3 AH-F295A-MC
CZMSDA0 FICHE 3 OF 4



MS11-L,-M MS11-L/M DIAG TEST 3 AH-F295A-MC
CZMSDA0 FICHE 4 OF 4

d13013

PRODUCT CODE: AC-F294A-MC

PRODUCT NAME: CZMSDA0 MS11-L/M MEMORY DIAGNOSTIC

DATE CREATED: DECEMBER 1979

MAINTAINER: BASIC SYSTEM DIAGNOSTIC ENGINEERING

AUTHOR: MICHAEL D. BIBEAULT

TABLE OF CONTENTS

1.0 GENERAL PROGRAM INFORMATION

1.1 Program Purpose (Abstract)

a. Intended for use on all PDP-11's which meet the conditions in 1.2.1.

b. This program will be used by system managers and operators to determine the correct operation of main memory and also it will be primarily used by field service and manufacturing to isolate failures to the memory and to isolate failures within the memory to the correct card.

c. The object of this software is to functionally test and verify all main memory functions as fast as possible.

d. There is the capability of testing mixed configurations (MS11-L, MS11-M and what ever else in on the system).

e. It has special a maintenance mode (Field Service Mode) to provide specific functional capabilities.

1.2 System Requirements

1.2.1 Hardware Requirements -

PDP-11 CPU and at least 64K (16 Bit Words) of Memory and Memory Management.

NOTE

Like memory types must be on 16K word boundaries starting at physical address 0.

The CSR with the lowest address that is to be used must control the block of memory in which the program resides (PA 0-77776) in order for the diagnostic to properly size.

1.2.2 Software Requirements -

This program is designed to run stand alone or under any of the following monitors:

XXDP
ACT
APT

1.3 Related Documents and Standards

1. PDP-11/04/34/45/55/60 Processor Handbook (EB9340)
2. PDP-11/44 User's Guide (EK-11044-UG)
3. MS11-M User's Guide (EK-MS11M-UG-001)
4. Programming Practices (175-003-009-02)
5. System Macro Manual (MAINDEC-11-DXQAC-C-D)
6. SUPER-MAC Reference Guide (130-380-007-00)
7. Standard APT System to PDP-11 Diagnostic Interface (APT/11-317-07-09)
8. ACT11/XXDP Programming Specification (AUTOCAT-11-QZAUB-B-D)

1.4 Diagnostic Hierarchy Prerequisites

If the program in any way misbehaves, then:

1. Try it again with Cache off (reference Section 2.3.3.1)
2. Inhibit relocation (reference section 2.3.1)
3. Try CPU Diagnostics
4. Try Memory Management Diagnostics
5. Try Cache Diagnostics (where applicable)
6. Try UNIBUS Map Diagnostics (where applicable)

1.5 Assumptions

This program assumes the correct operation of the CPU, Memory Management, Cache, and the UNIBUS Map. This program occupies (initially) Bank 0 (0-16K). The XXDP loaders are in bank 1.

2.0 OPERATING INSTRUCTIONS

2.1 Loading & Starting Procedures

2.1.1 Quick Starting -

1. Load address 200
2. Set switch register for options (normally 0)
3. Start

2.1.2 Stopping -

1. Set SW8, and/or
2. Type control "C" (Reference section 2.3.4.1).

2.1.3 Restarting (Preserve Configuration Table) -

1. Load address 202
2. Set switch register for options (Normally 0)
3. Start

2.1.4 Switch Register Options -

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	INHIBIT RELOCATION
11	QUICK VERIFY
10	BELL ON ERROR
9	LOOP ON ERROR
8	HALT PROGRAM (UNRELOCATE & RESTORE LOADERS)
7	DETAILED ERROR REPORTS
6	INHIBIT CONFIGURATION MAP
5	LIMIT MAX ERRORS PER BANK
4	FAT TERMINAL (132 COLUMNS OR BETTER)
3	TEST MODE - SEE DOCUMENT
2	TEST MODE - SEE DOCUMENT
1	TEST MODE - SEE DOCUMENT
0	DETECT SINGLE BIT ERRORS

2.2 Default Test Sequence

The following two lists give the test protocol for parity and ECC Memory. Tests marked with a "*" are not normally run except under ACT or APT, or through a Field Service Command (Reference Section 2.3.4.8).

2.2.1 Test Protocol for Parity Memory -

Pattern	Pattern Name	Time (sec/16K)
34	Soft Error Test	<1
6	Initial Data Test	<1
17	Holding 1's and 0's Test	<1
7	Address Bit Test	<1
1	Address Test	<1
2	Complement Address Test	<1
3	3 XOR 9 Test	1
4	Rotating 0's Test	1
5	Rotating 1's Test	1
21	Marching 1's and 0's Test	1
35	Worst Case Noise Parity Test	n/a
* 22	Refresh Test	10
* 23	Shifting Diagonal Test	10
26	Random Data Test	<1
* 24	Fast Galloping Pattern Test	20
* 31	Sob-a-long Test	3
* 32	Write Recovery Test	<1
* 33	Branch Gobble Test	35
34	Soft Error Test	<1

2.2.2 Test Protocol for ECC Memory -

	Pattern	Pattern Name	Time(sec/16K)
@	25	Interrupt Enable Test	<1
+@	11	Single Bit Error Test	<2
+@	12	Write Byte Clears SBE Test	<1
+@	13	Create Double Bit Error Test	1
%+@	14	Write Inhibit DATIP w/DBE Test	1
+@	15	Write Inhibit of Byte w/DBE Test	1
+@	16	Write Inhibit of Word w/DBE Test	<1
	34	Soft Error Test	<1
	6	Initial Data Test	<1
	10	Byte Address Test	<1
	17	Holding 1's and 0's Test	<1
	7	Address Bit Test	<1
	1	Address Test <1	
	2	Complement Address Test	<1
	4	Rotating 0's Test	1
	5	Rotating 1's Test	1
	21	Marching 0's and 1's Test	1
@	20	Marchin 0's and 1's in CB's Test	<1
*	22	Refresh Test	10
	26	Random Data Test	<1
*	24	Fast Galloping Pattern Test	20
*	31	Sob-a-long Test	3
*	32	Write Recovery Test	<1
*	33	Branch Gobble Test	35
	34	Soft Error Test	<1

@ - Run only on the first Pass when under ACT or APT

+ - Run twice for each 16K Bank if Interleaved

% - Run only for MF11S-K

At the end of each Pass the program will run cleanup Patterns #30, and #27 for all banks.

2.3 Special Environments

2.3.1 XXDP -

The first pass will be a quick verify pass if and only if it is in chain mode.

2.3.2 ACT & APT Automatic Mode -

The program will not create double bit errors (DBE's) after the 1st pass.

2.3.2.1 APT Execution Times -

Here are some measured execution times for an 11/44 with cache under APT

	1st QV Pass	2nd Pass & onward
128K MS11-M (non-interleaved)	10 min 15 sec	7 min 40 sec
128K MS11-L	9 min 50 sec	7 min 30 sec
256K MS11-M (interleaved)	19 min 50 sec	14 min 45 sec

The first pass will be a quick verify pass

NOTE

Even though the first pass is a QV pass it takes longer than the subsequent non-QV passes due to the fact that it is running more patterns, some of which (patterns #24 and #33 for example) can be extrememly time consuming.

2.3.2.2 APT Environment Table -

The following table gives some of the standard settings for the APT E-Table. They may be modified as noted as the user sees fit.

FIRST PASS RUN TIME:
This parameter should be set according to the amount and type of memory to be tested. The above table (APT Execution Times) gives some measured times. For any patterns deleted (through use of the Device Descriptor Words) reference section 2.2 for individual pattern times.

NOTE

The times given in section 2.2 are for 16K chunks of memory, not 128K boards!

LONGEST TEST TIME:
This parameter should be set to the execution time of the longest pattern being run. for the default case this is 35 seconds for Pattern #33.

ADDITIONAL RUN TIME:
Not Used By Program.

SOFTWARE ENVIRONMENT:
For APT auto mode this parameter should be set to a "1". For dump mode set this to a "0".

ENVIRONMENT MODE:
When this parameter is set to a "0" the program does it's own sizing. If the users sets bit #7 however, he must specify the types and amounts of memory to be tested.

SWITCH 1:
The default setting of this switch is "101". APT uses this as the switch register for the program. Reference section 2.4.1 for more information on switch settings.

SWITCH 2:
This switch, if set to any non-zero number, is used to limit the amount of passes APT will make. The program will hang after this count has been reached.

CPU OPTIONS:
Not Used By Program.

MEMORY TYPE n (n=1 to 4)
If bit #7 of ENVIRONMENT MODE is set these four words are used to log the different types of memory to be tested. If bit #7 is not set these location are not used.

MAXIMUM ADDRESS n (n=1 to 4)
These four words are used in conjuntion with the corresponding MEMORY TYPE words to indicate the highest address that memory type occupies.

NOTE

The above two parameters do not actually have to represent an accurate configuration of memory. All the program looks for is an accurate tally of memory amount!

INTERRUPT VECTOR n (n=1 to 2)
Not Used By Program.

BUS PRIORITY n (n-1 to 2)
Not Used By Program.

BASE ADDRESS:
Not Used By Program.

DEVICE MAP:
Not Used By Program.

CONTROLLER DESCRIPTOR CODE n (n 1 to 2)
Not Used By Program.

DEVICE DESCRIPTOR CODES:
The Device Descriptor codes are used by the program to determine which patterns it will run. The default values of these words are all "1"s, indicating that all of the patterns shown in section 2.2 are executed (save for exceptions as noted there). Each set of words controls a table in the program as follows:

DD WORDS	PROGRAM TABLE (Symbolic location)
Words 0-1	MKCSRT
Words 2-3	MKPAT
Words 4-5	MJPAT

Bit #0 set in the first word indicates that the first pattern in the table will be executed, bit #1 the second, bit #2 the third,... bit #0 of the second word indicates that the 17th entry in the table will be executed, and so on.

2.3.3 No SBE Free Banks -

If the program cannot find any SBE (Single Bit Error) free locations (in non-protected ECC memory) it will print out an error message and continue testing by-passing the ECC logic tests.

2.3.4 Mixed Parity & ECC Configurations -

The program will function normally in mixed environments. The sequence of testing may seem strange due to the recursive test mode algorithm (reference sections 2.3.1.1, 2.3.1.2, & 2.3.1.3).

2.4 Program Options

2.4.1 Switch Register Details -

If a hardware switch register is not available then the software switch register is in location 176. IF under APT if BIT7 is set in the E-TABLE symbolic location '$ENVM' the APT software switch register will be used (location $SWREG).

To change the software switch register contents: Type ''control G''. This will cause display the current value of the SWR and prompt for the octal input of the new SWR value from the terminal. This routine will ignore you (not respond to control 'G') if you have a hardware switch register.

SW15 = HALT ON ERROR
(100000)

Continuing from this halt will first check for a change in the software switch register (''Control G'' in the TTY input buffer) then it will continue testing.

SW14 - LOOP ON TEST
(40000)

This will cause looping on the present test or pattern (back to last scope trap). If in a pattern then the looping will be for an entire bank of 16K addresses.

SW13 = INHIBIT ERROR TYPEOUTS
(20000)

This will cause returns from the error routine without the typed messages. Other on error functions are not affected.

SW12 INHIBIT RELOCATION
(10000)

This prevents the program from moving and consequently prevents the program from testing at least 32K of memory.

SW11 = QUICK VERIFY
(4000)

If this switch is selected approximately one 64th of the possible combinations of SBE's & DBE's are tested.

Each pass complete typeout will indicate this mode by preceding the pass number with 'QV'.

SW10 - BELL ON ERROR
(2000)

This causes a bell (or beep or click) on each error trap

SW9 LOOP ON ERROR
(1000)

This will cause looping from failure point back to the last correctly initialized area of the current test.

SW8 - HALT PROGRAM
(400)

This initiates the following sequence:

1. If program is relocated it moves back to bank zero.
2. Flush out all possible DBE's.
3. Turns off Memory Management.
4. Restore loaders.
5. Unmap the Unibus Map (if there is one).
6. Halt if under APT or ACT branch sel.

SW7 = DETAILED ERROR REPORTS
(200)

After any normal error report is typed this option causes the contents of the following registers to be typed:
R0, R1, R2, R3, R4, R5, SP, ''CONTROL'', ''CPUERR''

SW6 = INHIBIT CONFIGURATION MAP
(100)

This inhibits the printing of a map showing the memory configuration - reference section 7.4.

SW5 = LIMIT MAX ERRORS PER BANK
(40)

This will limit the number of error typeouts per bank. The default is 10. DECIMAL, however this can be changed by changing location ''ERRMAX'' manually or with ODT (reference Section 2.3.4.2).

SW4 FAT TERMINAL
(20)

This informs the program that the console terminal has a width of at least 132 columns (LA36 with wide paper).

SW3-1 = TEST MODE

Test modes determine the recursion algorithm to be used during pattern tests.

	MODE	NAME	DESCRIPTION
(0)		0	BAFPAF Banks forward, patterns forward
(2)		1	BAFPAR Banks forward, patterns reverse
(4)		2	BAWPAF Banks worst first, patterns forward.
(6)		3	BAWPAR Banks worst first, patterns reverse.
(10)		4	PAFBAF Patterns forward, banks forward
(12)		5	PAFBAW Patterns forward, banks worst first
(14)		6	PARBAF Patterns reverse, banks forward
(16)		7	PARBAW Patterns reverse, banks worst first.

For more details reference section 2.3.1.1, 2.3.1.2 and 2.3.1.3.

SW0 = DETECT SINGLE BIT ERRORS (SBI's)
(1)

For manufacturing purposes this switch should always be on. For field service purposes this switch should always be off.

This switch will allow all ECC Single Bit errors to be reported by disabling error correction.

Error printouts of SBE's are not distinguishable from DBE's.

NOTE

If Double Bit Errors are found in the memory, this switch should be set to make sure that new data can be written to the DBE locations.

2.4.1.1 Test Mode Example -

Example analysis of mode 5 'PAFBAW'. Assume Banks 0 & 1 are MS11-L and Banks 2,3,4,& 5 are MS11-M.

Assume also that Bank 3 is known bad by the program via the sizing routine or previous runs (reference section 3.1 and 6.1 test 22). The testing sequence would be as follows:

```
;TEST MS11-M MEMORY TYPES FIRST
;TEST KNOWN BAD MEMORY (BANK 3)

PATTERN 17,     BANK 3
PATTERN  7,     BANK 3
PATTERN  1      BANK 3
PATTERN  2,     BANK 3
PATTERN  4,     BANK 3
PATTERN  5,     BANK 3
PATTERN 21,     BANK 3
PATTERN 20,     BANK 3
PATTERN 22,     BANK 3
PATTERN 26,     BANK 3

;TEST PRESUMED GOOD MEMORY (BANKS 2,4,5)

PATTERN 17,     BANK 2
PATTERN  7,     BANK 2
PATTERN  1,     BANK 2
PATTERN  2,     BANK 2
PATTERN  4,     BANK 2
PATTERN  5,     BANK 2
PATTERN 21,     BANK 2
PATTERN 20,     BANK 2
PATTERN 22,     BANK 2
PATTERN 26,     BANK 2
PATTERN 17,     BANK 4
PATTERN  7,     BANK 4
PATTERN  1,     BANK 4
PATTERN  2,     BANK 4
PATTERN  4,     BANK 4
PATTERN  5,     BANK 4
PATTERN 21,     BANK 4
PATTERN 20,     BANK 4
PATTERN 22,     BANK 4
PATTERN 26,     BANK 4
PATTERN 17,     BANK 5
PATTERN  7,     BANK 5
PATTERN  1,     BANK 5
PATTERN  2,     BANK 5
PATTERN  4,     BANK 5
PATTERN  5,     BANK 5
PATTERN 21,     BANK 5
PATTERN 20,     BANK 5
```

```
PATTERN 22,     BANK 5
PATTERN 26,     BANK 5


;RELOCATE  TEST PROGRAM SPACE (BANK 0 & 1)

PATTERN  1,     BANK 0
PATTERN  2,     BANK 0
PATTERN  3,     BANK 0
PATTERN  4,     BANK 0
PATTERN  5,     BANK 0
PATTERN 26,     BANK 0
PATTERN  1,     BANK 1
PATTERN  2,     BANK 1
PATTERN  3,     BANK 1
PATTERN  4,     BANK 1
PATTERN  5,     BANK 1
PATTERN 26,     BANK 1
```

NOTE

This is an example & not an actual sequence.

The pattern sequence was forward (the simple patterns first, complex patterns last) sequence of patterns (MS11-M = 17, 7, 1, 2, 4, 5, 21, 20, 22, 26)(MS11-L = 1, 2, 3, 4, 5, 26).

If the bank selection is forward the banks will be tested in the following order:

1. ECC banks that are not protected or program space (from 0 to 200).
2. Parity banks that are not program space (from 0 to 200).
3. The program now relocates & tests:
4. ECC banks that were protected or program space (from 0 to 200).
5. Parity banks that were program space (from 0 to 200).

If bank selection is worst first the configuration table will be consulted and banks will be tested in the following order.

1. ECC banks that are known bad and are not protected or program space (from 0 to 200).
2. Parity banks that are known bad and are not program space (from 0 to 200).
3. ECC banks that are presumed good and are not protected or program space (from 0 to 200).
4. Parity banks that are presumed good and are not program space (from 0 to 200).
5. The program now relocates & tests:
6. ECC banks that are known bad and were protected or program space (from 0 to 200).
7. Parity banks that are known bad and were program space (from 0 to 200).
8. ECC banks that are presumed good and were protected or program space (from 0 to 200).
9. Parity banks that are presumed good and were program space (from 0 to 200).

2.4.1.2 Test Mode Details -

MODE 0 = 'BAFPAF'' banks forward, patterns forward

This is the default and simplest mode.

This mode tests each bank completely from 0 to 200 except those requiring relocation*.

While testing each bank the patterns are run with the simple ones first building to the more complex.

MODE 1 = 'BAFPAR'' = banks forward, patterns reverse

This mode tests each bank completely from 0 to 200 except those requiring relocation*.

While testing each bank the patterns are run with the most complex ones first, working to the simple ones.

MODE 2 = 'BAWPAF'' = Banks worst first, patterns forward

This mode first tests each known bad bank completely from 0 to 200 except those requiring relocation*, then presumed good banks are tested from 0 to 200 except those requiring relocation*.

While testing each bank the patterns are run with the simple ones first, building to the more complex.

MODE 3 'BAWPAR'' = Banks worst first, patterns reverse

This mode first tests each known bad bank completely from 0 to 200 except those requiring relocation*, then presumed good banks are tested from 0 to 200 except those requiring relocation*.

While testing each bank the patterns are run with the most complex ones first, working to the simple ones.

MODE 4 'PAFBAF'' - Patterns forward, banks forward

This mode tests each pattern completely with the simple ones first, building to the more complex.

While testing each pattern the banks are run from 0 to 200 except those requiring relocation*.

MODE 5 = 'PAFBAW'' = Patterns forward, banks worst first

This mode tests each pattern completely with the simple ones first, building to the more complex.

While testing each pattern first each known bad bank from 0 to 200 except those requiring relocation* is run, then presumed good banks are run from 0 to 200 except those requiring relocation*.

MODE 6 - 'PARBAF'' = Patterns Reverse, Banks Forward

This mode tests each pattern completely with the most complex ones first, working to the simple ones.

While testing each pattern the banks are run from 0 to 200 except those requiring relocation*.

MODE 7 - 'PARBAW'' = Patterns Reverse, Banks Worst First

This mode tests each pattern completely with the most complex ones first, working to the simple ones.

While testing each pattern first each known bad bank from 0 to 200 except those that require relocation* is run, then presumed good banks are run from 0 to 200 except those requiring relocation*.

NOTE

* Relocation is required to test the bank(s) in program space and also to test any ECC banks protected by diagnostic checkmode with the inhibit mode pointer off (zero)!

2.4.1.3 Test Mode Applications -

1. To verify correct operation of the memory system use Mode 0 'BAFPAF''.

   Advantages: Easy to understand.

   Disadvantages: In case of a failing Bank, it may take a long time to find the failure.

2. To get detailed error information on known bad Banks (found by sizing routine) use Mode 2 'BAWPAF''.

   Advantages: Seeks Bad Banks. Easy to understand.

   Disadvantages: Failures other than zeros & ones may take a long time to find.

3. To get good error info on any memory problem fast use Mode 4 'PAFBAF''.

   Advantages: Covers all banks fast. Easy to understand.

   Disadvantages: Failures from only complex patterns may take a long time to find.

4. To find any problem fast use Mode 7 'PARBAW''.

   Advantages: Covers all Banks fast.

   Disadvantages: Difficult to understand failures reported are not necessarily the most basic failure modes.

2.4.2 Display Register -

A software display register exists in location 174 in addition to any hardware display existence.

Display fields are as follows:

```
      15 ! 14  13  12  11  10   9   8 ! 7   6   5 ! 4   3   2   1
Relocated !          Bank  #          ! Not Used  .    Pattern #
          .                           .           !
==========================================================================
```

PATTERN # = The number of the pattern presently being run. All patterns are described in section 6.2. Any pattern can be found in the Diagnostic by Looking up the symbolic Tags 'MTOONN'' and 'MTPONN'' - where 'NN' is the Pattern number. MTOONN refers to the routine that sets up for the test Pattern whereas MTPONN is the actual pattern itself.

NOTE

The pattern # is not necessarily an indication of degree of difficulty.

BANK = The number of the Bank (16K) of memory under test (0-200). these bits directly map to physical address bits (21:15).

RELOCATED This bit indicates that the program is relocated and no longer in Bank 0. It will be relocated to the first known good non-protected memory bank indicated on the configuration map (reference section 7.5).

NOTE

Another way to obtain this information is to type a CONTROL/T at the console (reference Section 2.3.4.5).

2.4.3 Special Memory Locations -

2.4.3.1 CACHE Constant -

The CACHE constant is located at symbolic location ''CACHK'' and is used to enable CACHE.

NOTE

Bit 0 in the CACHE constant has no effect since it is unconditionally set by the program whenever it tries to enable CACHE.

2.4.3.2 Configuration Table -

The configuration table is located at symbolic location ''CONFIG'' and has the following format:

```
CONFIG:  First 16K Configuration words (2 each)
         2nd   16K Configuration words (2 each)
         ..........................
         200th 16K configuration words (2 each)

Configuration Words:
                LOW:    BIT 0      ERRORS PRESENT
                        BIT 1      MEMORY EXISTS
                        BIT 2-4    RESERVED
                        BIT 5      SKIP ECC LOGIC TESTS FLAG (1=SKIP)
                        BIT 6      PROTECTED REGION OF AN ECC MEMORY
                        BIT 7      PROTECTED (PROGRAM SPACE)
                        BIT 8-11   CSR CODE
                        BIT 12-15  INTERLEAVED CSR CODE
                MED:    BIT 0-7    NUMBER OF ERRORS
                        BIT 8-10   MEMORY TYPE
                        BIT 11     CSR TESTED OK
                        BIT 12     INTERLEAVE ENABLED
                        BIT 13     ''BACKGROUND PATTERN VALID'' FLAG
                        BIT 14     BANK SELECTED FOR TEST BY FIELD SERVICE MODE
                        BIT 15     LOADERS HOME BANK
```

This table is used as the source for the configuration Map (reference. section 7.4).

2.4.4 Terminal Commands -

2.4.4.1 Control "C" -

This command will:

1. If Switch 8 (Halt P)gram) in the switch register is set halt the program.
2. If Switch 8 is not set, unrelocate if program was relocated.
3. Flush out any DBE's.
4. Turn off Memory Management.
5. Attempt to Boot RK05 Drive 0.
6. Failing 4, attempt to Boot RK04 Drive 1.
7. Failing 5, go to 4.

This command will only be recognized at the completion of the current test or pattern, or at the end of a line of an error message.

2.4.4.2 Control "D" (Debug) -

This command will enter a modified version of ODT.

This is a subset of the RT11 ODT with the addition of Control "C", Control "F", & control "E" whose functions are outlined here.

NOTE

It is advisable not to enter ODT when the program is relocated.

2.4.4.2.1 Standard ODT Commands -

RETURN	Closes open location and accepts next command.
LINE FEED	Closes current location and opens next sequential location.
-	Negates number typed in.
^	Opens previous location.
_	Indexes contents of opened location by contents of PC and opens resulting location.
>	Uses contents of open location as a relative branch, and opens the resulting location.
<	Returns to sequence prior to @, >, or , and opens next location.
@	Uses contents of open location as absolute address and opens the referenced location.
r/	Opens word at location 'r'.
/	Reopens last location opened.
r\	Opens byte at location 'r'.
\	Reopens last location opened as a byte.
$n/	Opens general register 'n'.
$B/	Opens first word of te breakpoint table.
$C/	Opens constant register.
$F/	Opens format register.
$M/	Opens first mask register.
$P/	Opens priority register.
$R/	Opens first relocation register.
$S/	Opens status register.
;B	Removes all breakpoints.
r;nB	Sets breakpoint 'n' at location 'r'.
;nB	Removes breakpoint 'n'.
r;G	Starts execution of program at location 'r'.

r;O	Calculates offset from currently open location to 'r'.
;P	Execution proceeds from breakpoint.
n;P	Execution proceeds from breakpoint for the next 'n' instructions.
;kP	Execution proceeds from breakpoint; stops after encountering the breakpoint 'k' times.
;S	Disables single-instruction mode.
;nS	Enables single-instruction mode; disables breakpoints.

2.4.4.3 Control 'E'' (procEEd) -

This command will allow you to exit ODT. It is the reverse of the control 'D'' command. If you type control 'E'' after a Breakpoint - you're on your own!

2.4.4.4 Control 'K'' (Kill error printout and skip pattern) -

This command will allow you to stop an error printout and skip to the next pattern. This is handy, for example, when you have a whole bank full of errors, have gotten enough information, and wish to skip to the next pattern.

2.4.4.5 Control 'T'' (Tell me what's happening) -

This command will print out the information encoded in the display register. This is mainly intended for CPU's without a hardware display register.

Example:

RELOCATED BANK= 23 PAT= 26

2.4.4.6 Control 'S'' (Stop) -

This command will stop typeout (soon) and will wait for a Control 'Q''.

2.4.4.7 Control 'Q'' (Quintinue) -

This command will continue typing that has been stopped by Control 'S''. If there has been no Control ''S'' typed then this command is ignored.

2.4.4.8 Control 'F'' (Field Service mode) -

This command will cause you to enter a mode which looks for sub commands.

When the program is looking for a sub command any number that is not a legal command will cause a mini help message to be typed. Therefore when in doubt type 99 (CR) and you will get help.

NOTE

Typing just carriage return is a default command 0.

2.4.4.8.1 Field Service Command 0 (Exit) -

This command will exit Field Services Mode and return to whatever task it was in prior to typing control 'F''. Note typing just carriage return is a default Command 0.

2.4.4.8.2 Field Service Command 1 (Read CSR) -

This command will typeout the contents of the CSR.

If there is more than one CSR on the CPU (or if the program has not determined the CSR status yet), it will Ask you 'WHICH CSR(0-F)'' to which you must respond with an Hexidecimal number from 0 to F. Note typing just carriage return is a default 0.

If the CSR you se * causes a trap to 4 the program will type 'THIS CSR DOES NOT EXIS' .

NOTE

CSR references are done in accordance with section 5.1.

2.4.4.8.3 Field Service Command 2 (Load CSR) -

This command will enable you to load the CSR.

If there is more than one CSR on the CPU (or if the program has not yet determined the CSR status yet) it will ask you 'WHICH CSR(0-F)'' to which you must respond with an Hexidecimal number from 0 to F. Note typing just carriage return is a default 0.

If the CSR you select causes a trap to 4 the program will type ''THIS CSR DOES NOT EXIST''.

The CSR will be read and displayed as in command 1.

The program will then ask you for the ''CSR?'' to which you must respond with an Octal number. Note typing just carriage return is a default 0.

The program will then load the CSR and Read it again displaying its new contents.

2.4.4.8.4 Field Service Command 3 (Examine Memory) -

This command will allow you to examine any physical address and does the necessary memory management mapping for you.

The program will ask you for the 'PHYSICAL ADDRESS (0-17757776)'' to which you must respond with an Octal number.

If the address access causes a trap to 4 the program will type 'TIME-OUT TRAP''. If the address access causes a trap to 114 the program will type 'PARITY ABORT''.

The contents of your physical address will be typed.

2.4.4.8.5 Field Service Command 4 (Modify Memory) -

This command allows you to modify any physical address and does the necessary memory management mapping for you.

The program will ask you for the 'PHYSICAL ADDRESS (0-17757776)'' to which you must respond with an Octal number.

If the address access causes a trap to 4 the program will type 'TIME-OUT TRAP''. If the address access causes a trap to 114 the program will type 'PARITY ABORT''.

The program will type 'OLD DATA WAS'' and the contents of your physica address.

The program will then type ''INPUT NEW DATA'' to which you must respond with an Octal number. Note typing just carriage return is a default 0.

The program will attempt to write this new data into your physical address after which it will read it again and type 'DATA IS NOW'' and the new contents of your physical address.

NOTE

If you can't change the data, that would indicate that you have a Double Bit Error in that double word pair.

2.4.4.8.6 Field Service Command 5 (Select Bank & Pattern) -

This command allows you to run any bank with any pattern forever.

The program will ask you 'BANK(0-200)'' to which you must respond with an Octal number. If the bank is not accessible. The program will type 'BANK NOT ACCESSIBLE'' and ask question over.

The program will then ask 'PATTERN (0-37)'' to which you must respond with an Octal number.

Any pattern can be run including those that are not part of the

APT E-TABLE defaults (reference section 6.2.1). If you select Pattern 0, the program will ask 'PATTERN 0 DATA IS?'' to which you must respond with an Octal number.

If the Bank you selected requires relocation the program will type 'BANK REQUIRES RELOCATION'' and exit this command. Note normally this is true for Bank 0.

The program will then arm the console keyboard for interrupts and type 'TO ESCAPE TYPE ANY KEY!''.

The test pattern will be entered and run until a console key is depressed to escape this loop.

2.4.4.8.7 Field Service Command 6 (Type Configuration Map) -

This command types the configuration map.

This is useful after a long run (overnight) to see all the banks that are marked as bad. (Especially if your console is a video terminal).

For a detailed explanation of the map reference section 7.3.

2.4.4.8.8 Field Service Command 7 (Battery Backup Test) -

This command will check that memory does not forget while powered down.

It will write and check the address pattern in every non-protected Memory Bank.

It will then prompt with 'REMOVE SYSTEM POWER FOR 10 SECONDS MAX!".

If you change your mind and do not wish to power down the system type any character and the program will think that it has gone thru a power fail/auto restart sequence.

The program will then read check each non-protected bank for the address pattern and when done type "TEST COMPLETE" and exit this command.

2.4.4.8.9 Field Service Command 8 (SOB-A-LONG TEST) -

This command allows execution of the SOB-A-LONG Test on all non-protected Banks reference Section 6.2.2.26. Operation is identical to command 5 except that no Pattern or Bank is entered and each pass causes a Bell.

2.4.4.8.10 Field Service Command 9 (Error Summary) -

This command types out the number of passes and the total number of errors. If there were any errors it will type out the Banks and the number of errors per bank up to 255 DECIMAL.

This becomes useful after long runs (all night) on systems with a video console terminal.

2.4.4.8.11 Field Service Command 10 (Refresh TEST) -

This command allows execution of the Refresh Test on all non-protected Banks reference Section 6.2.2.19. Operation is identical to command 5 except that no Pattern or Bank is entered and each pass causes a Bell.

2.4.4.8.12 Field Service Command 11 (Set Fill Count) -

This command allows setting of the terminal fill count (necessary for LA30's, ASR33's, and VT05's). It is normally set to zero for LA36's, VT52's, VT100's, etc.

2.4.4.8.13 Field Service Command 12 (Enter Kamikaze Mode) -

This command allows you to run patterns that are normally not executed unless under APT or ACT. They are usually very time consuming and can result in failures that are fatal to the program. In effect you are trying to find a hardware failure regardless of the consequences. Note that most crashes do not wipe out the display information which is telling you what the program was doing just prior to failure. There are two ways to die here - Impatience and Crashes.

2.4.4.8.14 Field Service Command 13 (Exit Kamikaze Mode) -

Return to the default mode of testing (undo Command 13).

2.4.4.8.15 Field Service Command 14 (Turn Cache Off) -

This changes the Cache constant to bypass cache (reference section 2.3.3.1).

2.4.4.8.16 Field Service Command 15 (Turn Cache On) -

This changes the Cache constant to use cache (reference section 2.3.3.1).

2.4.4.8.17 Field Service Command 16 (Test Only Selected Banks) -

This command allows you to center the test effort on only those banks that you are troubleshooting. You may also test banks that require relocation and were inaccessable via command 5.

2.4.4.8.18 Field Service Command 17 (Resume Testing All Banks) -

Return to the default mode of testing (undo Command 17).

2.5 Execution Times

2.5.1 Typical (System) -

Execution time depends on many variables; however here are some measured times on an 11/44 with cache:

128K words of MS11-L Memory
Normal Pass 0 Min 50 Sec
Quick Verify 0 Min 50 Sec
Kamikaze Mode 10 Min 5 Sec
Kamikaze QV 10 Min 5 Sec

128K words of MS11-M Memory (Non-Interleaved)
Normal Pass 2 Min 25 Sec
Quick Verify 1 Min 0 Sec
Kamikaze Mode 11 Min 0 Sec
Kamikaze QV 10 Min 30 Sec

128K words of MS11-M Memory (Interleaved)
Normal Pass 3 Min 55 Sec
Quick Verify 1 Min 50 Sec
Kamikaze Mode 22 Min 0 Sec
Kamikaze QV 20 Min 5 Sec

2.5.2 Calculations (System) -

Normal Pass
Add 16 Sec per BANK of Non-Intereaved MS11-M
Add 16 Sec per BANK of Interleaved MS11-M
Add 8 Sec per BANK of MS11-L

Quick Verify Pass
Add 7 Sec per BANK of Non-Interleaved MS11-M
Add 7 Sec per BANK of Interleaved MS11-M
Add 8 Sec per BANK of MS11-L

Kamikaze Mode
Add 10 min. per 128K words for approximate pass times.

2.5.3 Typical (Patterns) -

Pattern	Time	Description
MT0000	;<1 SEC	DATA PATTERN TEST
MT0001	;<1 SEC	ADDRESS TEST
MT0002	;<1 SEC	COMPLEMENT ADDRESS TEST
MT0003	; 1 SEC	3 XOR 9 WORST CASE NOISE TEST
MT0004	; 1 SEC	ROTATING ZEROS TEST
MT0005	; 1 SEC	ROTATING ONES TEST
MT0006	;<1 SEC	INITIAL DATA TEST
MT0007	;<1 SEC	ADDRESS BIT TEST
MT0010	;<1 SEC	BYTE ADDRESSING TEST
MT0011	;<2 SEC	CREATE SINGLE BIT ERROR TEST
MT0012	;<1 SEC	WRITE BYTE CLEARS SBE TEST
MT0013	; 1 SEC	CREATE DOUBLE BIT ERROR TEST
MT0014	; 1 SEC	WRITE INHIBIT DURING DATIP WITH DBE
MT0015	; 1 SEC	WRITE INHIBIT OF BYTE WITH DBE
MT0016	;<1 SEC	WRITE INHIBIT OF WORD WITH DBE
MT0017	;<1 SEC	HOLDING 1'S 0'S TEST
MT0020	;<1 SEC	MARCHING 1'S 0'S IN CHECK BITS
MT0021	; 1 SEC	MARCHING 0'S 1'S TEST
MT0022	;10 SEC	REFRESH TEST
MT0023	;10 SEC	SHIFTING DIAGONAL TEST
MT0024	;20 SEC	FAST GALLOPING PATTERN TEST
MT0025	;<1 SEC	INTERRUPT ENABLE TEST
MT0026	;<1 SEC	RANDOM DATA TEST
MT0027	; 1 SEC	UNIQUE BANK TEST
MT0030	; 1 SEC	FLUSH OUT DBE'S TEST
MT0031	; 3 SEC	SOB-A-LONG TEST
MT0032	;<1 SEC	WRITE RECOVERY TEST
MT0033	;35 SEC	BRANCH GOBBLE TEST
MT0034	;<1 SEC	SOFT ERROR TEST
MT0035	;<1 SEC	WORST CASE PARITY TEST

3.0 ERROR INFORMATION

3.1 Error Reporting

Most errors are reported using the EMT trap and handler provided by SYSMAC.SML. Most errors will be of the 'MEMORY DATA ERROR' type which will be described here. MEMORY DATA ERRORS will also cause the bank to be marked as Bad in the configuration table.

Other errors are best explained by referencing the specific typeout and if necessary the program listing.

Example 1:

```
MEMORY DATA ERROR
  PC     BANK  VADD     PADD      GOOD    BAD     XOR    CSR MTYP  INT PAT
022132    37   060006  03700006  000000  000100  000100  0   M     -   06
022132    37   060006  03700006  000000  000100  000100  0   M     -   06
022132    37   060006  03700006  000000  000100  000100  0   M     -   06
022132    37   060006  03700006  000000  000100  000100  0   M     -   06
```

While testing Bank 37 at virtual address 60006 (virtual addresses are always between 60000 and 157776 for mapping purposes), physical address 3700006 (that's Bank 37 physical 6 within the Bank) with Pattern 6 (Initial Data Test), the good data expected was 0 but the data actually read (BAD) was 100, the exclusive OR at Good & Bad yields 100 which indicates only failing bit(s) (Bit 6). It is an MS11-M (ECC) Memory and it's not interleaved. The CSR is located at 172000.

Example 2:

```
MEMORY DATA ERROR
  PC     BANK  VADD     PADD      GOOD     BAD      XOR   CSR MTYP  INT PAT
022132    35   060000  03500000  000000   000001   000001  0   M     1   06
022132    35   060002  03500002  000000   000100   000100  0   M     1   06
022132    35   060006  03500006  000000   000100   000100  0   M     1   06
```

While testing Bank 35, virtual address 60000, physical address 3700000 with Pattern 6 (Initial Data Test), the good data expected was 0 but the data actually read (BAD) was 1, the exclusive OR at Good & Bad yields 1 which indicates only failing bit(s) (Bit 0). It is an MS11-M (ECC) Memory and it's interleaved; so since Address Bit 1 was not asserted, the CSR is located at 172000.

While also in Bank 35, virtual addresses 60002 and 60006 were expected to have 0, but the data read was 100, the exclusive OR of Good & Bad yields 100 which indicates one failing bit (Bit 6). Since it is interleaved MS11-M memory, and Address Bit 1 is asserted, the CSR is located at 172102 (CSR number 1 under the INT column)

NOTE

Subsequent errors of the same test do not type a new heading.

3.2 Error Abbreviations

The following is a list of ll abbreviations used in error reports.

# OF ERRORS	Number of Errors that were detected.
1ST ADD	First Address that failed.
ARRAY	The array number that was locked up in the MS11-M CSR.
APT#	The # of CPU's APT expects on the system.
APTCORE	APT Core size.
APTMOS	APT MOS size.
BAD	Bad data.
BAD-WD1	Bad Word #1 of a double word data value.
BAD-WD2	Bad Word #2 of a double word data value.
BAD-CHK	Bad Check Code Bits.
BANK	The Bank number. Banks are 16K words long.
BD-CC	Bad Check Code Bits.
CHKBITS	The 7 bit value of the Check Code Bits.
CONTRL	The CACHE Control register.
CPUERR	CPU Error register.
CSR	Control and Status Register.
CSRNO	CSR NUMBER (0-F Hexidecimal).
DATARG	The CACHE Data Register.
DBE	Double Bit Error (uncorrectable error).
DEV ADD	Device Address.
ECC	Error Correctable Code.
GD-CC	Good Check Code Bits.
GD-CHK	Good Check Code Bits.
GD-WD1	Good Word #1 of a double word data value.
GD-WD2	Good Word #2 of a double word data value.
GOOD	Good data.
INT	Interleaved (Address Bit 1 asserted) CSR number.
LSIZE	MS11-L Size.
MEMERR	Memory Error register.
MMR0	Memory Management Register #0.
MMR1	Memory Management Register #1.
MMR2	Memory Management Register #2.
MMR3	Memory Management Register #3.
MSIZE	MS11-M Size.
MTYP	Memory Type (MS11-L,MS11-M,MF11S-K,BIPOLAR or UNIBUS Parity).
PADD	Physical Address (asserted by the program after mapping).
PAT	Pattern number.
PC	Program Counter at the time the error occurred.
SBE	Single Bit Error (correctable error).
VADD	Virtual Address (asserted by the program before mapping).
WROTE1	The data that was written into the 1st half of a double word.
WROTE2	The data that was written into the 2nd half of a double word.
XOR	Exclusive OR of the good and bad data. Shows the bad bits.

3.3 Error Halts

There are several Halts in the program.

All unused trap vectors contain a trap catcher (.WORD .+2,HALT).

An undefined TRAP instruction halts at symbolic location '$HALT2''.

The APT down load sequence will halt at symbolic location ''APTHLT''.

Halt on Error option (SW15 Set) at symbolic location '$HALT''.

Halt program (SW8 Set) at symbolic location '$EXHALT''.

Power Fail will normally halt at the end of the shut down sequence (symbolic location '$DOWN'').

Power Fail has a fatal Halt at symbolic location '$ILLUP'' which can be caused by power up occurring before power down sequence completed or by power down before a power up sequence is completed.

4.0 PROGRESS REPORTS

Pass complete typeouts as follows:

END PASS	#	0
END PASS	#	1
END PASS	#QV	2

NOTE

Pass 2 was flagged as a Quick Verify Pass. (Because of a change in SW5)

To obtain progress reports while executing, typing a Control ''T'' will print out the information encoded in the display register.

Example:

BANK= 2 PAT- 34

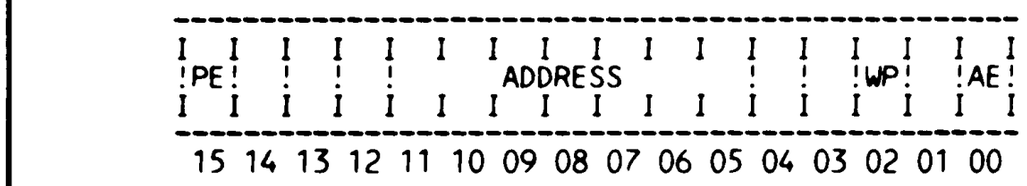
5.0 CSR INFORMATION TABLES

The following is a picture view of the current control status registers which can be tested by this program. It shows bit assignments and definitions to provide a handy reference, and shows the similarities and differences between each one:

NOTE

All unused bits in each CSR are equal to zero.

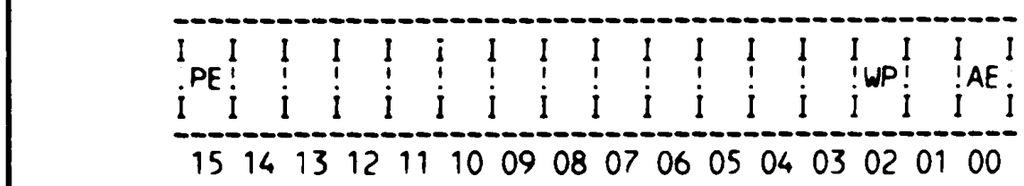
5.1 CORE/MOS PARITY REGISTER



Bit assignments are defined as follows:

BIT15	PARITY ERROR	
BITS 11-5	ERROR ADDRESS	High order address bits of address of parity error (Bits 17-11 of address).
BIT02	WRITE WRONG PARITY	Normal parity (odd) when clear; other parity (even) when set.
BIT00	ACTION ENABLE	No action when clear trap to vector 114 when set.

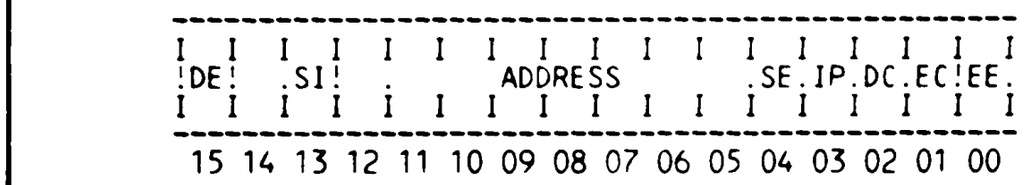
5.2 MOS BIPOLAR PARITY REGISTER (USED IN THE 11/45-55)



Bit assignments are defined as follows:

BIT15	PARITY ERROR	
BIT02	WRITE WRONG PARITY	Normal parity (odd) when clear; other parity (even) when set
BIT00	ACTION ENABLE	No action when clear trap to vector 114 when set.

5.3 MF11S-K CSR

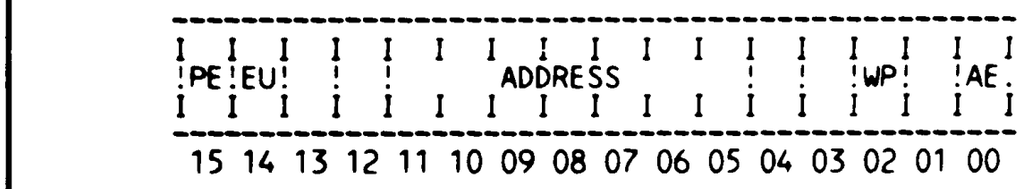


BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15	DOUBLE ERROR	Set whenever DBE occurs. If BIT2=0, the error address will be stored in Bits 11-5. If BIT2 =1, the check bits read will be stored in BITS 11-5.
BIT 13	SET INHIBIT MODE	When this bit is set to a 1, it enables the Inh Mode Pointer to inhibit either the first or second 16K from ever going into the Diag. Check or ECC Disable mode. When this bit is set to zero, the entire memory operates in in Diagnostic Check or ECC Disable Mode.
BITS 11-5	ERROR ADDRESS	With BIT02 cleared they contain the high order error address (Bits 17-11); with BIT02 set they contain the check bits for ECC.
BIT04	SINGLE ERROR	Set whenever single error occurs

BIT03	INHIBIT MODE POINTER	The Inhibit Mode Pointer works in conjunction with the Set Inhibit Mode bit. When BIT13 is set to a 1, a 16K portion of memory is inhibitted from operating in the ECC Disable mode or Diagnostic Check mode. the Inhibit Mode Pointer indicates which 16K is being inhibited; e.g.-when BIT 3 =1, the second 16K of memory is inhibitted. When BIT 13 is set to a 0, BIT 3 becomes inoperative.
BIT02	DIAGNOSTIC CHECK MODE	When set enables read-write of check bits(see Bits 11-5). If a DBE occurs in this mode (with BIT1 =0), BIT15 in the CSR is set but the check bits from memory are stored in CSR Bits 11-5 and not the DBE address bits.
BIT01	DISABLE ERROR CORRECTION	When set no single error correction takes place and the error is not logged in the csr; correct check bits are still written to the memory however.
BIT00	DOUBLE ERROR ENABLE	When set enables trap to vector 114 on double error.

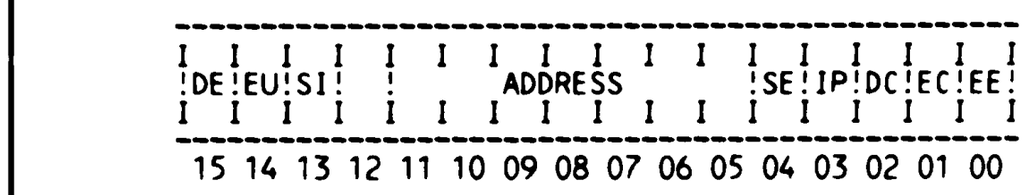
5.4 MS11-L CSR



Bit assignments are defined as follows:

BIT15	PARITY ERROR	
BIT14	EUB ERROR RETRIEVAL	If the memory is on an Extended UNIBUS, when BIT14 is zero, the low order failing addresses are available (Bits 11-17); when BIT14 is one, the high order failing addresses are available (Bits 18-21 of address). If the memory is on a UNIBUS, a jumper disables this bit so that it is read only, and equal to zero.
BITS 11-5	ERROR ADDRESS	With BIT14 set, they contain the high order parity error address (Bits 21-18 of address); with BIT14 cleared, they contain the low order parity-error address (Bits 17-11 of address).
BIT02	WRITE WRONG PARITY	Normal parity (odd) when clear; other parity (even) when set.
BIT00	ACTION ENABLE	No action when clear; trap to vector 114 when set.

5.5 MS11-M CSR



Bit assignments are defined as follows:

BIT15	UNCORRECTABLE ERROR	This bit is set if a DBE occurs, and the error address is stored in the CSR. This bit is also set in the ECC Disable mode if an SBE or DBE occurs.
BIT14	EUB ERROR RETRIEVAL	If the memory is on an Extended UNIBUS, when BIT14 is zero and either BIT4 or BIT 15 is a one, the low order failing addresses are available (Bits 11-17); when BIT14 is one, the high order failing addresses are available (Bits 18-21 of address). If the memory is on a UNIBUS, a jumper disables this bit so that it is read only, and equal to zero.
BIT13	SET INHIBIT MODE	When this bit is set to a 1, it enables the Inh Mode Pointer to inhibit either the first or second 16K from ever going into the Diag. Check or ECC Disable mode. When this bit is set to a 0, it allows the Diag. Check mode and/or ECC Disable mode to operate over the entire memory on the board.

BITS 11-5	ERROR ADDRESS	With BIT02 cleared and BIT14 set, they contain the high order error address (Bits 21-18); when BIT02 and BIT14 are cleared, they contain the low order error address (Bits 17-11); when BIT02 is set they contains check bits for ECC.
BIT04	SINGLE ERROR	Set whenever single error occurs.
BIT03	INHIBIT MODE POINTER	The Inhibit Mode Pointer works in conjunction with the Set Inhibit Mode bit. when BIT13 is set to a 1, a 16K portion of memory is inhibitted from operating in the ECC Disable mode or Diagnostic check mode. the Inhibit Mode Pointer indicates which 16K is being inhibited; e.g.-if BIT3 =1, the second 16K of memory is inhibitted. when BIT13 is set to a 0, BIT3 becomes inoperative.
BIT02	DIAGNOSTIC CHECK MODE	When set enables read-write of check bits(see Bits 11-5). If a DBE occurs in this mode (with BIT1=0), BIT15 is set, but the check bits read are stored in Bits 11-5, not the DBE address bits.

BIT01	DISABLE ERROR CORRECTION	When set no single error correction takes place. A single bit error will set BIT04 and BIT15 and assert BUS PBL L if BIT00 is asserted; a double error will set set BIT15 and assert BUS PBL L if BIT00 is asserted. The error address is stored in the CSR, and correct check bits are generated and stored on a write.
BIT00	UNCORRECTABLE ERROR ENABLE	When set enables trap to vector 114 on uncorrectable error.

6.0 SUB-TEST SUMMARIES

6.1 Tests

TEST 1

BIT TEST OF ALL CSR'S/MATCH ALL CSR'S WITH MEMORY
(CSR Access may cause wrong Type of Traps)

TEST 2

TEST BANK 0 ACCESSES
Failures are fatal.

TEST 3

TEST BANKS 1-200 (OCTAL) FOR ZEROS AND ONES
Errors are not typed here - only logged in
the configuration table

TEST 4

ECC INHIBIT MODE POINTER TEST

TEST 5

DIAGNOSTIC MODE DISPATCH ROUTINE
This test runs all the patterns in the
mode selected.

TEST 6

UNIQUE BANK TEST
Pattern 27 is run

6.2 Patterns

6.2.1 General Pattern Information -

Actual patterns are identified by symbolic locations 'MTPXYY'' where X may be any sub program indicator (A,B,C,etc) or 0 and YY will be the number of the pattern.

Setup procedures for each pattern are identified by symbolic locations 'MT00YY'' where YY will be the number of the pattern.

Patterns reside in 4 scripts that are scanned for execution. Symbolic location 'MKCSRT'' is a table of patterns that can run once for each ECC bank (twice for interleaved MS11-M's). Symbolic location 'MKPAT'' is a table of patterns that can run on each Bank of ECC memory. Symbolic location 'MJPAT'' is a table of patterns that can run on each Bank of Parity memory. Symbolic location 'FSPAT'' is a table of patterns that can be run in Field Service Mode (command 5).

The 1st 3 scripts are completely controlled by the APT E-table (even if not running under APT). Modifications to this table can be made (1) with APT, (2) manually, or (3) with ODT (reference Section 2.3.4.2).

Example E-table Segment:

```
;THE FOLLOWING LOCATIONS SPECIFY WHICH PATTERNS
;ARE TO BE RUN FOR PARTICULAR MEMORIES
;
;REFERENCE THE TABLE LISTED BELOW TO RELATE BITS TO PATTERNS.
;BIT0 SET WILL RUN THE FIRST ENTRY IN THE TABLE, BIT0 SET
;IN THE SECOND WORD WILL RUN THE 17TH ENTRY IN THE TABLE...
;
;NOTE**NULL TESTS DO NOT TAKE ANY TIME
;                                        RECOMMENDED VALUE
$DDW0:   .WORD   177777  ;ECC CSR TESTS     177777        TABLE = MKCSRT:
$DDW1:   .WORD   177777  ;ECC CSR TESTS     177777        TABLE = MKCSRT:
$DDW2:   .WORD   177777  ;ECC PATTERNS      103777        TABLE = MKPAT:
$DDW3:   .WORD   177777  ;ECC PATTERNS      177777        TABLE = MKPAT:
$DDW4:   .WORD   177777  ;PARITY PATTERNS   003777        TABLE = MJPAT:
$DDW5:   .WORD   177777  ;PARITY PATTERNS   177774        TABLE = MJPAT:
```

6.2.2 Specific Patterns -

6.2.2.1 Pattern 0 Basic Data Test -

Writes & Reads R2 into a 16K Bank.

This is used for Zeros and Ones testing and in Field Service Mode for any console selected pattern.

It can execute out of the USER Instruction PAR's.

NOTE

It is frequently modified dynamically such that (1) it returns after writing only (the 1st NOP is replaced with a RETURN) or (2) it only counts Errors (the code PERR02 and NOP are replaced with INC @#PATERR).

6.2.2.2 Pattern 1 Address Test -

Writes & Reads an incrementing pattern equivalent to physical addressed into a 16K Bank.

It can execute out of the USER Instruction PAR's.

6.2.2.3 Pattern 2 Complement Address Test -

Writes the complement of the physical address from high addresses to low (write down) and reads from low addresses to high (read up).

This provides the complement of the coverage of Pattern 1 in both data pattern and addressing sequence.

It can execute out of the USER Instruction PAR's.

6.2.2.4 Pattern 3 3 XOR 9 -

Writes & Reads a pattern that complements as address bits 3 and 9 change.

This pattern is run 4 times (1) with Zeros & Ones, (2) with Ones & Zeros, (3) with 401 & Ones, and (4) with Ones & 401. The pattern of the 401 is to force a the parity bits to become involved.

It can execute out of the USER Data PDR's, the User Instruction PAR's, the Kernel Data PAR's and the Supervisor Data PAR's.

6.2.2.5 Pattern 4 Rotating Zeros Test -

Writes a background pattern of Ones. Rotates a Zero Carry Bit left thru each par of bytes (18 times) and then checks that the carry is Zero and the word (2 bytes) is still all Ones.

It can execute out of the User Data PAR's and the Kernel Data PAR's.

NOTE

It is not uncommon to observe the good data equal to the bad data. This indicates that the carry was not clear after 18 ROLB's.

6.2.2.6 Pattern 5 Rotating Ones Test -

Writes a background pattern of Zeros. Rotates a One carry bit left thru each pair of bytes (18 times) and then checks that the Carry is a One and the Word (2 Bytes) is still all Zeros.

This provides the complement of the coverage of Pattern 4 in data.

It can execute out of the User Data PAR's and the Kernel Data PAR's.

NOTE

It is not uncommon to observe the good data equal the bad data. This indicates that the Carry was not set after 18 ROLB's.

6.2.2.7 Pattern 6 Initial Data Test -

Writes & Reads a double word first with all bits 0 except 1 (for every bit position), Second with all bits 1 except 1 (for every bit position).

This is a very quick check of the data paths.

6.2.2.8 Pattern 7 Address Bit Test -

Writes a background of all Zeros.
Read Address 1 for a 0 Byte.
Complement Address 1.
Read Address 1 for a non 0 Byte.
For each Address Bit position from Bit 1:
Virtual (2, 4, 10, 20, 40, 100, 200, 400, 1000, 2000, 4000, 10000, 60000, 20000)
Physical (60002, 60004, 60010, 60020, 60040, 60100, 60200, 60400, 61000, 62000, 64000, 70000, 140000, 100000)
  Read Address for a 0 word.
  Complement Address contents.
  Read Address for a non-zero word.

This is a very quick check of the address bit uniqueness.

6.2.2.9 Pattern 10 Byte Addressing Test -

With ECC Disabled.
Writes all ones to a double word.
For each of the 4 Bytes in the Double Word.
  Clears one byte.
  Reads all 4 bytes from double word.
  Checks for only proper byte clear.
  All other bytes set to all ones.

This is only done on one double word address.

NOTE

This is run for ECC memory only

6.2.2.10 Pattern 11 Single Bit Error Test -

1. Create a Single Bit Error.
2. Read data Uncorrected (with ECC Disable).
3. Check that SBE and DBE flags are set, and the error address is latched.
4. Read First Word of data corrected (with ECC Enabled)
5. Check that the CSR Single Bit Error Flag was set, and the error address was latched.
6. Clear SBE Flag.
7. Read Second word of data corrected (with ECC Enabled).
8. Check that the CSR Single Bit Error Flag was set.
9. Do (1-7) for a Single Bit Error in each of 32 positions of a double word.
   i.e. (32 TIMES)
10. If not in Quick Verify Mode then Do (1-8) for data consisting of 1 bit set in each of 32 positions of a double word.
    i.e. (32 X 32 = 1024 Times)
11. Do (1-9) for complemented data (1 Bit clear in each of 32 positions of a double word).
    i.e. (1024 X 2 = 2048 Times)
    or (32 X 2 = 64 Times (Quick Verify))
12. Do (1-7) for a double word equal to (000000,000000), and all possible Single Bit Error combinations forced into the Check Bits (CSR bits 5-11).
13. Clear any errors out of test locations.

This insures that all Single Bit Errors can be corrected and detected.

NOTE

This test is run for ECC memory only

6.2.2.11 Pattern 12 Write Byte Clears SBE Test -

1. Create a Single Bit Error.
2. Write a Byte of Double Word to Ones.
3. Read a Byte of Double Word.
4. If this is MS11-M, the SBE flag should be SET.
   If this is MF11S-K the SBE flag should be SET if this is the byte with the error.
5. The Byte should have been equal to Ones.
6. Do (1-5) for each of the 4 Bytes of the Double Word
7. Do (1-6) for a Single Bit Error in each of 32 positions of a Double Word
   i.e. (32 Times)
8. If not in Quick Verify Mode then do (1-7) for data consisting of 1 Bit set in each of 32 positions of a double word.
   i.e. (32 X 32 = 1024 Times)
9. Clear any errors out of test locations.

This insures that single bit errors in the data portion (not in check-bits) can be cleared by writing the corresponding byte and that writing any other byte does not change the existing single bit error.

NOTE

This test is run for ECC memory only.

6.2.2.12 Pattern 13 Create Double Bit Error Test -

1. Create a Double Bit Error.
2. Access the Data (TST instruction).
3. Check that the CSR DBE Flag is set, and the error address is latched.
4. Initialize CSR to allow parity traps on DBE's.
5. Access the Data (TST Instruction).
6. Check that a parity trap occurred.
7. Do (1-6) for the 2nd Bit of each Double Bit Error in each of 32 positions of a double word less the one position of the 1st Bad Bit.
   i.e. (31 Times)
8. If not in Quick Verify Mode then Do (1-7) for the 1st Bit of each of Double Bit Error in each of 32 positions of a double word.
   i.e. (31 X 32 = 992 Times)
9. Do (1-8) for complemented data (Ones versus Zeros in Double Word)
   i.e. (992 X 2 = 1984 Times)
   or (31 X 2 = 62 Times (Quick Verify))
10. Do (1-6) for a double word equal to (000000,000000), and all possible Double Bit Error combinations forced into each of the check bits (CSR bits 5-11).
11. Clear any errors out of test locations.

This insures that all Double Bit Errors can be created and detected and cause traps.

NOTE

This test is only run during the first (QV) PASS when under ACT or APT, and is run for ECC memory only.

6.2.2.13 Pattern 14 Write Inhibit During DATIP With DBE Test -

1. Create a Double Bit Error.
2. Do ASRB on Test Location.
3. Check that Double Word is STILL Bad (Unchanged-with DBE).
4. Do (2-3) on all 4 Bytes of Double Word.
5. Do (1-4) for the 2nd bit of each Double Bit Error in each of 32 positions of a Double Word less the one position of the 1st Bad Bit.
   i.e. (31 Times)
6. If not in Quick Verify Mode then Do (1-5) for the 1st Bit of each Double Bit Error in each of 32 positions of a double word.
   i.e. (32 X 32 = 922 Times)
7. Do (1-6) for complemented data (Ones versus Zeros in Double Word).
   i.e. (922 X 2 = 1984 Times)
   or (31 X 2 = 62 Times (Quick Verify))
8. Do (1-4) for a double word equal to (000000,000000), and all possible Double Bit Error combinations forced into the Check Bits(CSR bits 5-11).
9. Clear any errors out of test locations.

This insures that the Double Bit Error can be cleared by a DATIP to any affected Byte.

NOTE

This test is only run during the first (QV) pass when under ACT or APT, and is run for MF11S-K only.

6.2.2.14 Pattern 15 Write inhibit Of Byte With DBE -

1. Create a Double Bit Error.
2. Do a MOVB immediate to test byte.
3. Check that Double Word is still Bad (unchanged-with DBE).
4. Do (2-3) on all 4 Bytes of Double Word.
5. Do (1-4) for the 2nd Bit of each Double Bit Error in each of 32 positions of a double word less the one position of the 1st Bad Bit.
   i.e. (31 Times)
6. If not in Quick Verify Mode then Do (1-5) for the 1st Bit of each Double Bit Error in each of 32 positions of a Double Word.
   i.e. (31 X 32 = 922 Times)
7. Do (1-6) for Complemented Data (Ones versus Zeros in Double Word).
   i.e. (992 X 2 = 1984 Times)
   or (31 X 2 = 62 Times (Quick Verify))
8. Do (1-4) for a double word equal to (000000,000000), and all possible Double Bit Error combinations forced into the Check Bits (CSR bits 5-11).
9. Clear any errors out of test locations.

This insures that no Double Bit Error can be cleared by a MOVB to any affected Byte.

NOTE

This test is only run during the first (QV) pass when under ACT or APT, and is run for ECC memry only.

6.2.2.15 Pattern 16 Write Inhibit Of Word With DBE Test -

1. Create a Double Bit Error.
2. Do MOV IMMEDIATE on test location.
3. Check that Double Word is STILL Bad (unchanged-with DBE).
4. Do (2-3) on both Double Words.
5. Do (1-4) for the 2nd Bit of each Double Bit Error in each of 32 positions of a Double Word less the one position of the 1st Bad Bit.
   i.e. (31 Times)
6. If not in Quick Verify Mode then Do (1-5) for the 1st Bit of each Double Bit Error in each of 32 positions of a Double Word.
   i.e. (32 X 32 = 992 Times)
7. Do (1-6) for Complemented Data (Ones versus Zeros in Double Word).
   i.e. (992 X 2 = 1984 Times)
   or (31 X 2 = 62 Times (Quick Verify))
8. Do (1-4) for a double word equal to (000000,000000), and all possible Double Bit Error combinations forced into the Check Bits (CSR bits 5-11).
9. Clear any errors out of test locations.

This insures that no Double Bit Error can be cleared by a MOV to any affected word.

NOTE

This test is only run during the first (QV) pass when under ACT or APT, and is run for ECC memory only.

6.2.2.16 Pattern 17 Holding 1's & 0's Test -

1. Write a 16K Bank with alternating Bytes of Zeros & Ones writing a Byte at a time.
2. Read each word for correct pattern.
3. Do (1-2) again for a complement pattern.

This checks the memory for the capability of holding 0's & 1's.

SEQ 0062

6.2.2.17 Pattern 20 Marching 0's & 1's In Check Bits Test -

1. Write Double Words of 000000,,000000 which causes check bits to equal 077 while addressing increments.
   (Write Up/077 --> check bits)
2. If in Quick Verify Mode then Go to Step (5).
3. Read Double Words & check while writing 000000,,100000 and addressing decrements.
   (Down/077 --> 100)
   This flips all the checkbits.
4. Read Double Words & check while writing Zeros while addressing increments.
   (Up/100 --> 077)
5. Read Double Words & check while writing 000000,,100000 & addressing increments.
   (Up/077 --> 100)
6. Read Double Words & check while writing Zeros while addressing decrements.
   (Down/100 --> 077)
7. Read Double Words & check while Addressing increments.
   (Up/077)

This checks the integrity of the MOS chips that store the checkbits.

6.2.2.18 Pattern 21 Marching 0's & 1's Test -

1. Write a Background of alternating Bytes of Zeros & Ones

2. For the 16K Bank addressing Down
   (a) Read check a word
   (b) Byte Swap a word
   (c) Read check a word

3. For the 16K Bank addressing Up
   (a) Read check a word
   (b) Byte Swap a word
   (c) Read check a word

4. For the 16K Bank addressing Up
   (a) Read check a word
   (b) Byte Swap a word
   (c) Read check a word

5. For the 16K Bank addressing Down
   (a) Read check a word
   (b) Byte Swap a word
   (c) Read check a word

This checks the integrity of the 32 Bit Double Words.

It can execute out of the User Data PAR's.

NOTE

It is not uncommon to see a misleading error typeout because the second test in each case is based upon a byteswap of the first test which may or may not have failed. If the error report indicates errors in pairs with the bad bit in the second report being the same bit position relative to a byte then you should ignore the second error report.

6.2.2.19 Pattern 22 Refresh Test -

1. Write a diagonal pattern of ones on every KDIAG(TH) stripe & write zeros elsewhere.

   This pattern is on addresses not bit positions.

   Example:

Address	MSB's							
LSB's	0	0	0	1	0	0	0	1
	0	0	1	0	0	0	1	0
	0	1	0	0	0	1	0	0
	1	0	0	0	1	0	0	0
	0	0	0	1	0	0	0	1
	0	0	1	0	0	0	1	0
	0	1	0	0	0	1	0	0
	1	0	0	0	1	0	0	0

NOTE

Example uses KDIAG of value 4 more typical is a value of 8. Consult the symbolic definition of 'KDIAG" in the program listing to be sure.

2. Disturb each row for > 3.2ms
3. Read check diagonal pattern
4. Do (1-3) KDIAG times moving the placement of the diagonal stripe to cover all address positions.
5. Do (1-4) for a complement pattern (zeros in a background of ones)

NOTE

This test is not normally executed except under APT or ACT. It may be inoked VIA Field Service Command 13 (Kakaze Mode).

6.2.2.20 Pattern 23 Shifting Diagonal Pattern Test -

Similar in overall operation to pattern 22 except it does not delay for refresh and disturb rows.

NOTE

This test is not normally executed except under APT or ACT. It may be invoked VIA Field Service Command 13 (Kamikaze Mode).

6.2.2.21 Pattern 24 Fast Galloping Pattern Test -

This does a classical galloping pattern except that addressing is incremented by 400 Octal (every 64th double word)

NOTE

This test is not normally executed except under APT or ACT. It may be invoked VIA Field Service Command 13 (Kamikaze Mode).

6.2.2.22 Pattern 25 Interrupt Enable Test -

1. Set CSR to Allow Uncorrectable Error Traps.
2. Access Test Double Words.
3. Check that no Uncorrectable Error Trap occurred.
4. Enable CSR for SBE Traps.
5. Access Test Double Words.
6. Check that no SBE Trap occurred.
7. Write a SBE in 1 Byte.
8. Disable CSR Traps.
9. Access Test Double Words.
10. Check that no Traps occurred.
11. Enable CSR for SBE Traps.
12. Access Test Double Words.
13. Check to Insure Trap Occurred.
14. Do (7-13) for the 3 other Bytes in the Double Word.
15. Create a DBE in 1 Byte.
16. Disable CSR Traps.
17. Access the Test Double Word.
18. Check that no Traps occurred.
19. Enable CSR for DBE Traps.
20. Access the Test Double Word.
21. Check to Insure Trap Occurred.
22. Enable CSR for SBE Traps.
23. Access the Test Double Word.
24. Check to Insure Trap Occurred.
25. Do (15-24) for the 3 other Bytes in the Double Word.

This insures that SBE's & DBE's give the correct type of traps.

NOTE

This test is run for ECC memory only.

6.2.2.23 Pattern 26 Random Data Test -

Write Random Data in a 16K Bank while incrementing the Addresses.

Read check Random Data.

This routine regenerates the same random numbers by using the same seed as the write sequence. After the read check the seed is updated so that the next use of this pattern will not invoke the same sequence of random numbers.

If you wish to change the random sequence so that it is different than any other run in the same configuration then there are 2 ways of doing so.

1. Modify symbolic locations ''SEEDHI'' and ''SEEDLO'' to any number you like.

2. Enter Field Service Mode and execute this pattern (command 5) on some (any good) bank for a short time (30 sec or so).

This can execute out of the User Data PAR's, the Kernel Data PAR's, and the Supervisor Data PAR's.

6.2.2.24 Pattern 27 Unique Bank Test -

This pattern uses Pattern 0 to write & read the Bank number in each bank.

It does not test Banks that require relocation to test.

It does not run as part of any script but rather is always run after normal pattern tests are complete.

6.2.2.25 Pattern 30 Flush Out DBE's Test -

This Reads each location then moves the old value back in. This is done with ECC Disabled and therefore corrects any DBE's or SBE's (if possible).

It does not run as part of any script but rather is always run just prior to the End of Pass Code, as part of a Control "C" (Boot) command, as part of End of Pass shutdown for ACT or XXDP Chain Mode, as part of hanging sequence after an error if under ACT or APT, and as part of a shutdown sequence directed by Switch 8 (Halt Program).

PAGE 69

6.2.2.26 Pattern 31 SOB-A-LONG Test -

Rationalization

In order to concentrate the memory cycles of a test into a particular address, we must cut the overhead cycles to a minimum. Frequently, the instruction itself may provide adequate data or set up a background in which any complemented bit may find it hard to survive.

The SOB instruction is the only PDP-11 instruction that is (1) a single operand, (2) can be repeatedly executed at the same PC and, (3) can escape this repetitious loop.

Hence, it can be possible to SOB a MOS cell to death (or at least brain wash him), and to SOB a core into over-heating (or at least warm discomfort).

The SOB Routine will be loaded and called with R0 set equal to the SOB constant "SOBK", R1 set equal to the complement of a "SOB R0,." Instruction "100776".

Simplified SOB Example:

```
1$:     SOB     R0,1$   ;SOB till R0 underflows
        MOV     R1,1$   ;Write complement of SOB
```

```
        CMP     R1,1$   ;Read  check not SOB
        BEQ        2$   ;Skip if OK
        SOBFAIL         ;Trap  report error
2$:     SOBMOV1         ;Code to get self moved
        SOBMOV2         ;forward 1 word and run again
        SOBMOV3
        SOBMOV4
        SOBMOV...
```

SEQ 0070

The value of the SOB constant can be found at symbolic location "SOBK" (typical 25 decimal).

This test is not in the normal script of execution but may be added via the APT E-TABLE, reference symbolic locations 'MKPAT', 'MJPAT', '$DDW2-5". Field Service Mode command 8 is the normal method of running this pattern.

NOTE

This test is not normally executed except under APT or ACT. It may be invoked VIA Field Service Command 13 (Kamikaze Mode).

6.2.2.27 Pattern 32 Write Recovery Test -

This test causes a WRITE, READ, WRITE, READ, ... to occur in memory and if the 1st, 3rd, 5th, ...READ is bad the program may bomb or if the 2nd, 4th, 6th, ... READ is bad the program will gracefully type out the error.

WRITE RECOVERY TEST
THIS TEST DIFFERS FROM OTHER TESTS IN THAT IT CONSISTS
OF A SMALL TEST PROGRAM ACTUALLY RUNNING IN THE BANK UNDER TEST.
THE PROGRAM IS SELF MODIFYING AND MAY BE DIFFICULT TO DEBUG.
TO AID IN THE DEBUG, REMEMBER THAT THE BANK AND MARGIN ARE BEING
DISPLAYED. THIS WILL ALLOW THE USER TO AT LEAST SEE WHICH MEMORY
BANK FAILED.
THE TEST CONSISTS OF 1/2 OF THE BANK STORED WITH 'MOV R2,-(PC)''
AND THE OTHER 1/2 CONTAINING ''177667''. ''177667'' IS THE COMPLEMENT
OF ''JMP (R0)'' INSTRUCTION. R2 CONTAINS ''COM -(R1)'' INSTRUCTION
ON ENTRY TO THE BANK AND R1 CONTAINS THE HIGHEST TEST ADDRESS IN
THAT BANK.
IF YOU UNDERSTAND THIS SO FAR THE REST IS EASY.
THE TEST EXECUTION IS AS FOLLOWS:
1. THE 'MOV R2,-(PC)'' INSTRUCTION EXECUTES STORING
THE CONTENTS OF R2 IN THE ADDRESS IT VACATED (DUE TO -(PC).
2. SINCE R2 CONTAINS A ''COM -(R1)'' INSTRUCTION IT COMPLEMENTS
THE HIGHEST ADDRESS UNDER TEST. THIS ADDRESS CONTAINED
''177667'' SO AFTER THE COM -(R1) IT EQUALS 110
CLEVERLY THIS IS THE ''JMP (R0)'' INSTRUCTION.
3. THIS SEQUENCE CONTINUES UNTIL THE 'MOV R2,-(PC)''
INSTRUCTIONS REACH THE MIDDLE OF THE TEST BANK.
THEN THE ''JMP (R0)'' INSTRUCTION IS MET
AND EXECUTED. R0 CONTAINED THE RETURN ADDRESS BACK
TO TEST 13.
4. THESE STEPS ARE REPEATED FOR EACH BANK UNDER TEST.

NOTE

This test is not normally executed except under APT or ACT. It may be invoked VIA Field Service Command 13 (Kamikaze Mode).

6.2.2.28 Pattern 33 Branch Gobble Test -

This test loads a small routine into the memory under test. The routine moves itself along in memory one word after each pass so that when it reaches the end every instruction has executed from every location with the exception of the beginning and end of each test area.

The Branch Gobble's general format after you eliminate setup code and code to move the program along is as follows.

```
BGTEST: 0                       ;TEST WORD
BRGOBB: SEC
        ADCB    BGTEST          ;INC LOW BYTE
        BMI     1$              ;END LOOP AFTER 128 TIMES
        INCB    BGTEST+1        ;INC HIGH BYTE
        BR      BRGOBB          ;LOOP 128 TIMES
^$:     BVS     2$              ;BRANCH IF V-BIT SET (SHOULD BE)
        ERROR                   ;ERROR TRAP
2$:     CLV                     ;CLEAR V-BIT
        INCB    BGTEST          ;INC HIGH BYTE ONE LAST TIME
        BCS     3$              ;BRANCH IF C-BIT SET (SHOULD NOT BE)
        BVC     3$              ;BRANCH IF V-BIT CLEAR (SHOULD NOT BE)
        BMI     4$              ;BRANCH IF N-BIT SET (SHOULD BE)
3$:     ERROR                   ;ERROR TRAP
4$:     RETURN
```

This code origionally came from the PDP-11 Family Instruction Exerciser DZQKA-A. The first MOS memorys fell succeptable to this section of that diagnostic and it has been an important memory exerciser ever since.

NOTE

This test is not normally executed except under APT or ACT. It may be invoked VIA Field Service Command 13 (kamikaze Mode).

6.2.2.29 Pattern 34 Soft Error Test -

Rationalization

MOS chips have a failure mode in which they can randomly pick or drop bits. This is caused by Alpha particles bombarding tne cell. If the cell is very small (and they are) then the electrons displaced by the Alpha particle are sufficient to cause the cell to change from a one to a zero or from a zero to a one.

This test is controlled by the main program so that it is used to create a pattern of 125252 and 52525 on alternate passes of the program. The configuration table is used to flag banks that have the pattern invalidated because another pattern was written over this background.

This pattern is nothing more than a clever use of pattern 0.

6.2.2.30 Pattern 35 Worst Case Parity Test -

1. Force Write Wrong Parity in each 1K word block of the Memory Under Test.
2. Read with Parity Trapping enabled, making sure that a trap occurrs.
3. Make sure error address bits are set correctly.
4. Write good parity without trapping, and make sure no trap occurrs when read.

NOTE

This test is run for parity memory which is not controlled by the same CSR as the program.

6.2.2.31 Pattern 999 Null Test -

This is an instant return added to preserve the software structure.

This pattern replaces any real patterns when the APT E-Table does not specify a pattern to be run.

7.0 Program Features

7.1 Fast Data Access Rates

One of the main areas of concern in testing memory in systems environments is speed. One of the prime reasons that system programs like RSTS,IAS and MUMPS can crash due to memory failures not detectable by memory diagnostics (0-124K,0-2 MEG,etc.) is because of multiple NPR devices contending for the bus. After some delay a NPR device becomes bus master and does several memory transfers at memory data rates.

On the other hand most diagnostics when writing reading and/or checking patterns spend most of their time fetching instructions and operands out of their program space and proportionally little time accessing the memory under test.

This diagnostic's error detecting abilities have been optimized around the primary design criteria of speed. To this end the following steps have been taken.

7.1.1 Fast City -

Utilization of Memory Management Registers as Non Memory Bus, Non UNIBUS, Bipolar Memory. Since User Mode is only used for relocation and Data Space is never used, then subroutines can be executed from the UIPAR's, UDPAR's, KDPAR's, SDPAR's and with some Bit Pattern restrictions the UIPDR's, UDPDR's, KDPDR's, and SDPDR's.

The program runs in Kernel mode and Patterns are executed in Supervisor mode for mapping purposes. All core patterns and some MOS Patterns are subroutines that are moved to this Bipolar region referred to in the program as Fast City.

NOTE

18-Bit PDP-11's cannot execute from the PAR's because their PAR's are only 12 bits wide; they also have no Supervisor Mode. Therefore, all patterns are executed in memory, using User Mode (reference Section 7.5).

7.1.2 SOB's -

Utilization of the full PDP-11 Instruction Set to speed pattern algorithms (principally the SOB).

7.1.3 CACHE -

CACHE is used between pattern tests to decrease program pass times. CACHE can be defeated by the operator (reference section 2.3.3.1).

7.2 Bank Zero Testing

Bank Zero has been traditionally neglected by memory diagnostics for the following reason.

The vector space exists there and ALL traps must not access test pattern data. If the area is tested the diagnostic must not use any traps, and it is against the rules for power to fail.

Systems with Memory Management can overcome this because all traps are to Kernel Virtual space even if the power should fail (caution must be observed because power up goes to physical address 24 (because the Memory Management Unit comes up off)).

However, Catch 22 is that the diagnostic is not APT compatible in this mode because APT Accesses Physical Memory Locations.

The PDP-11/44 can over come this because the UNIBUS Map can fool APT.

Because of the previous arguments this program does not relocate in the true since of the word (i.e. no position independent code was written (at least not on purpose)), but rather this program moves and remaps (hereafter referred to as relocates). This enables the complete testing of Bank Zero or any other program space or privileged space exactly as all other banks are tested. (The conditional test to see if a bank is protected is complemented when relocated).

NOTE

The program will relocate only in the first pass under APT; after this, the program will remain fixed in Banks 0 and 1.

7.3 Memory Configuration Map

This map is printed out immediately after sizing the memory unless SW6 is set (reference section 2.3.1). It can also be printed at any later time in Field Service Mode (reference section 2.3.4.3.7)

Example:

```
                        MEMORY CONFIGURATION MAP
                               16K BANKS
                1         2         3         4         5         6         7
        01234567012345670123456701234567012345670123456701234567012345670123
ERRORS            XX
CPU MAP 11111111111111111111111111111111
INTRLV  ----------------3333333333333333
MEMTYPE LLLLLLLLMMMMMMMMMMMMMMMMMMMMMMMM
CSR     00000000111111112222222222222222
PROTECT PP      I       I       P
            0         1         2         3         4         5         6
        45670123456701234567012345670123456701234567012345670123456701234567
ERRORS
CPU MAP
INTRLV
MEMTYPE
CSR
PROTECT
```

Displayed are Banks 0-167 Octal (2 meg words). If the Fat Terminal Switch was set (reference section 2.3.1) then all Banks would fit on one line. If this was an 18-Bit PDP-11 (eg - 11/34), only Banks 0-7 would be printed.

The sizing routine could not write zeros and ones in Banks 10 & 11, hence they are marked as bad with X's.

The CPU was able to access banks 0-37 (512 K words).

There is interleaving on Banks 20-37, with CSR 172004 controlling the Address Bit 1 Non-Asserted addresses, and CSR 172006 controlling the Address Bit 1 Asserted addresses.

Banks 0-7 are Memory Type L (MS-11L), and Banks 10-37 are Memory Type M (MS11-M); while Banks 40-167 do not exist. Memory Type K would indicate MF11S-K, Memory Type P would indicate UNIBUS Parity, and Memory Type B would indicate 11/45-type Bipolar memory.

Banks 0-7 are assigned to CSR 172100, 10-17 to CSR 172102, and 20-37 to CSR's 172104 and 172106. CSR locations should be assigned in order as shown, however the section of memory containing the program (i.e. thelowest 16K chunk, addresses 0 to 77776) must be assigned the lowest CSR to be used (as in the above example) for the sizing routine to function properly.

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

Banks 0 and 1 are protected because they are program space. Bank 0 and 1 can also be protected because they are in the bottom 16K of an MS11-M CSR. The protection is hierarchical and program space overshadows MS11-M protection. Banks 0 and 1 will not be tested until the program relocates. If any bank is protected by MS11-M (or MF11S-K) and not because it is in program space it will have an "I" typed in this row. This is to point out where the protected banks start for each ECC CSR. Note the "P" at Bank 30; This points out the "Shadow" protection which occurrs when two MS11-M memories are interleaved. Therefore, Bank 30 will not be tested until the program has relocated.

7.4 Everything You've Always Wanted To Know About SUPERMAC ...

SUPER-MAC is a set of structured programming macros that allows programs to be written in a high level, easily understood language.

As a general rule, most SUPER-MAC statements can be single-line statements or multiple-line (nested) block statements. A single-line statemen+ must be completed on one source line; no continuation lines are allowed. Single-line statements should be as short and simple as possible. Comments may also be included on a source line. All the general rules, conditions, etc., that govern MACRO-11 also govern SUPER-MAC. Spacing on a source lne is very important. The elements should be separated by a comma or a space. Tabs should never be used for spacing. For example: The expression A+B is interpreted different than A + B.

All the conditional statements can be written as multiple-line nested blocks. Each level of nesting within a block must be terminated with an associated END statement. Each level of nesting should be indented two spaces.

User written macros or assembly language instructions may be included in a program if desired. As a debugging aid, if the symbol LST$$ is defined, it will cause generated code and labels to be listed. All programs must begin with the macro call SMACIT. This call initializes SUPER-MAC. All legal PDP-11 source and destination operands are legal in SUPER-MAC.

7.4.1 Sample Source File -

```
        .ENABL ABS
        .ENABL AMA
        .MCALL .SUPER
        .SUPER
        ;LST$$=0
        BITS=40
A:      0
B:      0
C:      0
D:      0
E:      0
F:      0
G:      0
H:      0
I:      0
J:      0
        .PAGE
;LET EXAMPLES
        LET R0 := A
        LET B :  C + D
        LET E :  F +  1
        LET G :- H +  2
        LET J := J +  01
        LET A :B= B
;IF EXAMPLES
        IF A IS TRUE
          MOV    23,D
        END ;OF IF A
        IF B IS FALSE
          MOV    34,E
        END ;OF IF B
        IF A EQ B THEN LET C := D
        IF A LT B
          MOV   C,D
        ELSE
          MOV   E,D
        END ;OF IF A
        IF A EQ B AND C NE D
          MOV   F,G
        END ;OF IF A
        IF A EQ B OR C NE D
          MOV   F,G
        END ;OF IF A
        IFB A EQ B AND C EQ  1
          MOV   H,
        ELSE
          MOV   E,J
        END ;OF IFB A
        IFB A EQ B ANDB C EQ  1
          MOV   H,J
        ELSE
          MOV   E,J
        END ;OF IFB A
        IF RESULT IS EQ
```

```
  MOV   A,B
END ;OF IF RESULT
IF  BIT5 SET.IN A
  MOV   B,C
```

SEQ 0080

```
        END ;OF IF  BIT5
        IF  BIT5 OFF.IN A
          MOV   C,D
        END ;OF IF  BIT5
;ON.ERROR IS LIKE AN IF STATEMENT ON THE C-BIT
;ON.ERROR EXAMPLES
        ON.ERROR
          MOV   A,B
        ELSE
          MOV   C,B
        END ;OF ON.ERROR
        ON.NOERROR
          MOV   C,B
        ELSE
          MOV   A,B
        END ;OF ON.NOERROR
        ON.ERROR THEN LET A :B- B
,FOR EXAMPLES
        FOR I ·=  -5 TO  23
          INC   A
        END ;OF FOR I
        FOR R0 :=  0 TO  140 BY  4
          DEC   A(R0)
        END ;OF FOR R0
        FOR I :=  133 DOWNTO  3 BY  2
          ADD   A,B
        END ;OF FOR I
;BEGIN EXAMPLES
        BEGIN ALPHA
          FOR R0 :-  0 TO  167
            MOVB       A(R0),B
            IF B LT  0 THEN LEAVE ALPHA
          END ;OF FOR R0
          FOR R0 :=  400 TO  567
            IF B GE  0 THEN LEAVE ALPHA
          END ;OF FOR R0
        END ALPHA
;$RETURN EXAMPLES
        $RETURN
        $RETURN ERROR
        $RETURN NOERROR
;CASE EXAMPLES
        MOV     A,R0
        CASE R0
          A
          B
          C
          D
          E
          F
        END ;OF CASE R0

        .END
```

7.4.2 Sample Listing File (with no expanded macros) - -

```
.MAIN.  MACRO M1111  01-APR-79 16:41  PAGE 2


      1 000000                              .ENABL ABS
      2                                     .ENABL AMA
      3                                     .MCALL .SUPER
      4 000000                              .SUPER
      5                                     ;LST$$=0
      6          000040                     BIT5=40
      7 000000   000000              A:     0
      8 000002   000000              B:     0
      9 000004   000000              C:     0
     10 000006   000000              D:     0
     11 000010   000000              E:     0
     12 000012   000000              F:     0
     13 000014   000000              G:     0
     14 000016   000000              H:     0
     15 000020   000000              I:     0
     16 000022   000000              J:     0
```

```
.MAIN.  MACRO M1111  01-APR-79 16:41  PAGE 3


     18                                 ;LET EXAMPLES
     19 000024                                  LET R0 := A
     20 000030                                  LET B := C + D
     21 000044                                  LET E := F +  1
     22 000056                                  LET G := H +  2
     23 000072                                  LET J := J +  01
     24 000100                                  LET A :B= B
     25                                 ;IF EXAMPLES
     26 000106                                  IF A IS TRUE
     27 000114  012737  000023  000006            MOV    23,D
     28 000122                                  END ;OF IF A
     29 000122                                  IF B IS FALSE
     30 000130  012737  000034  000010            MOV    34,E
     31 000136                                  END ;OF IF B
     32 000136                                  IF A EQ B THEN LET C :- D
     33 000154                                  IF A LT B
     34 000164  013737  000004  000006            MOV   C,D
     35 000172                                  ELSE
     36 000174  013737  000010  000006            MOV   E,D
     37 000202                                  END ;OF IF A
     38 000202                                  IF A EQ B AND C NE D
     39 000222  013737  000012  000014            MOV   F,G
     40 000230                                  END ;OF IF A
     41 000230                                  IF A EQ B OR C NE D
     42 000250  013737  000012  000014            MOV   F,G
     43 000256                                  END ;OF IF A
     44 000256                                  IFB A EQ B AND C EQ  1
     45 000276  013737  000016  000022            MOV   H,J
     46 000304                                  ELSE
     47 000306  013737  000010  000022            MOV   E,J
     48 000314                                  END ;OF IFB A
     49 000314                                  IFB A EQ B ANDB C EQ  1
     50 000334  013737  000016  000022            MOV   H,J
     51 000342                                  ELSE
     52 000344  013737  000010  000022            MOV   E,J
     53 000352                                  END ;OF IFB A
     54 000352                                  IF RESULT IS EQ
     55 000354  013737  000000  000002            MOV   A,B
     56 000362                                  END ;OF IF RESULT
     57 000362                                  IF  BIT5 SET.IN A
     58 000372  013737  000002  000004            MOV   B,C
     59 000400                                  END ;OF IF  BIT5
     60 000400                                  IF  BIT5 OFF.IN A
     61 000410  013737  000004  000006            MOV   C,D
     62 000416                                  END ;OF IF  BIT5
     63                                 ;ON.ERROR IS LIKE AN IF STATEMENT ON THE C-BIT
     64                                 ;ON.ERROR EXAMPLES
     65 000416                                  ON.ERROR
     66 000420  013737  000000  000002            MOV   A,B
     67 000426                                  ELSE
     68 000430  013737  000004  000002            MOV   C,B
     69 000436                                  END ;OF ON.ERROR
```

```
70 000436                                    ON.NOERROR
71 000440  013737  000004  000002              MOV    C,B                                  SEQ 0084
72 000446                                    ELSE
73 000450  013737  000000  000002              MOV    A,B
```

```
74 000456                    END ;OF ON.NOERROR
```

```
.MAIN.  MACRO M1111  01-APR-79 16:41  PAGE 3-1


     75 000456                                  ON.ERROR THEN LET A :B= B
     76                                    ;FOR EXAMPLES
     77 000466                                  FOR I :=  -5 TO  23
     78 000474  005237  000000                    INC   A
     79 000500                                  END ;OF FOR I
     80 000514                                  FOR R0 :=  0 TO  140 BY  4
     81 000516  005360  000000                    DEC   A(R0)
     82 000522                                  END ;OF FOR R0
     83 000534                                  FOR I :=  133 DOWNTO  3 BY  2
     84 000542  063737  000000  000002            ADD   A,B
     85 000550                                  END ;OF FOR I
     86                                    ;BEGIN EXAMPLES
     87 000566                                  BEGIN ALPHA
     88 000566                                    FOR R0 :=  0 TO  167
     89 000570  116037  000000  000002              MOVB        A(R0),B
     90 000576                                    IF B LT  0 THEN LEAVE ALPHA
     91 000604                                    END ;OF FOR R0
     92 000614                                    FOR R0 :=  400 TO  567
     93 000620                                    IF B GE  0 THEN LEAVE ALPHA
     94 000626                                    END ;OF FOR R0
     95 000636                                  END ALPHA
     96                                    ;$RETURN EXAMPLES
     97 000636                                  $RETURN
     98 000640                                  $RETURN ERROR
     99 000644                                  $RETURN NOERROR
    100                                    ;CASE EXAMPLES
    101 000650  013700  000000                  MOV     A,R0
    102 000654                                  CASE R0
    103 000664  000000                            A
    104 000666  000002                            B
    105 000670  000004                            C
    106 000672  000006                            D
    107 000674  000010                            E
    108 000676  000012                            F
    109 000700                                  END ;OF CASE R0
    110
    111         000001                          .END
```

7.4.3 Sample Listing File (with expanded macros) - -

```
.MAIN.  MACRO M1111  01-APR-79 16:10  PAGE 2


      1 000000                              .ENABL ABS
      2                                     .ENABL AMA
      3                                     .MCALL .SUPER
      4 000000                              .SUPER
      5           000000                    LST$$=0
      6           000040                    BIT5=40
      7 000000    000000              A:    0
      8 000002    000000              B:    0
      9 000004    000000              C:    0
     10 000006    000000              D:    0
     11 000010    000000              E:    0
     12 000012    000000              F:    0
     13 000014    000000              G:    0
     14 000016    000000              H:    0
     15 000020    000000              I:    0
     16 000022    000000              J:    0
```

.MAIN. MACRO M1111 01-APR-79 16:10 PAGE 3

```
                                        ;LET EXAMPLES
000024                                          LET R0 :- A
000024  013700  000000                          MOV A,R0
000030                                          LET B := C + D
000030  013737  000004  000002                  MOV C,B
000036  063737  000006  000002                  ADD D,B
000044                                          LET E := F +  1
000044  013737  000012  000010                  MOV F,E
000052  005237  000010                          INC E
000056                                          LET G := H +  2
000056  013737  000016  000014                  MOV H,G
000064  062737  000002  000014                  ADD  2,G
000072                                          LET J := J +  01
000072  062737  000001  000022                  ADD  01,J
00010'                                          LET A :B= B
000100  113737  000002  000000                  MOVB B,A
                                        ;IF EXAMPLES
000106                                          IF A IS TRUE
000106  005737  000000                          TST A
000112  001403                                  BEQ L0
000114  012737  000023  000006                    MOV    23,D
000122                                          END ;OF IF A
000122                                  L0:
000122                                          IF B IS FALSE
000122  005737  000002                          TST B
000126  001003                                  BNE L1
000130  012737  000034  000010                    MOV    34,E
000136                                          END ;OF IF B
000136                                  L1:
000136                                          IF A EQ B THEN LET C := D
000136  023737  000000  000002                  CMP A,B
000144  001003                                  BNE L2
000146  013737  000006  000004                  MOV D,C
000154                                  L2:
000154                                          IF A LT B
000154  023737  000000  000002                  CMP A,B
000162  002004                                  BGE L3
000164  013737  000004  000006                    MOV   C,D
000172                                          ELSE
000172  000403                                  BR L4
000174                                  L3:
000174  013737  000010  000006                    MOV   E,D
000202                                          END ;OF IF A
000202                                  L4:
000202                                          IF A EQ B AND C NE D
000202  023737  000000  000002                  CMP A,B
000210  001007                                  BNE L5
000212  023737  000004  000006                  CMP C,D
000220  001403                                  BEQ L5
000222  013737  000012  000014                    MOV   F,G
000230                                          END ;OF IF A
000230                                  L5:
```

```
41 000230                                          IF A EQ B OR C NE D                                    SEQ 0089
   000230  023737  000000  000002                  CMP A,B
   000236  001404                                  BEQ L6
   000240  023737  000004  000006                  CMP C,D
```

```
000246  001403                  BEQ L7
```

```
.MAIN.  MACRO M1111  01-APR-79 16:10  PAGE 3-1


     000250                                     L6:
  42 000250  013737  000012  000014                    MOV   F,G
  43 000256                                     END ;OF IF A
     000256                                     L7:
  44 000256                                     IFB A EQ B AND C EQ  1
     000256  123737  000000  000002                    CMPB A,B
     000264  001010                                    BNE L10
     000266  023727  000004  000001                    CMP C, 1
     000274  001004                                    BNE L10
  45 000276  013737  000016  000022                    MOV   H,J
  46 000304                                     ELSE
     000304  000403                                    BR L11
     000306                                     L10:
  47 000306  013737  000010  000022                    MOV   E,J
  48 000314                                     END ;OF IFB A
     000314                                     L11:
  49 000314                                     IFB A EQ B ANDB C EQ  1
     000314  123737  000000  000002                    CMPB A,B
     000322  001010                                    BNE L12
     000324  123727  000004  000001                    CMPB C, 1
     000332  001004                                    BNE L12
  50 000334  013737  000016  000022                    MOV   H,J
  51 000342                                     ELSE
     000342  000403                                    BR L13
     000344                                     L12:
  52 000344  013737  000010  000022                    MOV   E,J
  53 000352                                     END ;OF IFB A
     000352                                     L13:
  54 000352                                     IF RESULT IS EQ
     000352  001003                                    BNE L14
  55 000354  013737  000000  000002                    MOV   A,B
  56 000362                                     END ;OF IF RESULT
     000362                                     L14:
  57 000362                                     IF  BIT5 SET.IN A
     000362  032737  000040  000000                    BIT  BIT5,A
     000370  001403                                    BEQ L15
  58 000372  013737  000002  000004                    MOV   B,C
  59 000400                                     END ;OF IF  BIT5
     000400                                     L15:
  60 000400                                     IF  BIT5 OFF.IN A
     000400  032737  000040  000000                    BIT  BIT5,A
     000406  001003                                    BNE L16
  61 000410  013737  000004  000006                    MOV   C,D
  62 000416                                     END ;OF IF  BIT5
     000416                                     L16:
  63                                            ;ON.ERROR IS LIKE AN IF STATEMENT ON THE C-BIT
  64                                            ;ON.ERROR EXAMPLES
  65 000416                                             ON.ERROR
     000416  103004                                     BCC L17
  66 000420  013737  000000  000002                    MOV   A,B
  67 000426                                     ELSE
     000426  000403                                    BR L20
```

```
    000430                          L17:
68  000430  013737  000004  000002          MOV   C,B                          SEQ 0092
69  000436                        END ;OF ON.ERROR
    000436                          L20:
```

```
70 000436                    ON.NOERROR
```

```
.MAIN.  MACRO M1111  01-APR-79 16:10  PAGE 3-2


     000436  103404                          BCS L21
  71 000440  013737  000004  000002            MOV   C,B
  72 000446                                  ELSE
     000446  000403                          BR L22
     000450                              L21:
  73 000450  013737  000000  000002            MOV   A,B
  74 000456                                  END ;OF ON.NOERROR
     000456                              L22:
  75 000456                                  ON.ERROR THEN LET A :B= B
     000456  103003                          BCC L23
     000460  113737  000002  000000          MOVB B,A
     000466                              L23:
  76                                     ;FOR EXAMPLES
  77 000466                                  FOR I :=  -5 TO  23
     000466  012737  177773  000020          MOV  -5,I
     000474                              B0:
  78 000474  005237  000000                    INC   A
  79 000500                                  END ;OF FOR I
     000500  005237  000020                  INC I
     000504  023727  000020  000023          CMP I, 23
     000512  003770                          BLE B0
     000514                              E0:
  80 000514                                  FOR R0 :=  0 TO  140 BY  4
     000514  005000                          CLR R0
     000516                              B1:
  81 000516  005360  000000                    DEC   A(R0)
  82 000522                                  END ;OF FOR R0
     000522  062700  000004                  ADD  4,R0
     000526  020027  000140                  CMP R0, 140
     000532  003771                          BLE B1
     000534                              E1:
  83 000534                                  FOR I :-  133 DOWNTO  3 BY  2
     000534  012737  000133  000020          MOV  133,I
     000542                              B2:
  84 000542  063737  000000  000002            ADD   A,B
  85 000550                                  END ;OF FOR I
     000550  162737  000002  000020          SUB  2,I
     000556  023727  000020  000003          CMP I, 3
     000564  002366                          BGE B2
     000566                              E2:
  86                                     ;BEGIN EXAMPLES
  87 000566                                  BEGIN ALPHA
     000566                              B3:
  88 000566                                    FOR R0 :=  0 TO  167
     000566  005000                          CLR R0
     000570                              B4:
  89 000570  116037  000000  000002            MOVB        A(R0),B
  90 000576                                    IF B LT  0 THEN LEAVE ALPHA
     000576  005737  000002                  TST B
     000602  002415                          BLT E3
  91 000604                                    END ;OF FOR R0
     000604  005200                          INC R0
```

```
    000606  020027  000167                  CMP R0, 167                                    SEQ 0095
    000612  003766                          BLE B4
    000614                          E4:
 92 000614                                    FOR R0 :=  400 TO  567
```

```
000614  012700  000400                MOV  400,R0
```

```
.MAIN.  MACRO M1111  01-APR-79 16:10  PAGE 3-3


      000620                                    B5:
   93 000620                                            IF B GE  0 THEN LEAVE ALPHA
      000620  005737  000002                          TST B
      000624  002004                            BGE E3
   94 000626                                      END ;OF FOR R0
      000626  005200                            INC R0
      000630  020027  000567                    CMP R0, 567
      000634  003771                            BLE B5
      000636                                    E5:
   95 000636                                    END ALPHA
      000636                                    E3:
   96                                           ;$RETURN EXAMPLES
   97 000636                                    $RETURN
      000636  000207                            RTS PC
   98 000640                                    $RETURN ERROR
      000640  000261                            SEC
      000642  000207                            RTS PC
   99 000644                                    $RETURN NOERROR
      000644  000241                            CLC
      000646  000207                            RTS PC
  100                                           ;CASE EXAMPLES
  101 000650  013700  000000                    MOV     A,R0
  102 000654                                    CASE R0
      000654  010046                            MOV R0,-(SP)
      000656  006316                            ASL @SP
      000660  004737  000700                    JSR PC,L24
  103 000664  000000                              A
  104 000666  000002                              B
  105 000670  000004                              C
  106 000672  000006                              D
  107 000674  000010                              E
  108 000676  000012                              F
  109 000700                                    END ;OF CASE R0
      000700                                    L24:
      000700  062616                            ADD (SP)+,@SP
      000702  013646                            MOV @(SP)+,-(SP)
      000704  004736                            JSR PC,@(SP)+
  110
  111         000001                            .END
```

7.5 Memory Management Mapping

7.5.1 Memory Management Mapping for the 11/44 -

PAR	SUPERVISOR	KERNEL	USER
0	Program	Program	Dst Bk/Fst Mem
1	Program	Program	Src Bk/Fst Mem
2	Program	Program	Src Bk/Fst Mem
3	Test Area	Program	Src Bk/Fst Mem
4	Test Area	Program	Dst Bk/Fst Mem
5	Test Area	Program	Dst Bk/Fst Mem
6	Test Area	Map to CSR's	Dst Bk/Fst Mem
7	Perif Page	Perif Page	Dst Bk/Fst Mem

7.5.2 Memory Management Mapping for UNIBUS-11's with Supervisor Mode (eg 11/45) -

PAR	SUPERVISOR	KERNEL	USER
0	Program	Program	Dst Bk
1	Program	Program	Src Bk
2	Program	Program	Src Bk
3	Test Area	Program	Src Bk
4	Test Area	Program	Dst Bk
5	Test Area	Program	Dst Bk
6	Test Area	Map to CSR's	Dst Bk
7	Perif Page	Perif Page	Dst Bk

7.5.3 Memory Management Mapping for UNIBUS-11's without Supervisor Mode (eg 11/34) -

PAR	KERNEL	USER
0	Program	Program/Dst Bk
1	Program	Program/Src Bk
2	Program	Program/Src Bk
3	Program	Test Area/Src Bk
4	Program	Test Area/Dst Bk
5	Program	Test Area/Dst Bk
6	Map to CSR's	Test Area/Dst Bk
7	Perif Page	Perif Page/Dst Bk

7.6 Listing Format

The program listing is in two parts. The first part is a complete listing the way it was used by the author. Macros are not expanded and an effort was made to keep each section on one or two pages. The second part is a complete listing with all macros expanded to show all generated code in all program locations. This last listing is for those who have a need to look at the actual assembly language code and the generated binary.

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

```
                        .TITLE  CZMSDA0 MS11-L/M DIAGNOSTIC
                        .REM    &


                                        IDENTIFICATION
                                        --------------

                          PRODUCT CODE:

                          PRODUCT NAME:         CZMSDA0 MS11-L/M MEMORY DIAGNOSTIC

                          PRODUCT DATE:         DECEMBER 1979

                          MAINTAINER:           DIAGNOSTIC ENGINEERING

                          THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE
                          WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT
                          BY DIGITAL EQUIPMENT CORPORATION.  DIGITAL EQUIPMENT
                          CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT
                          MAY APPEAR IN THIS MANUAL.

                          THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE
                          PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER
                          SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S
                          COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY
                          OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

                          DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR
                          THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS
                          NOT SUPPLIED BY DIGITAL.

                          COPYRIGHT (C)  1979 DIGITAL EQUIPMENT CORPORATION
                        &
```

```
                        ;                       REVISION HISTORY
                        ;                       ---------- --
                        ;
                        ;  REVISION   VERSION     DATE            AUTHOR            CHANGES
                        ;  ========   =======   ========     ================     =======
                        ;   CZMSDA     V00.00    1-DEC-79    MICHAEL D. BIBEAULT  NONE-NEW PROGRAM
                        ;
```

```
                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        .SBTTL ;SWITCH REGISTER DEFINITIONS
                        .SBTTL ;*
                        .SBTTL ;*       SWITCH              USE
                        .SBTTL ;*       ------              ------------------
                        .SBTTL ;*         15                HALT ON ERROR
                        .SBTTL ;*         14                LOOP ON TEST
                        .SBTTL ;*         13                INHIBIT ERROR TYPEOUTS
                        .SBTTL ;*         12                INHIBIT RELOCATION
                        .SBTTL ;*         11                QUICK VERIFY
                        .SBTTL ;*         10                BELL ON ERROR
                        .SBTTL ;*          9                LOOP ON ERROR
                        .SBTTL ;*          8                HALT PROGRAM (UNRELOCATED & RESTORE LOADERS)
                        .SBTTL ;*          7                DETAILED ERROR REPORTS
                        .SBTTL ;*          6                INHIBIT CONFIGURATION MAP
                        .SBTTL ;*          5                LIMIT MAX ERRORS PER BANK
                        .SBTTL ;*          4                FAT TERMINAL (132 COLUMNS OR BETTER)
                        .SBTTL ;*          3                TEST MODE - SEE DOCUMENT
                        .SBTTL ;*          2                TEST MODE - SEE DOCUMENT
                        .SBTTL ;*          1                TEST MODE - SEE DOCUMENT
                        .SBTTL ;*          0                DETECT SINGLE BIT ERRORS
```

```
     72 000000                          .ENABL  ABS
     73                                 .ENABL  AMA
     74                                 .DSABL  GBL
     75                                 ;NOTE:  CZMSDA.SML IS THE SUPER.MAC SOURCE AND IS RELEASED WITH
     76                                 ;THIS PROGRAM.  ALL THESE .MCALL STATEMENTS REFERENCE THAT FILE.
     77                                 .MCALL  SMACIT,..PUSH,..POP,..TAG,..BRAN,.EMIT,.EMITN,.EMITL,.EMITR
     78                                 .MCALL  .IFOPR,.IS,.GENBR,.OPADD,.OPSUB,CLEAR,SET,CLEARB,SETB
     79                                 .MCALL  RNE,REQ,RLT,RGE,RGT,RLE,RPL,RMI,RHI,RLOS,RHIS,RLO,RCS,RCC
     80                                 .MCALL  IF,.OR,.IFARI,.LEAVE,.GOTO,OR,AND,THEN,ELSE,WHILE,CASE
     81                                 .MCALL  FOR,TO,DOWNTO,REPEAT,UNTIL,THRU,END,BEGIN
     82                                 .MCALL  $$END,LEAVE,JUMPTO,GOTO,PUSH,POP,LET
     83                                 .MCALL  .SIMPLE,.ARITH,ORB,ANDB,IFB,UNTILB,WHILEB,ON.ERROR,ON.NOERROR
     84                                 .MCALL  $CALL,$RETURN
     85
     86                                 .NLIST  TTM                     ;I WANT FAT PAPER!
     87                                 .LIST   MC,SYM                  ;LIST MACRO CALLS, SYMBOL TABLE
     88                                 .NLIST  MD,CND,ME               ;DON'T LIST MACRO DEFS & CONDITIONALS & EXPANSIONS
     89                                 ;LST$$= 0                       ;DEFINED TO LIST SUPERMAC EXPANSIONS
     90          163000                 $SWR=   163000                  ;USE THESE SYSMAC SWITCHES
     91          000001                 $TN=    1                       ;FIRST TEST NUMBER TO ONE(1)
     92 000000                          SMACIT
```

```
                                .SBTTL  DEFINE  TRAPS
                        ;ALL ENTRIES HERE MUST HAVE A CORRESPONDING ENTRY IN THE
                        ;TRAP TABLE "$TRPAD" (NEAR END OF PROGRAM).
                        ;*TRAP DEFINITIONS
                        ;
                        ;HERE IS HOW TRAPS WORK IN THIS PROGRAM
                        ;
                        ;ALL TRAPS EXECUTE A "TRAP" INSTRUCTION WHICH TAKES THE PROGRAM
                        ;TO SYMBOLIC LOCATION "$TRAP"
                        ;
                        ;AT $TRAP THE PROGRAM PICKS UP THE RIGHT BYTE OF THE TRAP INSTRUCTION
                        ;AND INDEXES INTO A TABLE AT LOCATION "$TRPAD" WHICH SENDS THE PROGRAM TO
                        ;THE SPECIFIC ROUTINE TO HANDLE THAT SPECIFIC TRAPS TASK.
                        ;
                        ;THE ULTIMATE DESTINATION OF A TRAP INSTRUCTION CAN BE GUESSED AT AS FOLLOWS
                        ;
                        ;EXAMPLE:        NOP
                        ;                NOP
                        ;                NOP
                        ;                KERNEL                  ;ENTER KERNEL MODE
                        ;                NOP
                        ;
                        ;       ADD A DOLLAR SIGN TO THE SYMBOLIC NAME AND CHECK THE CRF FOR SOMETHING CLOSE
                        ;       IN THIS CASE THE CRF HAS $KERNE LISTED AS 032546
                        ;       AT LOCATION 32546 YOU FIND THE ROUTINE $KERNEL
                        ;
                        ;       NOTE THAT CRF SYMBOLS ARE TRUCNATED TO 6 CHARACTERS
                        ;       SYMBOLIC NAMES GREATER THAT 6 CHARACTERS ARE USED SO I CAN
                        ;       REMEMBER WHAT THEY MEAN!
                        ;
                        ;       ALL GOOD TRAP ROUTINES RETURN VIA AN "RTI" INSTRUCTION
104401                  TYPEIT= 104401          ;;TTY TYPEOUT ROUTINE
104402                  TYPOC=  104402          ;;TYPE OCTAL NUMBER (WITH LEADING ZEROS)
104403                  TYPOS=  104403          ;;TYPE OCTAL NUMBER (NO LEADING ZEROS)
                        ;TYPON= 104404          ;;TYPE OCTAL NUMBER (AS PER LAST CALL)
104405                  TYPDS=  104405          ;;TYPE DECIMAL NUMBER (WITH SIGN)
                        ;TYPBN= 104406          ;;TYPE BINARY (ASCII) NUMBER

104407                  GTSWR=  104407          ;;GET SOFT-SWR SETTING
104410                  CKSWR=  104410          ;;TEST FOR CHANGE IN SOFT-SWR

104411                  RDCHR=  104411          ;;TTY TYPEIN CHARACTER ROUTINE
104412                  RDLIN=  104412          ;;TTY TYPEIN STRING ROUTINE
104413                  RDOCT=  104413          ;;READ AN OCTAL NUMBER FROM TTY
104414                  RDDEC=  104414          ;;READ A DECIMAL NUMBER FROM TTY

104415                  SAVREG= 104415          ;;SAVE R0-R5 ROUTINE
104416                  RESREG= 104416          ;;RESTORE R0-R5 ROUTINE

104417                  KERNEL= 104417          ;ENTER KERNEL MODE

104420                  ENERGIZE=104420         ;TURN ON MEMORY MANAGEMENT & TRAPS
104421                  DEENERGIZE=104421       ;TURN OFF MEMORY MANAGEMENT & TRAPS
104422                  KMAP=   104422          ;MAP KERNEL 1 TO 1

104423                  CACHON= 104423          ;TURN ON CACHE
104424                  CACHOFF=104424          ;TURN OFF CACHE
```

```
152
153          104425          LOADCSR=104425          ;LOAD CORRECT CSR
154          104426          READCSR=104426          ;READ CORRECT CSR
155
156          104427          PERR01= 104427          ;PROGRAM DETECTED ERROR
157          104430          PERR02= 104430          ;PROGRAM DETECTED ERROR
158          104431          PERR03= 104431          ;PROGRAM DETECTED ERROR
159          104432          PERR04= 104432          ;PROGRAM DETECTED ERROR
160          104433          PERR07= 104433          ;PROGRAM DETECTED ERROR
161          104434          PERR10= 104434          ;PROGRAM DETECTED ERROR
162          104435          PERR11= 104435          ;PROGRAM DETECTED ERROR
163          104436          PERR12= 104436          ;PROGRAM DETECTED ERROR
164          104437          PERR13= 104437          ;PROGRAM DETECTED ERROR
165          104440          PERR14= 104440          ;PROGRAM DETECTED ERROR
166          104441          PERR15= 104441          ;PROGRAM DETECTED ERROR
167          104442          PERR16= 104442          ;PROGRAM DETECTED ERROR
168          104443          PERR17= 104443          ;PROGRAM DETECTED ERROR
169          104444          PERR20= 104444          ;PROGRAM DETECTED ERROR
170          104445          PERR21= 104445          ;PROGRAM DETECTED ERROR
171          104446          PERR22= 104446          ;PROGRAM DETECTED ERROR
172          104447          PERR23= 104447          ;PROGRAM DETECTED ERROR
173          104450          PERR24= 104450          ;PROGRAM DETECTED ERROR
174          104451          PERR25= 104451          ;PROGRAM DETECTED ERROR
175          104452          PERR26= 104452          ;PROGRAM DETECTED ERROR
176          104453          PERR27= 104453          ;PROGRAM DETECTED ERROR
177          104454          PERR30= 104454          ;PROGRAM DETECTED ERROR
178          104455          PERR31= 104455          ;PROGRAM DETECTED ERROR
179          104456          PERR32= 104456          ;PROGRAM DETECTED ERROR
180          104457          PERR33= 104457          ;PROGRAM DETECTED ERROR
181          104460          PERR34= 104460          ;PROGRAM DETECTED ERROR
182          104461          PERR35= 104461          ;PROGRAM DETECTED ERROR
183          104462          PERR36= 104462          ;PROGRAM DETECTED ERROR
184          104463          PERR37= 104463          ;PROGRAM DETECTED ERROR
185          104464          PERR40= 104464          ;PROGRAM DETECTED ERROR
186          104465          PERR41= 104465          ;PROGRAM DETECTED ERROR
187          104466          PERR42= 104466          ;PROGRAM DETECTED ERROR
188          104467          PERR43= 104467          ;PROGRAM DETECTED ERROR
189
190          104470          ECCDIS= 104470          ;DISABLE ECC ON ALL CSR'S
191          104471          ECC1DIS=104471          ;DISABLE ECC ON 1 SELECTED CSR
192          104472          ECCINIT=104472          ;INITIALIZE ALL ECC CSR'S
193          104473          ECC1INIT=104473         ;INITIALIZE 1 SELECTED ECC CSR
194          104474          CBCSR=  104474          ;WRITE GENERATED CHECKBITS IN ALL CSR'S
195          104475          CB1CSR= 104475          ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
196          104476          WASSBE= 104476          ;WAS THERE A SBE ON ANY CSR?
197          104477          WAS1SBE=104477          ;WAS THERE A SBE ON 1 SELECTED CSR?
198          104500          WASDBE= 104500          ;WAS THERE A DBE ON ANY CSR?
199          104501          WAS1DBE=104501          ;WAS THERE A DBE ON 1 SELECTED CSR?
200          104502          CLRCSR= 104502          ;CLEAR ALL CSR'S
201          104503          CLR1CSR=104503          ;CLEAR 1 SELECTED CSR
202          104504          CHKDIS= 104504          ;DISABLE ECC & WRITE CHECKBITS FROM ALL CSR'S
203          104505          CHK1DIS=104505          ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
204          104506          ENASBE= 104506          ;ENABLE TRAPS ON SBE'S FROM ALL CSR'S
205          104507          ENA1SBE=104507          ;ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
206          104510          TSTREAD=104510          ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
207          104511          INVALID=104511          ;INVALIDATE BACKGROUND PATTERN ON 'BANK''
208          104512          ERRGEN =104512          ;CHECK ERROR ADDRESS
```

```
                                .SBTTL  DEFINE  BASIC PDP11 STUFF

                        ;*INITIAL ADDRESS OF THE STACK POINTER
002000                  STACK=  2000            ;;FIRST ADDRESS OF THE STACK
002000                  KERSTK= STACK           ;;KERNEL STACK
000740                  SUPSTK= 740             ;;SUPERVISOR STACK
000700                  USESTK= 700             ;;USER STACK
104000                  ERROR=EMT               ;;BASIC DEFINITION OF ERROR CALL
000004                  SCOPE=IOT               ;;BASIC DEFINITION OF SCOPE CALL
177776                  PSW=    177776          ;;PROCESSOR STATUS WORD
                        ;STKLMT=177774          ;;STACK LIMIT REGISTER
                        ;PIRQ=  177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
177546                  LKS=    177546          ;;LINE CLOCK (KW11-L) STATUS REGISTER

                        ;*MISCELLANEOUS DEFINITIONS
000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                  LF=     12              ;;CODE LINE FEED
000015                  CR=     15              ;;CODE CARRIAGE RETURN
000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED

                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
                        ;SP=R6                 ;;STACK POINTER
                        ;KSP=SP                ;;KERNEL STACK POINTER
000006                  SSP=SP                  ;;SUPERVISOR STACK POINTER
000006                  USP=SP                  ;;USER STACK POINTER
                        ;PC=R7                 ;;PROGRAM COUNTER

                        ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW9=    1000
000400                  SW8=    400
000200                  SW7=    200
000100                  SW6=    100
000040                  SW5=    40
000020                  SW4=    20
000010                  SW3=    10
000004                  SW2=    4
000002                  SW1=    2
000001                  SW0=    1

                        ;*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                  BIT9=   1000
000400                  BIT8=   400
000200                  BIT7=   200
```

```
000100                  BIT6=   100
000040                  BIT5=   40
000020                  BIT4=   20
000010                  BIT3=   10
000004                  BIT2=   4
000002                  BIT1=   2
000001                  BIT0=   1

                        ;*BASIC ''CPU'' TRAP VECTOR ADDRESSES
000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
                        ;TBITVEC=14             ;;''T'' BIT
                        ;TRTVEC=        14              ;;TRACE TRAP
                        ;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
                        ;TPVEC= 64              ;;TTY PRINTER VECTOR
                        ;LKVEC= 100             ;;LINE CLOCK (KW11-L) VECTER
000114                  CACHVEC=114             ;;CACHE ERROR INTERRUPT VECTOR
000114                  PARVEC=CACHVEC
                        ;PIRQVEC=240            ;;PROGRAM INTERRUPT REQUEST VECTOR
000250                  MMVEC=  250             ;;MEMORY MANAGEMENT VECTOR
                                .SBTTL  DEFINE  CACHE REGISTERS
                        ;MEMERR = 177744                ;;CACHE ERROR REGISTER
177746                  CONTRL = 177746                 ;;MEMORY CONTROL REGISTER
177750                  MAINT  = 177750                 ;;MEMORY MAINTENENCE REGISTER
                        ;HITMIS = 177752                ;;HIT MISS REGISTER ''1'' IMPLIES HIT IN CACHE
177754                  DATARG  = 177754                ;;DATA REGISTER

                                .SBTTL  DEFINE  CPU REGISTERS
177766                  CPUERR = 177766         ;;CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED

                                .SBTTL  DEFINE  MEMORY MANAGEMENT REGISTERS
                                ;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES
177572                  MMR0=   177572
177574                  MMR1=   177574
177576                  MMR2=   177576
172516                  MMR3=   172516

                                ;*USER ''I'' PAGE DESCRIPTOR REGISTERS
177600                  UIPDR0= 177600
                        ;UIPDR1=        177602
                        ;UIPDR2=        177604
                        ;UIPDR3=        177606
                        ;UIPDR4=        177610
                        ;UIPDR5=        177612
                        ;UIPDR6=        177614
                        ;UIPDR7=        177616

                                ;*USER ''D'' PAGE DESCRIPTOR REGISTORS
                        ;UDPDR0=        177620
                        ;UDPDR1=        177622
                        ;UDPDR2=        177624
                        ;UDPDR3=        177626
```

```
                                ;UDPDR4=        177630
                                ;UDPDR5=        177632
                                ;UDPDR6=        177634
                                ;UDPDR7=        177636

                                        ;*USER ''I'' PAGE ADDRESS REGISTERS
        177640                  FASTCITY=UIPAR0
        177640                  UIPAR0= 177640          ;PATTERN PROGRAM SPACE
        177642                  UIPAR1= 177642          ;PATTERN PROGRAM SPACE
        177644                  UIPAR2= 177644          ;PATTERN PROGRAM SPACE
        177646                  UIPAR3= 177646          ;PATTERN PROGRAM SPACE
        177650                  UIPAR4= 177650          ;PATTERN PROGRAM SPACE
        177652                  UIPAR5= 177652          ;PATTERN PROGRAM SPACE
        177654                  UIPAR6= 177654          ;PATTERN PROGRAM SPACE
                                ;UIPAR7=        177656          ;PATTERN PROGRAM SPACE

                                        ;*USER 'D'' PAGE ADDRESS REGISTERS
        177660                  UDPAR0= 177660          ;PATTERN PROGRAM SPACE
                                ;UDPAR1=        177662          ;PATTERN PROGRAM SPACE
                                ;UDPAR2=        17766           ;PATTERN PROGRAM SPACE
                                ;UDPAR3=        177666          ;PATTERN PROGRAM SPACE
                                ;UDPAR4=        177670          ;PATTERN PROGRAM SPACE
                                ;UDPAR5=        177672          ;PATTERN PROGRAM SPACE
                                ;UDPAR6=        177674          ;PATTERN PROGRAM SPACE
        177676                  UDPAR7= 177676          ;PATTERN PROGRAM SPACE

                                        ;*SUPERVISOR ''I'' PAGE DESCRIPTOR REGISTERS
        172200                  SIPDR0= 172200
                                ;SIPDR1=        172202
                                ;SIPDR2=        172204
                                ;SIPDR3=        172206
                                ;SIPDR4=        172210
                                ;SIPDR5=        172212
                                ;SIPDR6=        172214
                                ;SIPDR7=        172216

                                        ;*SUPERVISOR 'D'' PAGE DESCRIPTOR REGISTERS
                                ;SDPDR0=        172220
                                ;SDPDR1=        172222
                                ;SDPDR2=        172224
                                ;SDPDR3=        172226
                                ;SDPDR4=        172230
                                ;SDPDR5=        172232
                                ;SDPDR6=        172234
                                ;SDPDR7=        172236

                                        ;*SUPERVISOR ''I'' PAGE ADDRESS REGISTERS
        172240                  SIPAR0= 172240
                                ;SIPAR1=        172242
                                ;SIPAR2=        172244
        172246                  SIPAR3= 172246          ;TEST AREA
                                ;SIPAR4=        172250          ;TEST AREA
        172252                  SIPAR5= 172252          ;TEST AREA
        172254                  SIPAR6= 172254          ;TEST AREA
                                ;SIPAR7=        172256

                                        ;*SUPERVISOR 'D'' PAGE ADDRESS REGISTERS
```

```
                                SDPAR0= 172260
                                ;SDPAR1=        172262
                                ;SDPAR2=        172264
                                ;SDPAR3=        172266
                                ;SDPAR4=        172270
                                SDPAR5= 172272
                                SDPAR6= 172274
                                SDPAR7= 172276

                                        ;*KERNEL ''I'' PAGE DESCRIPTOR REGISTERS
                                KIPDR0= 172300
                                ;KIPDR1=        172302
                                ;KIPDR2=        172304
                                ;KIPDR3=        172306
                                ;KIPDR4=        172310
                                ;KIPDR5=        172312
                                ;KIPDR6=        172314
                                ;KIPDR7=        172316

                                        ;*KERNEL 'D'' PAGE DESCRIPTOR REGISTERS
                                ;KDPDR0=        172320
                                ;KDPDR1=        172322
                                ;KDPDR2=        172324
                                ;KDPDR3=        172326
                                ;KDPDR4=        172330
                                ;KDPDR5=        172332
                                ;KDPDR6=        172334
                                ;KDPDR7=        172336

                                        ;*KERNEL ''I'' PAGE ADDRESS REGISTERS
                                KIPAR0= 172340
                                ;KIPAR1=        172342
                                ;KIPAR2=        172344
                                ;KIPAR3=        172346
                                KIPAR4= 172350
                                KIPAR5= 172352
                                KIPAR6= 172354
                                ;KIPAR7=        172356

                                        ;*KERNEL 'D'' PAGE ADDRESS REGISTERS
                                KDPAR0= 172360
                                ;KDPAR1=        172362
                                ;KDPAR2=        172364
                                ;KDPAR3=        172366
                                ;KDPAR4=        172370
                                ;KDPAR5=        172372
                                KDPAR6= 172374
                                KDPAR7= 172376
```

```
                                .SBTTL  DEFINE  UNIBUS MAP REGISTERS
                                ;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
                                ;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'
170200                          MAPL0 = 170200
170202                          MAPH0 = 170202
170204                          MAPL1 = 170204
                                ;MAPH1 = 170206
                                ;MAPL2 = 170210
                                ;MAPH2 = 170212
                                ;MAPL3 = 170214
                                ;MAPH3 = 170216
                                ;MAPL4 = 170220
                                ;MAPH4 = 170222
                                ;MAPL5 = 170224
                                ;MAPH5 = 170226
                                ;MAPL6 = 170230
                                ;MAPH6 = 170232
                                ;MAPL7 = 170234
                                ;MAPH7 = 170236
                                ;MAPL10 = 170240
                                ;MAPH10 = 170242
                                ;MAPL11 = 170244
                                ;MAPH11 = 170246
                                ;MAPL12 = 170250
                                ;MAPH12 = 170252
                                ;MAPL13 = 170254
                                ;MAPH13 = 170256
                                ;MAPL14 = 170260
                                ;MAPH14 = 170262
                                ;MAPL15 = 170264
                                ;MAPH15 = 170266
                                ;MAPL16 = 170270
                                ;MAPH16 = 170272
                                ;MAPL17 = 170274
                                ;MAPH17 = 170276
                                ;MAPL20 = 170300
                                ;MAPH20 = 170302
                                ;MAPL21 = 170304
                                ;MAPH21 = 170306
                                ;MAPL22 = 170310
                                ;MAPH22 = 170312
                                ;MAPL23 = 170314
                                ;MAPH23 - 170316
                                ;MAPL24 = 170320
                                ;MAPH24 = 170320
                                ;MAPL25 = 170324
                                ;MAPH25 = 170326
                                ;MAPL26 = 170330
                                ;MAPH26 = 170332
                                ;MAPL27 = 170334
                                ;MAPH27 = 170336
                                ;MAPL30 = 170340
                                ;MAPH30 = 170342
                                ;MAPL31 = 170344
                                ;MAPH31 = 170346
                                ;MAPL32 = 170350
                                ;MAPH32 = 170352
```

```
                        ;MAPL33 = 170354
                        ;MAPH33 = 170356
                        ;MAPL34 = 170360
                        ;MAPH34 = 170362
                        ;MAPL35 = 170364
                        ;MAPH35 = 170366
                        ;MAPL36 = 170370
                        ;MAPH36 = 170372
                        ;MAPL37 = 170374
                        ;MAPH37 = 170376

                                .SBTTL  DEFINE  SOFTWARE SWITCH & DISPLAY REGISTERS
000174                  DISPREG=174
000176                  SWREG=  176

                                .SBTTL  DEFINE  CONTROL STATUS REGISTERS
172100                  CSRADD=172100

                                .SBTTL  DEFINE  PARAMETERS
060000                  FIRST=60000             ;START OF THE 16K TEST PATTERN AREA
157776                  LAST=157776             ;END OF THE 16K TEST PATTERN AREA
040000                  SIZE=40000              ;SIZE OF THE 16K TEST PATTERN AREA (FOR SOB INSTRUCTIONS)
```

```
                        .LIST   MD                      ;BE NICE TO SEE MY DEFINITIONS
                        .SBTTL  MACRO   FATAL
                ;********************* FATAL ***************************
                ;
                ;FATAL IS USED TO REPORT FATAL ERRORS (ERRORS THAT PREVENT
                ;       THE PROGRAM FROM CONTINUING).
                ;
                ;*****************************************************
                        .MACRO  FATAL   ARG             ;***MACRO***MACRO***MACRO***
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        INC     FATAL$                  ;SET FATAL INDICATOR
                        ERROR   +ARG
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   FATAL

                        .SBTTL  MACRO   TYPE
                        .MACRO  TYPE    ARG
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        .IF B ARG
                        TYPEIT
                        .IFF
                        TYPEIT  ,ARG
                        .ENDC
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   TYPE
```

```
                        .SBTTL  MACRO   NEWTST
                ;********************** NEWTST **********************
                ;NEWTST IS USED AS THE FIRST INSTRUCTION OF A TEST.
                ;IT WILL:
                ;1)  GENERATE A TEST NUMBER FOR THE LABEL OF THIS TEST
                ;2)  PUT STARS BEFORE AND AFTER A MESSAGE
                ;ARGUMENTS
                ;1)  ASCII  --  THIS IS THE MESSAGE THAT WILL APPEAR
                ;               ON THE LISTING
                ;2)  ICOUNT --  IF NON-BLANK AND BIT 11 OF $SWR = 1 IT WILL BE
                ;               THE NUMBER OF ITERATIONS TO MAKE ON THIS TEST
                ;3)  RETURN --  IF NON-BLANK WILL BE THE ADDRESS TO
                ;               WHICH THE NEXT SCOPE STATEMENT WILL
                ;               LOOP BACK TO.
                ;4)  COMAND --  IF NON-BLANK WILL BE THE FIRST
                ;               INSTRUCTION OF THE TEST
                ;               IF BLANK SCOPE WILL BE THE
                ;               FIRST INSTRUCTION
                ;*****************************************************
                        .MACRO  NEWTST  ASCII,ICOUNT,RETURN,COMAND
                        $STN=1
                        $NWTST=0
                        .NLIST  MC
                        .IF B <COMAND>
                        $$NEWTEST       \$TN,<ASCII>,SCOPE
                        .IFF
                        $$NEWTEST       \$TN,<ASCII>,<COMAND>
                        .ENDC
                        .NLIST
                        .LIST   ME
                        .LIST
                        .IF NE 4000&$SWR
                        .IF NB ICOUNT
                        .IF LE <ICOUNT-1>
                        MOV     #1,$TIMES       ;;DO 1 ITERATION
                        .IFF
                        MOV     #ICOUNT,$TIMES  ;;DO ICOUNT ITERATIONS
                        .ENDC
                        .ENDC
                        .IF NB RETURN
                        MOV     #RETURN,$LPADR  ;;SET SCOPE LOOP ADDRESS
                        .ENDC
                        .ENDC
                        .NLIST
                        .LIST   MC
                        .LIST
                        .NLIST  ME
                        .ENDM   NEWTST
```

```
                        .SBTTL  MACRO   $$NEWTEST
                        .MACRO  $$NEWTEST       A,ASC,COMND
                        .IRP    ASCI,<ASC>
                        .IF EQ $NWTST
                        $NWTST=1
                        .SBTTL  T'A'    ASCI
                        .NLIST
                        .LIST   ME
                        .LIST
                ;*******************************************************************************
                ;*TEST A         ASCI
                        .IFF
                        ASCI
                        .ENDC
                        .ENDM
                ;*******************************************************************************
                TST'A:  COMND
                        .NLIST  ME
                        $TN=$TN+1
                        .ENDM   $$NEWTEST

                        .SBTTL  MACRO   SUBTST
                ;************** SUBTST **********************************
                ;
                ;THIS MACRO WILL FORMAT A SUBTEST HEADING WITH STARS
                ;A .SBTTL WILL BE FORCED & .NLISTED FOR THE TABLE OF CONTENTS.
                ;
                ;ARGUMENT:
                ;1) TXT --      THIS IS THE MESSAGE THAT WILL APPEAR IN THE TABLE OF CONTENTS & LISTING.
                ;
                ;EXAMPLE:       SUBTST  <<THIS IS A FUN SUBTST>>
                ;
                ;*****************************************************

                        .MACRO  SUBTST  ASCII
                        .NLIST  MC
                        $SUBTST <ASCII>
                        .LIST   MC
                        .ENDM   SUBTST

                        .SBTTL  MACRO   $SUBTST
                        .MACRO  $SUBTST ASC
                        .IRP    ASCI,<ASC>
                        .SBTTL  ASCI
                        .NLIST
                        .LIST   ME
                        .LIST
                ;*******************************************************************************
                ;*SUBTEST        ASCI
                        .ENDM
                ;*******************************************************************************
                        .NLIST  ME
                        .ENDM   $SUBTST
```

```
                        .SBTTL  MACRO   TYPOCT
                ;******************** TYPOCT ********************
                ;
                ;TYPOCT IS USED TO CHANGE A BINARY NUMBER
                ;       TO A 6 DIGIT OCTAL NUMBER AND TYPE IT
                ;
                ;ARGUMENTS:
                ;
                ;1)     NUM     THE NUMBER TO BE TYPED
                ;
                ;2)     REMARK  ALLOWS A COMMENT TO BE MADE
                ;
                ;ROUTINES REQUIRED
                ;
                ;1)     CONVERT BINARY TO OCTAL AND TYPE (.$TYPOCT)
                ;
                ;2)     TYPE AN ASCIZ STRING (.$TYPE)
                ;
                ;EXAMPLES:
                ;
                ;1)     TYPOCT  HILMT,<TYPES THE CONTENTS OF 'HILMT>
                ;2)     TYPOCT  #5,<TYPES ' 000005'>
                ;
                ;*****************************************************************

                        .MACRO  TYPOCT  NUM,REMARK
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                MOV     NUM,-(SP)       ;;SAVE NUM FOR TYPEOUT
                        .IIF NB <REMARK>,                       ;;REMARK
                TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   TYPOCT
```

```
                        .SBTTL  MACRO   TYPOCS
                ;******************** TYPOCS ********************
                ;
                ;TYPOCS IS USED TO CHANGE A BINARY NUMBER TO AN OCTAL
                ;       NUMBER AND TYPE 1 TO 6 DIGITS
                ;       WITH OR WITHOUT LEADING ZEROS.
                ;
                ;ARGUMENTS:
                ;
                ;1)     NUM     NUMBER TO BE TYPED
                ;
                ;2)     REMARK  ALLOWS A COMMENT TO BE MADE
                ;
                ;3)     N       NUMBER OF DIGITS (1 TO 6) TO BE TYPED
                ;
                ;4)     Z       BLANK=SUPPRESS LEADING ZEROS (TYPES SPACES)
                ;               NON-BLANK=TYPE LEADING ZEROS
                ;
                ;ROUTINES REQUIRED
                ;
                ;1)     CONVERT BINARY TO OCTAL AND TYPE (.$TYPOCT)
                ;
                ;2)     TYPE AN ASCIZ STRING (.$TYPE)
                ;
                ;EXAMPLES:
                ;1)     TYPOCS  #12345,<TYPES '5'>,1
                ;2)     TYPOCS  #004,<TYPES '04'>,2,X
                ;3)     TYPOCS  #004,<TYPES ' 4'>,2
                ;
                ;*****************************************************************
                ;
                .MACRO  TYPOCS  NUM,REMARK,N,Z
                .NLIST
                .DSABL  CRF
                .IIF DF LST$$ .LIST  ME
                .ENABL  CRF
                .LIST
                MOV     NUM,-(SP)       ;;SAVE NUM FOR TYPEOUT
                .IIF NB <REMARK>,                               ;;REMARK
                TYPOS                   ;;GO TYPE--OCTAL ASCII
                .IF NB N
                .BYTE   N               ;;TYPE N DIGIT(S)
                .IFF
                .BYTE   6               ;;TYPE 6 DIGITS
                .ENDC
                .IF NB Z
                .BYTE   1               ;;TYPE LEADING ZEROS
                .IFF
                .BYTE   0               ;;SUPPRESS LEADING ZEROS
                .ENDC
                .DSABL  CRF
                .IIF DF LST$$ .NLIST  ME
                .ENABL  CRF
                .ENDM   TYPOCS
```

```
                        .SBTTL  MACRO   TYPDEC
                ;******************** TYPDEC ********************
                ;
                ;TYPDEC IS USE TO CHANGE A BINARY NUMBER TO A SIGNED
                ;       DECIMAL NUMBER AND TYPE IT REPLACING LEADING ZERO
                ;       WITH SPACES.
                ;NOTE:  IF THE NUMBER IS NEGATIVE A
                ;       MINUS SIGN WILL BE TYPED.
                ;
                ;ARGUMENTS:
                ;
                ;1)     NUM     NUMBER TO BE TYPED
                ;
                ;2)     REMARK ALLOWS A COMMENT TO BE MADE
                ;
                ;ROUTINES REQUIRED
                ;
                ;1)     CONVERT BINARY TO DECIMAL AND TYPE (.$TYPDEC)
                ;
                ;2)     TYPE AN ASCIZ STRING (.$TYPE)
                ;
                ;EXAMPLES
                ;
                ;1)     TYPDEC  SIZE,<TYPE THE CONTENTS OF SIZE>
                ;2)     TYPDEC  #-10.,<TYPE A MINUS TEN>
                ;
                ;*****************************************************************

                        .MACRO  TYPDEC  NUM,REMARK
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        MOV     NUM,-(SP)       ;;SAVE NUM FOR TYPEOUT
                        .IIF NB <REMARK>,                       ;;REMARK
                        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   TYPDEC
```

```
                        .SBTTL  MACRO   BMOV
                ;****************** BMOV ****************************
                ;
                ; THIS MACRO MOVES A BLOCK OF DATA.
                ;
                ;ARGUEMENTS:
                ;
                ;1)      FROMHERE        THE FIRST ADDRESS OF THE SOURCE BLOCK.
                ;
                ;2)      TOHERE          THE FIRST ADDRESS OF THE DESTINATION BLOCK.
                ;                        IF BLANK THE 1ST ADDRESS OF THE USER INSTRUCTION
                ;                        PAR'S IS USED (FASTCITY).
                ;
                ;3)      SIZE            THE SIZE OF THE SOURCE BLOCK.
                ;                        IF BLANK A 16 WORD TRANSFER IS ASSUMED.
                ;                        'WHY DEFAULT TO 16 WORDS?'' YOU ASK!
                ;                        'BECAUSE THAT'S HOW MANY WORDS TO THE USER PAR
                ;                        REGISTERS & THAT'S WHERE I INTEND TO MOVE LOTS
                ;                        OF STUFF.'' I REPLY!
                ;
                ;************************************************

                        .MACRO  BMOV    FROMHERE,TOHERE,SIZE
                          .IF B TOHERE
                            .NLIST
                            .DSABL      CRF
                            .IIF DF LST$$ .LIST  ME
                            .ENABL      CRF
                            .LIST
                            JSR R5,BLOCK1
                            FROMHERE
                            .DSABL      CRF
                            .IIF DF LST$$ .NLIST  ME
                            .ENABL      CRF
                            .MEXIT
                        .ENDC
                        .IF B SIZE
                            .NLIST
                            .DSABL      CRF
                            .IIF DF LST$$ .LIST  ME
                            .ENABL      CRF
                            .LIST
                            JSR R5,BLOCK2
                            TOHERE
                            FROMHERE
                            .DSABL      CRF
                            .IIF DF LST$$ .NLIST  ME
                            .ENABL      CRF
                            .MEXIT
                        .IFF
                            .NLIST
                            .DSABL      CRF
                            .IIF DF LST$$ .LIST  ME
                            .ENABL      CRF
                            .LIST
                            JSR R5,BLOCK3
                            SIZE
```

```
                                        TOHERE
                                        FROMHERE
                                        .DSABL      CRF
                                        .IIF DF LST$$ .NLIST  ME
                                        .ENABL      CRF
                                    .ENDC
                                .ENDM   BMOV
```

```
                        .SBTTL  MACRO   MAP
                ;****************** MAP ****************************
                ;
                ; THIS MACRO MAPS A MEMORY BANK (16K) INTO THE
                ; TEST PATTERN AREA (SUPERVISOR VIRTUAL (60000-157777)).
                ;
                ;ARGUEMENTS:
                ;
                ;1)     BANK    THE BANK OF 16K WORDS TO BE MAPPED.
                ;               THERE ARE 120 BANKS OF 16K WORDS
                ;
                ;EXAMPLES
                ;
                ;       MAP     LOC             ;LOCATION 'LOC'' CONTAINS THE # OF THE BANK TO MAP
                ;
                ;       MAP     #28.            ;BANK 34 (OCTAL) WILL BE MAPPED
                ;
                ;***************************************************

                .MACRO  MAP     BANK
                PUSH    R3
                .NLIST
                .DSABL  CRF
                .IIF DF LST$$ .LIST  ME
                .ENABL  CRF
                .LIST
                .IF B BANK
                MOV     #120.,R3
                .IFF
                MOV     BANK,R3
                .ENDC
                CALL    MAPPER
                .DSABL  CRF
                .IIF DF LST$$ .NLIST  ME
                .ENABL  CRF
                POP     R3
                .ENDM   MAP
```

```
                        .SBTTL  MACRO   SUPERVISOR
                ;****************** SUPERVISOR **********************
                ;
                ; THIS MACRO SWITCHES TO SUPERVISOR MODE.
                ;
                ;ARGUEMENTS: NONE.
                ;
                ;*****************************************************

                        .MACRO  SUPERVISOR
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        BIS     #BIT14,PSW                      ;GO TO SUPERVISOR MODE
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   SUPERVISOR

                        .SBTTL  MACRO   USER
                ;********************* USER *************************
                ;
                ; THIS MACRO SWITCHES TO USER MODE.
                ;
                ;ARGUEMENTS: NONE.
                ;
                ;*****************************************************

                        .MACRO  USER
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        BIS     #BIT15!BIT14,PSW                ;GO TO USER MODE
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   USER
```

```
                        .SBTTL  MACRO   TESTAREA
                ;****************** TESTAREA ************************
                ;
                ; THIS MACRO SWITCHES TO THE SPECIFIED TEST MODE.
                ;
                ;ARGUEMENTS: NONE.
                ;
                ;************************************·*****************

                        .MACRO  TESTAREA
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        BIS     TESTMODE,PSW                    ;GO TO SYSTEM TEST MODE
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   TESTAREA
```

```
                        .SBTTL  MACRO   SET4 & RES4
                ;****************** SET4 & RES4 **********************
                ;
                ; THESE MACROS SET & RESTORE VECTOR 4(TIMEOUT TRAP)
                ;
                ; IN IT'S RESTORED MODE TRAPS ARE REPORTED AS SUCH.
                ;
                ;ARGUEMENTS:    LOC     ;THE LOCATION TO VECTOR TO (ONLY USED IN ''SET4'' NOT ''RES4'')
                ;
                ;I USE THE SET4 AND RES4 MACROS AROUND CODE THAT I EXPECT TO TRAP TO 4
                ;LIKE LOOKING FOR ALL POSSIBLE CSR'S AND ETC.  WHENEVER CODE IS NOT
                ;SURROUNDED BY SET4 AND RES4 THEN ANY TRAPS TO 4 WILL CAUSE AN ERROR
                ;PRINTOUT THAT SAYS 'UNEXPECTED TRAP TO 4'' AND ALL THE ASSOCIATED REGISTER JUNK
                ;*****************************************************

                        .MACRO  SET4    ARG
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        MOV     ARG,4
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   SET4

                        .MACRO  RES4
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        MOV     #TIMEOUT,4
                        TST     NO22BIT         ;IS THIS AN 11/44?
                        BNE     101$            ;BRANCH IF NOT
                        CLR     CPUERR          ;CLEAR OUT THE CPU ERROR REGISTER BITS
                101$:                           ;THAT A EXPECTED TRAP COULD HAVE SET
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   RES4
```

```
                        .SBTTL  MACRO   DLEFT
                ;****************** DLEFT ****************************
                ;
                ; THIS MACRO DOES A DOUBLE WORD LEFT SHIFT
                ;
                ;ARGUEMENTS:    LOC     ;THE LOCATION TO BE SHIFTED LEFT (CARRY TO LOC+2)
                ;
                ;*****************************************************

                        .MACRO  DLEFT   ARG
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .L'ST
                        ROL     ARG
                        ROL     ARG+2
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   DLEFT
                        .NLIST  MD              ;DON'T NEED TO SFE THEM ANY MORE
```

```
                                        .SBTTL  TRAP CATCHER
                000000                  .=0
000000  000000  000000                  .WORD   0,0
                000177                  .REPT   177             ;.WORD  .+2,HALT
                                        .SBTTL  ACT11 HOOKS
                                ;*THE HOOKS REQUIRED BY ACT11 ARE DEFINED AND SETUP BELOW:
                                ;*      DEFINITIONS:
                                ;*                      1)LOC.46        ''END-OF-PASS'' HOOK
                                ;*                              =ADDRESS OF END OF PASS ROUTINE
                                ;*                               MODIFIED BY ACT11.
                                ;*                      2)LOC.52        PROGRAM NEEDS HOOK
                                ;*                              BIT 15=1 PROGRAM SHOULD BE POWER
                                ;*                                       FAILED WHILE RUNNING
                                ;*                                    =0 NO POWER FAIL
                                ;*                              BIT 14=1 PROGRAM MEMORY SIZE DEPENDENT
                                ;*                                    =0 NOT MEMORY SIZE DEPENDENT
                                ;*                              BIT 13=1 PROGRAM REQUIRES MANUAL INTERVENTION
                                ;*                                    =0  MANUAL INTERVENTION NOT REQUIRED
                                ;*                              BITS 12-0  MUST BE ZERO'S
                000046                  .-46
000046  013326                  $ENDAD                          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                000052                  .=52
000052  000020                  .WORD   BIT4                    ;;2)SET LOC.52 TO INDICATE MEMORY SIZE DEPENDANT
                                        .SBTTL  APT11 HOOKS
                000024                  .-24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                  200     ;;FOR APT START UP
                000042                  .-42
000042  002000                  STACK                           ;SO RT11 CAN START WITH RUN COMMAND
                000044                  .-44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044  061356                  $APTHDR ;;POINT TO APT HEADER BLOCK
                000200                  .-200
000200  000437          START3: BR      START1                  ;''NORMAL'' START
000202  000442                  BR      START2                  ;RESTART (SAVE ERROR ACCOUNTING)
                000300                  .=300
000300  005037  002566  START1: CLR     RESTART
000304  000137  003630          JMP     START
000310                  START2: SET     RESTART
000316  000137  003630          JMP     START
                002000                  .-STACK
```

```
                                .SBTT   VARIABLES       INITIALIZED TO ZERO
                        ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                        ;*USED IN THE PROGRAM.
002000                  $CMTAG:                         ;;START OF COMMON TAGS
002000  000000          SELONLY:0                       ;SELECT ONLY BANKS MARKED BY FIELD SERVICE MODE FLAG
002002  000000          DIAGFLAG:0                      ;SET FOR SHIFTING DIAGONAL TEST
002004  000000          KAMIKAZE:0                      ;SET FOR KAMIKAZE MODE TESTING
002006  000000          SKIPKAMI:0                      ;USED TO SKIP RESTORING KAMIKAZE MODE WHEN MODIFIED
                        ;NEXT TWO BYTES ARE DISPLAYED IN THE DISPLAY REGISTER
002010      000         $PATMAR:.BYTE   0               ;PATTERN NUMBER
002011      000         $BANK:  .BYTE   0               ;BANK & SIGN
002012      000         $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
002013      000         $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
002014  000000          LASTERROR:.WORD 0               ;NUMBER OF ERRORS ON LAST PASS
002016  000000          ERRPC:  .WORD   0               ;CONTAINS PC OF ERROR FOR TYPEOUT
002020  000000          BADPC:  .WORD   0               ;CONTAINS PC OF ERROR
002022  000000          ERRSP:  .WORD   0               ;CONTAINS SP OF ERROR FOR TYPEOUT
002024  000000          BADSP:  .WORD   0               ;CONTAINS SP OF ERROR
002026  000000          ERRPSW: .WORD   0               ;CONTAINS PSW OF ERROR FOR TYPEOUT
002030  000000          BADPSW: .WORD   0               ;CONTAINS PSW OF ERROR
002032  000000          ADDRESS:.WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
002034  000000          PADDRESS:.WORD  0               ;ADDRESS OF PARITY ERROR
002036  000000  000000  PHYADD: .WORD   0,0             ;22 BIT PHYSICAL ADDRESS
002042  000000          GOOD:   .WORD   0               ;;CONTAINS 'GOOD' DATA
002044  000000          GOOD2:  .WORD   0               ;;CONTAINS 'GOOD2' DATA
002046  000000          GOOD3:  .WORD   0               ;;CONTAINS 'GOOD3' DATA
002050  000000          BAD:    .WORD   0               ;;CONTAINS 'BAD' DATA
002052  000000          BAD2:   .WORD   0               ;;CONTAINS 'BAD2' DATA
002054  000000          BAD3:   .WORD   0               ;;CONTAINS 'BAD3' DATA
002056  000000          BADXOR: .WORD   0               ;XOR OF GOOD & BAD = BAD BITS!
002060  000000          $AUTO:  .WORD   0               ;;AUTOMATIC MODE INDICATOR FOR APT,ACT, & XXDP
002062  000000          FATAL$: .WORD   0               ;FATAL ERROR INDICATOR
002064  000000          SKPERR: .WORD   0               ;USED TO SKIP ERROR MESSAGE IN ''$ERRGEN''
002066  000000          NEMCNT: 0                       ;NON-EXISTANT MEMORY COUNTER (HOLES)
002070  000000          PARCNT: 0                       ;PARITY ERROR COUNTER
002072  000000          PATERR: 0                       ;PATTERN ERROR COUNTER
002074  000000          NOPAR:  0                       ;NO PARITY ERROR MODE INDICATOR
002076  000000          NONEM:  0                       ;NO NON-EXISTANT MEMORY (HOLES) MODE INDICATOR
002100  000000          BANK:   0                       ;MEMORY BANK UNDER TEST
002102  000000          BANKINDEX:0                     ;USED TO INDEX INTO CONFIG TABLE
002104  000000          CPUBIT: 0                       ;CONTAINS 1 BIT TO IDENTIFY CPU TO CONFIGURATION TABLE
002106  000000          MUT:    0                       ;MEMORY UNDER TEST FLAG
002110  000000          PATTERN:0                       ;PATTERN NUMBER UNDER TEST
002112  000000          KPFLAG: .WORD   0               ;BANK IS PROTECTED REGION OF ECC
002114  000000          ACFLAG: .WORD   0               ;BANK CAN BE ACCESSED BY THIS CPU
002116  000000          MKFLAG: .WORD   0               ;IF SET INDICATES MS11-M OR MF11S-K UNDER TEST
002120  000000          PFLAG:  .WORD   0               ;BANK IS IN PROGRAM SPACE
002122  000000          RRFLAG: .WORD   0               ;BANK IS WHERE PROGRAM RELOCATION IS REQUIRED TO TEST
002124  000000          RLFLAG: .WORD   0               ;PROGRAM IS RELOCATED FLAG
002126  000000          BMFLAG: .WORD   0               ;''BANK IS IDENTIFIED AS BAD MEMORY'' FLAG
002130  000000          EUFLAG: .WORD   0               ;''BANK HAS EUB MEMORY'' FLAG
002132  000000          TMFLAG: .WORD   0               ;''TYPE OF MEMORY TO TEST'' FLAG; 0 = PARITY, 1 = ECC
002134  000000          INTFLAG:.WORD   0               ;''BANK IS INTERLEAVED'' FLAG
002136  000000          INT64K: .WORD   0               ;''BANK IS 64K INTERLEAVED'' FLAG
002140  000000          ABORTFLAG:.WORD 0               ;''ABORT OCCURED'' FLAG
002142  000000          CTLKVEC:.WORD   0               ;HOLDS OLD KERNAL STACK POINTER IN CASE OF CNTL/K
002144  000000          CSR:    .WORD   0               ;DATA TO OR FROM CSR
```

```
1146 002146  000000                CSRNO:  0                       ;CSR ADDRESS NUMBER (4 LSB'S)
1147 002150  000000                OLDCSR: .WORD   0               ;OLD CSR NUMBER(USED IN INH PTR TEST)
1148                               ;THESE LOCATIONS STORE GPR'S DURING SUPERVISOR TESTS
1149 002152  000000                SUPDR0: 0
1150 002154  000000                SUPDR1: 0
1151 002156  000000                SUPDR2: 0
1152 002160  000000                SUPDR3: 0
1153 002162  000000                SUPDR4: 0
1154 002164  000000                SUPDR5: 0
1155 002166  000000                SUPDR6: 0
1156 002170  000000                DUMMY:  0                       ;DUMMY LOCATION FOR ADDRESS PASSING
1157                               ;THESE LOCATIONS STORE GPR'S & PSW DURING DETAILED ERROR PRINTOUTS
1158 002172  000000                DETR0:  0
1159 002174  000000                DETR1:  0
1160 002176  000000                DETR2:  0
1161 002200  000000                DETR3:  0
1162 002202  000000                DETR4:  0
1163 002204  000000                DETR5:  0
1164 002206  000000                DETSP:  0
1165 002210  000000                DETPSW: 0
1166 002212  000000                DETFLAG:0                       ;DETAILED REPORT FLAG
1167 002214  000000                CONTFLAG:0                      ;CSR'S HAVE BEEN TESTED FLAG
1168 002216  000000                TOTCSRS:.WORD   0               ;1 BIT PER EXISTING CSR, EG-
1169                                                               ;CSR 0 REPRESENTED BY BIT 15, ETC.
1170 002220  000000                CSRFIRST:.WORD  0               ;FIRST ADDRESS UNDER CONTROL OF THIS CSR
1171 002222  000000                CSRLAST: .WORD  0               ;
1172 002224  000000                CSRFBANK:.WORD  0               ;
1173 002226  000000                CSRLBANK:.WORD  0               ;
1174 002230  000000                CSRINT:  .WORD  0               ;
1175 002232  000000                SPLTCSR: .WORD  0               ;
1176 002234  000000  000000        DATBUF: .WORD   0,0             ;TWO WORD DATA BUFFER
1177 002240  000000  000000        TSTDAT: .WORD   0,0             ;TWO WORD TEST DATA
1178 002244  000000  000000        SBEMSK: .WORD   0,0             ;TWO WORD SINGLE BIT ERROR MASK
1179 002250  000000  000000        DBEMSK: .WORD   0,0             ;TWO WORD DOUBLE BIT ERROR MASK
1180 002254  000000                SUPDOADD:.WORD  0               ;ADDRESS OF SUBROUTINE TO EXECUTE IN SUPERVISOR MODE
1181 002256     000                PASFLG: .BYTE   0               ;LOCAL LOOP PASS CONTROL
1182 002257     000                UPPFLG: .BYTE   0               ;LOCAL LOOP PASS CONTROL
1183 002260  000000                REALPAT:.WORD   0               ;REAL PATTERN UNDER TEST
1184 002262  000000                OLDCACHE:.WORD  0               ;BACKED UP VALUE OF CACHE CONTROL REGISTER
1185 002264  000000                PARTHERE:.WORD  0               ;PARITY TRAPS SOMETIMES GO TO ADDRESS STORED HERE
1186 002266  000000                FSSTACK:.WORD   0               ;STACK SAVED HERE IF IN FIELD SERVICE MODE
1187 002270  000000                NEWBANK:.WORD   0               ;USED FOR RELOCATION TO A NEW BANK
1188 002272  000000                SOURCE: .WORD   0               ;SOURCE OF DATA WORDS FOR CHECKBIT GENERATION SUBROUTINE
1189 002274  000000                CHECK:  .WORD   0               ;CHECKBITS TO BE LOADED INTO CSR
1190 002276  000000                PCBUMP: .WORD   0               ;VALUE TO BUMP THE PC BY TO RECOVER AFTER A PARITY TRAP
1191 002300  000000                CSRINC: .WORD   0               ;VALUE TO INCREMENT ADDRESS BY TO REMAIN IN THE SAME CSR
1192 002302  000000                CSRLOOP:.WORD   0               ;LOOP CONTROL FOR CSR TESTING
1193 002304  000000                SUCCESS:.WORD   0               ;FLAG SET BY SUCCESSFULL TASK OR SUBROUTINE
1194 002306  000000                ZEROS:  .WORD   0               ;FOR AID IN "MOV" INSTRUCTIONS
1195 002310  000000                TIME:   .WORD   0               ;SECONDS THAT BATTERIES SHOULD LAST
1196 002312  000000                SKIPMK: .WORD   0               ;FLAG TO SKIP MKCONTROL SUBROUTINE
1197 002314  000000                NULLFLAG:.WORD  0               ;SET WHEN RUNNING NULL PATTERNS
1198 002316  000000                QVFLAG: 0                       ;FLAGS QUICK VERIFY PASS UNDER APT, ACT, OR XXDP CHAIN MODE
1199 002320  000000                ACTFLAG:0                       ;FLAGS ACT AUTOMATIC MODE PROGRAMMING RULES
1200 002322  000000                APTFLAG:0                       ;FLAGS APT AUTOMATIC MODE PROGRAMMING RULES
1201 002324  000000                XXDPCHAIN:0                     ;FLAGS XXDP CHAIN MODE PROGRAMMING RULES
1202                               ;NOTE: THESE TWO BYTES MUST STAY TOGETHER
```

```
1203 002326  000                  $NULL:  .BYTE   0              ;;CONTAINS NULL CHARACTER FOR FILLS
1204 002327  000                  $FILLS: .BYTE   0              ;CONTAINS # OF FILL CHARACTERS
1205 002330  000                  $TPFLG: .BYTE   0              ;;''TERMINAL NOT AVAILABLE'' FLAG
1206                                       .EVEN
1207 002332  000000               $ESCAPE:0                      ;;ESCAPE ON ERROR ADDRESS
1208 002334  000000               EVEN:   0                      ;USED FOR ALTERNATE DATA PATTERNS
1209 002336  000000               STRIPES:0                      ;COUNTS DIAGONAL STRIPES
1210 002340  000000               COUNT:  0                      ;BACKED UP COPY OF STRIPES
1211 002342  000000               NOTAB:  0                      ;NO TABLE BEING PRINTED - NOW
1212 002344  000000               BSIZE:  0                      ;SIZE OF 11/45 MOS MEMORY IN K WORDS
1213 002346  000000               KSIZE:  0                      ;SIZE OF MF11S-K MEMORY IN K WORDS
1214 002350  000000               LSIZE:  0                      ;SIZE OF MS11-L MEMORY IN K WORDS
1215 002352  000000               MSIZE:  0                      ;SIZE OF MS11-M MEMORY IN K WORDS
1216 002354  000000               PSIZE:  0                      ;SIZE OF UNIBUS PARITY MEMORY IN K WORDS
1217 002356  000000               TOOMANY:0                      ;FLAGS WHEN TOO MANY ERRORS HAVE BEEN PRINTED FOR A BANK
1218 002360  000000               READONLY:0                     ;FLAG TO PATTERNS TO READ ONLY
1219 002362  000000  000000       TESTADD:0,0                    ;THE ADDRESS TO RUN CSR TESTS ON
1220 002366  000000               UNITOP: 0                      ;HIGHEST ACCESSABLE BANK OF MEMORY THRU UNIBUS MAP
1221 002370  000000               STOPOK: 0                      ;FLAG TO ALLOW STOPPING WITH SWITCH REGISTER
1222 002372  000000               APTPAR: .WORD   0              ;AMOUNT OF PARITY MEMORY ACCORDING TO APT
1223 002374  000000               APTECC: .WORD   0              ;AMOUNT OF ECC MEMORY ACCORDING TO APT
1224 002376  000000               NOFSMODE:0                     ;FLAG TO DISABLE FIELD SERVICE MODE
1225 002400  000000               NOERROR:0                      ;''THIS IS NOT AN ERROR'' FLAG
1226 002402  000000               LOADBANK:0                     ;BANK LOADERS ARE RELOCATED TO
1227 002404  000000               TEMP:   0                      ;USED FOR JUNK
1228 002406  000000               QUICK:  0                      ;QUICK STOP FLAG FOR APT POWER FAIL
1229 002410  000000               NOSCOPE:0                      ;''NO SCOPE LOOP ALLOWED'' FLAG
1230 002412  000000               FSINFLAG:0                     ;''FIELD SERVICE - NO INTERNAL INTERLEAVE'' FLAG
1231 002414  000000               APTSIZE:0                      ;APT SIZING INFO FLAG
1232 002416  000000               FS7FLAG:0                      ;TRUE WHEN IN FIELD SERVICE COMMAND 7
1237 002420  000000               CONFGERROR:0                   ;CONFIGURATION ERROR FLAG
1238 002422  000000               I:      0                      ;USED FOR GENERAL PURPOSE INDEXING
1239 002424  000000               NO22BIT:0                      ;NO 22-BIT MODE FLAG
1240 002426  000000               NOSUPER:0                      ;NO SUPERVISOR MODE FLAG
1241 002430  000000               ERRADD: .WORD   0              ;HOLDS THE CSR'S ERROR ADDRESS
1242 002432  000000  000000  000000  CSRINFO:0,0,0,0,0,0,0,0     ;USED TO STORE INFORMATION ABOUT THE 16
     002440  000000  000000  000000
     002446  000000  000000
1243 002452  000000  000000  000000          0,0,0,0,0,0,0,0     ;POSSIBLE CSR'S
     002460  000000  000000  000000
     002466  000000  000000
1244 002472  000000               LINK1:  0                      ;USED TO HOLD LINKS TO PATTERNS WHICH
1245 002474  000000               LINK2:  0                      ;CAN EXECUTE IN THE PAR/PDR'S OR NOT
1246 002476  000000               CSRHOLD:0                      ;USED TO STORE CSR VALUES FOR CSR TESTS
1247 002500  000000               KFLAG:  0                      ;USED TO FLAG MF11S-K MEMORY TO TESTS
1248 002502  000000  000000       PGMCSR: .WORD   0,0            ;POINTS TO PROGRAM CSR
1249 002506  000000               INHECC: .WORD   0              ;FLAGS INHIBIT ECC TESTS ON RELOCATION
1250 002510  000000               INHBANK:.WORD   0              ;
1251 002512  000000               FULLREL:.WORD   0              ;
1289 002514                       $CMTGE: ;*END OF COMMON TAGS
```

```
                                        .SBTTL  VARIABLES       INITIALIZED TO NON ZERO
002514  000001  000000          CACHKN: 1,0                     ;CACHE CONSTANT (MOVED TO CONTRL TO TURN ON CACHE)
002520  001415                  CACHKF: 1415                    ;CACHE CONSTANT (MOVED TO CONTRL TO TURN OFF CACHE)
002522  040000                  TESTMODE:40000                  ;USED TO SELECT THE PROPER TEST MODE FOR A PATTERN RUN
002524  000012                  ERRMAX: 10.                     ;MAX # OF ERRORS PER BANK WITH SW11
002526  000167                  LASTBANK:167                    ;HIGHEST BANK OF MEMORY
002530  170000                  LASTBLOCK:170000                ;HIGHEST BANK OF MEMORY+1 (IN PAR FORMAT)
002532  000031                  SOBK:   25.                     ;SOB CONSTANT
002534  002000                  KSTACK: STACK                   ;STACK BEGINNING
002536  000001                  LOADHOME:1                      ;HOME BANK OF LOADERS
002540  177777                  WORST:  177777                  ;SET IF TESTING BANKS IN WORST FIRST MODE(1ST PASS)
002542  176543                  SEEDHI: 176543                  ;WORKING SEED HI (USED FOR RANDOM NUMBER GENERATOR)
002544  123456                  SEEDLO: 123456                  ;WORKING SEED LO (USED FOR RANDOM NUMBER GENERATOR)
002546  176543                  MSEEDH: 176543                  ;MASTER SEED HI (USED FOR RANDOM NUMBER GENERATOR)
002550  123456                  MSEEDL: 123456                  ;MASTER SEED LO (USED FOR RANDOM NUMBER GENERATOR)
002552  177777                  HEADER: 177777                  ;USED TO PRINT HEADINGS ONLY ONCE
002554  177777                  ONES:   177777                  ;FOR AID IN "MOV" INSTRUCTIONS
002556  000003                  FLIPLOC:3                       ;COUNTER FOR FLIPING DATA ON WORST CASE NOISE TEST
002560  052525                  SOFTPAT:52525                   ;PATTERN FOR SOFT ERROR BACKGROUND TESTS
002562  000000                  $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
002564  000000                  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
002566  000000                  RESTART:0                       ;RESTART (START ADD 202) FLAG
002570  000000                  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS

                                ;********************** NOTE THESE TWO LOCATIONS MUST STAY TOGETHER ************
002572  000377                  BAKPAT: .WORD   377             ;BACKGROUND PATTERN                             *
002574  177400                  SWAPAT: .WORD   177400          ;SWAPPED BAKPAT                                 *
                                ;*******************************************************************************

002576  177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
002600  177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
002602  177560                  $TKS:   177560                  ;;TTY KBD STATUS
002604  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
002606  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
002610  177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
002612     012                  $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED'
002613     207  377  377        $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
002616     000
002617     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
002620     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
002621     012  000              $LF:    .ASCIZ  <12>            ;;LINE FEED
                                        .EVEN
```

```
                                        .SBTTL  CONFIGURATION TABLE
                                ;CONFIG:FIRST 16K CONFIGURATION WORDS (2 EACH)
                                ;       2ND     16K CONFIGURATION WORDS (2 EACH)
                                ;       ........................
                                ;       200TH   16K CONFIGURATION WORDS (2 EACH)
                                ;
                                ;CONFIGURATION WORDS:
                                ;                       LOW:    BIT 0     ERRORS PRESENT
                                ;                               BIT 1     MEMORY SUCESSFULLY ACCESSED
                                ;                               BIT 2-4   RESERVED
                                ;                               BIT 5     SKIP ECC LOGIC TESTS FLAG (1-SKIP)
                                ;                               BIT 6     PROTECTED REGION OF ECC MEMORY
                                ;                               BIT 7     PROTECTED (PROGRAM SPACE)
                                ;                               BIT 8-11  CSR CODE
                                ;                               BIT 12-15 INTERLEAVED CSR CODE
                                ;                       HIGH:   BIT 0-7   NUMBER OF ERRORS
                                ;                               BIT 8-10  MEMORY TYPE
                                ;                               BIT 11    INTERLEAVED BOARD TYPE (0=128K, 1=4K)
                                ;                               BIT 12    INTERLEAVE ENABLED
                                ;                               BIT 13    "BACKGROUND PATTERN VALID" FLAG
                                ;                               BIT 14    BANK SELECTED FOR TEST BY FIELD SERVICE MODE
                                ;                               BIT 15    LOADERS HOME BANK
002624  000201                  CONFIG: .REPT   201
003630                          CONFIEND:
```

```
1365                                            .SBTTL  ********** MAIN **********
1366 003630                                START:  SUBTST  <<INITIALIZE VARIABLES TO ZERO>>
                                                ;*************************************************************************
                                                ;*SUBTEST        INITIALIZE VARIABLES TO ZERO
                                                ;*************************************************************************
1370 003630  000005                             RESET
1371 003632  013706  002534                     MOV     KSTACK,SP       ;;SETUP THE STACK POINTER
1377 003636  012700  002000                     MOV     #$CMTAG,R0      ;;FIRST LOCATION TO BE CLEARED
1378 003642  005020                        1$:  CLR     (R0)+           ;;CLEAR MEMORY LOCATION
1379 003644  022700  002514                     CMP     #$CMTGE,R0      ;;DONE?
1380 003650  001374                             BNE     1$              ;LOOP BACK IF NO
1381 003652                                     SUBTST  <<CLEAR NON-PROGRAM SPACE>>
                                                ;*************************************************************************
                                                ;*SUBTEST        CLEAR NON-PROGRAM SPACE
                                                ;*************************************************************************
1382                                            ;THIS ATTEMPS TO GET RID OF ANY PARITY ERRORS BY WRITING INTO
1383                                            ;EVERY LOCATION THAT IS NOT LOADED INTO BY THE PROGRAM OR ALLOCATED
1384                                            ;TO THE XXDP LOADERS
1385 003652  012737  000001  002074             MOV     #1,NOPAR                ;PARITY ACTION = COUNT & IGNORE
1386 003660  012700  001000                     MOV     #1000,R0
1387 003664  012720  000000                2$:  MOV     #0,(R0)+                ;CLEAR THE STACK AREA
1388 003670  020027  002000                     CMP     R0,#STACK
1389 003674  103773                             BLO     2$
1390 003676  012700  077662                     MOV     #END+2,R0
1391 003702  012720  000000                3$:  MOV     #0,(R0)+                ;CLEAR FROM THE END OF PROGRAM TO XXDP LOADERS
1392 003706  020027  152364                     CMP     R0,#152364
1393 003712  103773                             BLO     3$
1394 003714  005037  002074                     CLR     NOPAR                   ;RESTORE DEFAULT PARITY ACTION
1395
```

```
1404 003720                                    SUBTST  <<TYPE OF SYSTEM SIZER>>
                                       ;*****************************************************************************
                                       ;*SUBTEST       TYPE OF SYSTEM SIZER
                                       ;*****************************************************************************
1405 003720                            4$:     SET4    #4$
1406 003726  005737  177746                    TST     CONTRL                  ;SEE IF CACHE REGISTER RESPONDS
1407 003732                                    SET4    #9$                     ;YES - DO WE HAVE 11/44 TYPE CACHE
1408 003740  005737  177750                    TST     MAINT                   ;OR 11/60 TYPE CACHE?
1409 003744  000411                            BR      5$                      ;BRANCH IF 11/44 TYPE CACHE
1410 003746  012737  000014  002520  9$:      MOV     #14,CACHKF              ;TURN OFF CONSTANT FOR 11/60 CACHE
1411 003754  000405                            BR      5$
1412 003756  005037  002514            4$:     CLR     CACHKN                  ;NO CACHE ON SYSTEM
1413 003762  012737  002306  066404            MOV     #ZEROS,DT14             ;DO NOT PRINT CONTRL ERROR MESSAGES
1414 003770                            5$:     SET4    #6$
1415 003776  005737  172516                    TST     MMR3                    ;DO WE HAVE AN MMR3?
1416 004002  005037  172516                    CLR     MMR3                    ;YES WE DO
1417 004006  052737  000020  172516            BIS     #BIT4,MMR3              ;SEE IF THERE IS 22-BIT MODE
1418 004014  032737  000020  172516            BIT     #BIT4,MMR3
1419 004022  001023                            BNE     8$                      ;BRANCH IF 22-BIT RELOCATION
1420 004024  000411                            BR      7$                      ;BRANCH IF MMR3 BUT NO 22-BIT RELOC.
1421                                    ;* 11/34 TYPE MACHINES ENTER HERE
1422 004026  012737  140000  002522  6$:      MOV     #140000,TESTMODE        ;MAKE TEST MODE USER
1423 004034  005237  002426                    INC     NOSUPER                 ;NO SUPERVISOR MODE
1424 004040  005037  066254                    CLR     DT5+10
1425 004044  005037  066414                    CLR     DT14+10
1426                                    ;* 11/45 TYPE MACHINES ENTER HERE
1427 004050  005237  002424            7$:     INC     NO22BIT                 ;NO 22 BIT MODE
1428 004054  012737  000007  002526            MOV     #7,LASTBANK             ;124K MEMORY MAX. MEMORY SIZE
1429 004062  005037  066256                    CLR     DT5+12                  ;DO NOT TRY TO PRINT ERROR REGISTER
1430 004066  005037  066416                    CLR     DT14+12                 ;ERROR MESSAGES, BECAUSE THERE IS
1431                                                                            ;IS NO ERROR REGISTER!
1432 004072                            8$:     RES4
```

```
                                        SUBTST  <<INITIALIZE VARIABLES TO NON ZERO>>
                                ;*******************************************************************************
                                ;*SUBTEST        INITIALIZE VARIABLES TO NON ZERO
                                ;*******************************************************************************
                                        SET     WORST
004120  012737  000003  002556          MOV     #3,FLIPLOC
004126                                  SET     HEADER
004134  012737  176543  002546          MOV     #176543,MSEEDH
004142  012737  123456  002550          MOV     #123456,MSEEDL
004150  013737  002546  002542          MOV     MSEEDH,SEEDHI   ;PRIME THE RANDOM NUMBER GENERATOR
004156  013737  002550  002544          MOV     MSEEDL,SEEDLO   ;BOTH HIGH AND LOW WORDS
004164  012737  000377  002572          MOV     #377,BAKPAT
004172  012737  177400  002574          MOV     #177400,SWAPAT
004200                                  SUBTST  <<INITIALIZE VECTORS>>
                                ;*******************************************************************************
                                ;*SUBTEST        INITIALIZE VECTORS
                                ;*******************************************************************************
004200  012737  054154  000020          MOV     #$SCOPE,IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
004206  012737  000340  000022          MOV     #340,IOTVEC+2   ;;LEVEL 7
004214  012737  054446  000030          MOV     #$ERROR,EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
004222  012737  000340  000032          MOV     #340,EMTVEC+2   ;;LEVEL 7
004230  012737  061372  000034          MOV     #$TRAP,TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
004236  012737  000340  000036          MOV     #340,TRAPVEC+2;LEVEL 7
004244  012737  050450  000024          MOV     #$PWRDN,PWRVEC ;;POWER FAILURE VECTOR
004252  012737  000340  000026          MOV     #340,PWRVEC+2   ;;LEVEL 7
004260  012737  036420  000114          MOV     #PARITY,PARVEC;GET READY FOR PARITY ERRORS
004266  012737  000340  000116          MOV     #340,PARVEC+2
004274  012737  036614  000010          MOV     #PDP1105,RESVEC;RESERVED INSTRUCTION TRAP
004302  012737  000340  000012          MOV     #340,RESVEC+2
004310  012737  036570  000004          MOV     #TIMEOUT,ERRVEC;SETUP TIMEOUT ERRORS
004316  012737  000340  000006          MOV     #340,ERRVEC+2   ;SET PRIORITY OF ERROR TRAPS
004324  012737  036602  000250          MOV     #MMTRAP,MMVEC   ;VECTOR FOR MEMORY MANAGEMENT
004332  012737  000340  000252          MOV     #340,MMVEC+2
004340  104423                          CACHON                  ;TURN CACHE ON
```

```
1473 004342                             SUBTST  <<INITIALIZE PATTERNS>>
                                ;*****************************************************************************
                                ;*SUBTEST        INITIALIZE PATTERNS
                                ;*****************************************************************************
1474                                    ;THE APT E-TABLE DETERMINES WHICH PATTERNS ARE GOING TO BE RUN.
1475                                    ;EACH BIT SET REPRESENTS A PATTERN TABLE ENTRY THAT IS TO BE LEFT
1476                                    ;ALONE (TO BE RUN).  EACH BIT CLEARED REPRESENTS A PATTERN TABLE ENTRY
1477                                    ;THAT IS TO BE OVERLAYED WITH THE ADDRESS OF A NULL PATTERN.
1478 004342  012700  061342            MOV     #$DDW0,R0
1479 004346  012001                    MOV     (R0)+,R1
1480 004350  012703  016264            MOV     #MKCSRT,R3
1481 004354  012702  000020            MOV     #16.,R2
1482 004360  004737  004460            CALL    PATPLUG
1483 004364  012001                    MOV     (R0)+,R1
1484 004366  012702  000010            MOV     #8.,R2
1485 004372  004737  004460            CALL    PATPLUG
1486 004376  012001                    MOV     (R0)+,R1
1487 004400  012703  016514            MOV     #MKPAT,R3
1488 004404  012702  000020            MOV     #16.,R2
1489 004410  004737  004460            CALL    PATPLUG
1490 004414  012001                    MOV     (R0)+,R1
1491 004416  012702  000010            MOV     #8.,R2
1492 004422  004737  004460            CALL    PATPLUG
1493 004426  012001                    MOV     (R0)+,R1
1494 004430  012703  016700            MOV     #MJPAT,R3
1495 004434  012702  000020            MOV     #16.,R2
1496 004440  004737  004460            CALL    PATPLUG
1497 004444  012001                    MOV     (R0)+,R1
1498 004446  012702  000010            MOV     #8.,R2
1499 004452  004737  004460            CALL    PATPLUG
1500 004456  000417                    BR      SUBAAA
1501
1502 004460                         PATPLUG:SUBTST  <<SUBR  PLUG IN NULL PATTERNS>>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    PLUG IN NULL PATTERNS
                                ;*****************************************************************************
1503 004460                             FOR I := #1 TO R2
1504 004466  006001                       ROR   R1
1505 004470                               ON.NOERROR            ;IF CARRY CLEAR
1506 004472  012713  025074                 MOV #MT0999,(R3)
1507 004476                               END ;OF ON.ERROR
1508 004476  062703  000002               ADD #2,R3
1509 004502                             END ;OF FOR
1510 004514  000207                     RETURN
```

```
                                SUBAAA: SUBTST  <<CLEAR THE CONFIGURATION TABLE>>
                                ;*****************************************************************************
                                ;*SUBTEST        CLEAR THE CONFIGURATION TABLE
                                ;*****************************************************************************
                                        ;THIS ZEROS (UNLESS WE STARTED AT ADDRESS 202) THE CONFIG TABLE
                                        ;WHICH IS FULLY DISCRIBED AT LOCATION ''CONFIG''.
                                        .ENABLE LSB
004516                                  IF RESTART IS FALSE
004524  012700  002624                    MOV   #CONFIG,R0
004530  005020                  1$:       CLR   (R0)+
004532  022700  003630                    CMP   #CONFIEND,R0
004536  001374                            BNE   1$
004540                                  END ;OF IF RESTART
                                        .DSABL  LSB
004540  012737  000002  002104          MOV     #BIT1,CPUBIT            ;SET ID BIT
004546                                  SUBTST  <<SIZE FOR A HARDWARE SWITCH REGISTER>>
                                ;*****************************************************************************
                                ;*SUBTEST        SIZE FOR A HARDWARE SWITCH REGISTER
                                ;*****************************************************************************
                                        ;;IF NOT FOUND OR IT IS
                                        ;;EQUAL TO A ''-1'', SETUP FOR A SOFTWARE SWITCH REGISTER.
                                        .ENABL  LSB
004546                                  SET4    #3$             ;TRAPS TO 4 GOTO 3$
004554  012737  177570  002576          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWITCH REGISTER
004562  012737  177570  002600          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
004570                                  IF #-1 EQ @SWR          ;IF NO TRAP FROM REFERENCE TO @SWR AND @SWR   #-1
000403                                    BR    2$              ;;BRANCH IF NO TIMEOUT
004602  012716  004610          3$:       MOV   #2$,(SP)        ;;SET UP FOR TRAP RETURN
004606  000002                            RTI
004610                          2$:       RES4                  ;RESET TRAPS TO 4 TO DEFAULT
004630  012737  000176  002576            MOV   #SWREG,SWR      ;;POINT TO SOFTWARE SWR
004636  012737  000174  002600            MOV   #DISPREG,DISPLAY
004644                                  END ;OF IF #-1
                                        .DSABL  LSB
```

```
1543 004644                          SUBAAB: SUBTST  <<SETUP ACT, APT, & XXDP>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       SETUP ACT, APT, & XXDP
                                     ;*****************************************************************************
1544                                         ;THIS SETS UP A BUNCH OF FLAGS TO TELL THE PROGRAM EVERYTHING
1545                                         ;IT CARES TO KNOW ABOUT APT, ACT, & XXDP.
1546 004644  005037  061264                 CLR     $PASS           ;CLEAR PASS COUNT
1547 004650                                  IFB #BIT5 SET.IN $ENVM
1548 004660                                    SET   $TPFLG          ;INDICATE NO TERMINAL
1549 004666                                  END ;OF IFB #BIT5
1550 004666                                  IFB #BIT7 SET.IN $ENVM
1551 004676                                    SET   APTSIZE
1552 004704                                  END ;OF IFB #BIT7
1553 004704                                  IFB $ENV EQ #1
1554 004714                                    SET   APTFLAG,QVFLAG,$AUTO,QUICK
1555 004744  012737  043650  000024            MOV   #APTDOWN,PWRVEC
1556 004752  012737  061300  002576            MOV   #$SWREG,SWR     ;USE APT SWR
1557 004760                                  ELSE
1558 004762                                    IF 42 NE #STACK AND 42 NE #0
1559 005000                                      SET QVFLAG,$AUTO
1560 005014                                      IF 42 EQ #$ENDAD
1561 005024                                        SET       ACTFLAG
1562 005032                                      ELSE
1563 005034                                        SET       XXDPCHAIN
1564 005042                                      END ;OF IF 42
1565 005042                                    END ;OF IF 42
1566 005042                                  END ;OF IFB $ENV
```

```
1568 005042                                    SUBTST  <<PROTECT PROGRAM & LOADERS>>
                                       ;*****************************************************************************
                                       ;*SUBTEST       PROTECT PROGRAM & LOADERS
                                       ;*****************************************************************************
1569 005042  052737  000200  002624          BIS     #BIT7,CONFIG            ;PROTECT PROGRAM SPACE (BANK 0)
1570 005050  052737  000200  002630          BIS     #BIT7,CONFIG+4          ;PROTECT LOADER SPACE (BANK 1)
1571 005056                                  IF #$ENDAD NE 42                ;NOT ACT-11?
1572 005066                                    TYPE  MSG000                  ;TYPE PROGRAM TITLE
1573 005072                                  END ;OF IF #$ENDAD
1574
1575 005072                                    SUBTST  <<CHECK SYSTEM FOR CACHE>>
                                       ;*****************************************************************************
                                       ;*SUBTEST       CHECK SYSTEM FOR CACHE
                                       ;*****************************************************************************
1576                                           ;* THIS FIGURES OUT IF THERE IS A CACHE ON THE SYSTEM,
1577                                           ;* WHAT TYPE OF SYSTEM IT IS, AND WHETHER IT IS ENABLED
1578                                           ;* OR DISABLED.
1579 005072                                    SET4    #3$
1580 005100  005737  177746                    TST     CONTRL                  ;IS THERE A CONTROL REGISTER?
1581 005104                                    SET4    #2$
1582 005112  005737  177750                    TST     MAINT                   ;IS THERE A MAINTENANCE REGISTER?
1583 005116                                    SET4    #1$
1584 005124  005737  177754                    TST     DATARG                  ;IS THERE A DATA REGISTER?
1585 005130                                    TYPE    MSG117                  ; 11/44
1586 005134  000405                            BR      4$
1587 005136                             1$:    TYPE    MSG116                  ; 11/34
1588 005142  000402                            BR      4$
1589 005144                             2$:    TYPE    MSG118                  ; 11/60
1590 005150  052737  000014  177746     4$:    BIS     #BIT2!BIT3,CONTRL       ;SET CACHE DISABLE BITS
1591 005156  042737  000014  177746            BIC     #BIT2!BIT3,CONTRL       ;CLEAR CACHE DISABLE BITS
1592 005164  032737  000004  177746            BIT     #BIT2,CONTRL            ;IS THE BIT SET?
1593 005172  001004                            BNE     7$                      ;BRANCH IF THE BIT IS SET
1594 005174  032737  000010  177746            BIT     #BIT3,CONTRL            ;IS THE BIT SET?
1595 005202  001413                            BEQ     6$                      ;BRANCH IF THE BIT IS SET
1596 005204                             7$:    TYPE    MSG121                  ; CACHE BYPASSED
1597 005210  104424                            CACHOFF
1598 005212  013737  002514  002516            MOV     CACHKN,CACHKN+2         ;SAVE INFO ABOUT CACHE
1599 005220  005037  002514                    CLR     CACHKN                  ;CACHE CANNOT BE USED - IT'S BYPASSED
1600 005224  000404                            BR      8$
1601 005226                             3$:    TYPE    MSG119                  ; NO
1602 005232                             6$:    TYPE    MSG120                  ; CACHE AVAILABLE
1603
```

```
                                        SUBTST  <<SETUP USER & SUPERVISOR STACK>>
                                ;*******************************************************************************
                                ;*SUBTEST        SETUP USER & SUPERVISOR STACK
                                ;*******************************************************************************
1651 005236  104421            8$:     DEENERGIZE              ;TURN OFF MEMORY MANAGEMENT
1652 005240  005737  002426            TST     NOSUPER         ;IS THERE A SUPERVISOR MODE?
1653 005244  001011                    BNE     5$              ;NO-SKIP SUPERVISOR SETUP.
1654
1655                                    ;SET PREVIOUS MODE TO SUPERVISOR
1656 005246  042737  030000  177776    BIC     #BIT13.BIT12,PSW
1657 005254  052737  010000  177776    BIS     #BIT12,PSW
1658
1659 005262                            PUSH    #SUPSTK
1660 005266  006606                    MTPI    SSP
1661
1662                                    ;SET PREVIOUS MODE TO USER
1663 005270  052737  030000  177776  5$:     BIS     #BIT13!BIT12,PSW
1664
1665 005276                            PUSH    #USESTK
1666 005302  006606                    MTPI    USP
1667
1668 005304                            SUBTST  <<GET SOFTWARE SWITCH REGISTER IF NECESSARY>>
                                ;*******************************************************************************
                                ;*SUBTEST        GET SOFTWARE SWITCH REGISTER IF NECESSARY
                                ;*******************************************************************************
1672 005304                            IF $AUTO IS FALSE                       ;IF NOT(APT OR ACT)
1673 005312                              IF SWR EQ #SWREG               ;IF SOFTWARE SWITCH REG SELECTED
1674 005322  104407                        GTSWR                               ;;GET SOFT-SWR SETTINGS
1675 005324                              END ;OF IF SWR
1676 005324                            END ;OF IF $AUTO
1680
1681 005324                            SUBTST  <<GET MEMORY MANAGEMENT READY>>
                                ;*******************************************************************************
                                ;*SUBTEST        GET MEMORY MANAGEMENT READY
                                ;*******************************************************************************
1685 005324  104422                    KMAP                    ;MAP KERNEL SPACE 1 TO 1
1689 005326                            MAP                     ;MAP SUPERVISOR SPACE (TEST AREA) 1 TO 1
1690 005342  104420                    ENERGIZE                ;TURN ON MEMORY MANAGEMENT
```

```
1692
```

```
1695 005344                                     NEWTST  <<BIT TEST OF ALL CSR'S>>
                                        ;*****************************************************************************
                                        ;*TEST 1        BIT TEST OF ALL CSR'S
                                        ;*****************************************************************************
     005344  000004                     TST1:   SCOPE
1696                                    ;* THE FIRST PART OF THE CONFIGURATION ANALYSIS DOES THE FOLLOWING:
1697                                    ;*      1) FINDS WHICH CSR'S RESPOND, AND PUTS THEM INTO THE CSR INFORMATION
1698                                    ;*              TABLE, AND STORES ANOTHER BIT FOR 'TOTCSRS'.
1699                                    ;*      2) TESTS THE CSR BITS COMMON TO ALL CSR'S.
1700                                    ;*      3) FIGURES OUT IF THERE IS AN EUB BIT, AN ECC BIT, AND THE EXISTANCE
1701                                    ;*              OF THE ERROR ADDRESS BITS, AND MARKS THIS IN THE CSR
1702                                    ;*              INFORMATION TABLE.
1703                                    ;*      4) TESTS THE BITS PARTICULAR TO THAT TYPE OF CSR.
1704                                    ;*      5) IF ANY BITS TEST BAD IN THE CSR UNDER TEST, THE CSR OK BIT IN THE
1705                                    ;*              CSR INFORMATION TABLE IS CLEARED.
1706                                    ;* THE INFORMATION BITS ONE THROUGH THREE FORM A CODE WHICH GIVES THE TYPE
1707                                    ;* OF CSR:
1708                                    ;*                      ERR. ADDR.   PARITY  NOT EUB
1709                                    ;*      TYPE              BIT2        BIT1    BIT0     CODE TOTALS
1710                                    ;* UNIBUS PARITY          1           1       1          7
1711                                    ;*     MS11-L             1           1       0          6
1712                                    ;*     MF11S-K            1           0       1          5
1713                                    ;*     MS11-M             1           0       0          4
1714                                    ;* 11/45 BIPOLAR         0           1       1          3
1715                                    ;*
1716                                    ;* THIS MEMORY CODE WILL BE USED IN THE SECOND PART OF THIS ANALYSIS
1717                                    ;*
1718 005346  005005                             CLR     R5              ;R5 IS THE TOTAL CSR NUMBER
1719 005350  005000                             CLR     R0              ;R0 IS A TABLE INDEX
1720 005352  012703  172100                     MOV     #CSRADD,R3      ;R3 HAS THE CSR ADDRESS
1721 005356  012737  000001  002074             MOV     #1,NOPAR        ;IGNORE PARITY ERRORS
1722 005364                                     SET4    #2$
1723 005372  005713                     1$:     TST     (R3)    ;DOES THE CSR RESPOND?
1724 005374  052705  000001                     BIS     #1,R5
1725 005400  005004                             CLR     R4              ;CLEAR THE LAST CSR INDICATOR
1726 005402  052760  000007  002432             BIS     #7,CSRINFO(R0)  ;SET ALL THE MEMORY INFO BITS
1727 005410  052760  000030  002432             BIS     #BIT4!BIT3,CSRINFO(R0)  ;YES-MARK IT IN CSR INFORMATION TABLE
1728 005416  004537  005752                     JSR     R5,TEST         ;TEST BIT 0 AND 15
1729 005422  100001                             .WORD   BIT15!BIT0
1730 005424  012713  040000                     MOV     #BIT14,(R3)     ;IS THERE A BIT 14 RESPONDING
1731 005430  032713  040000                     BIT     #BIT14,(R3)     ;(IT'S THE EUB BIT)
1732 005434  001403                             BEQ     3$              ;BRANCH IF NO EUB BIT
1733 005436  042760  000001  002432             BIC     #BIT0,CSRINFO(R0) ;CLEAR EUB INFO IN THE CSR TABLE
1734 005444  005013                     3$:     CLR     (R3)            ;CLEAR THE CSR UNDER TEST
1735 005446  012713  020000                     MOV     #BIT13,(R3)     ;DOES BIT 13 RESPOND
1736 005452  032713  020000                     BIT     #BIT13,(R3)     ;(TO TEST FOR ECC CSR)
1737 005456  001417                             BEQ     4$              ;BRANCH IF NO, ECC CSR
1738 005460  042760  000002  002432             BIC     #BIT1,CSRINFO(R0) ;CLEAR PARITY INFO IN THE CSR TABLE
1739 005466  012713  020000                     MOV     #BIT13,(R3)     ;SET THE INHIBIT MODE POINTER TO 1ST 16K
1740 005472  004537  005752                     JSR     R5,TEST         ;TEST BIT 3
1741 005476  000010                             .WORD   BIT3
1742 005500  012713  020000                     MOV     #BIT13,(R3)
1743 005504  004537  005752                     JSR     R5,TEST         ;TEST BIT 1 AND 4
1744 005510  000022                             .WORD   BIT4!BIT1
1745 005512  012713  020000                     MOV     #BIT13,(R3)
```

```
1747 005516  004537  005752          4$:   JSR     R5,TEST          ;TEST BIT 2
1748 005522  000004                        .WORD   BIT2
1749 005524  005013                        CLR     (R3)
1750 005526  052713  007740                BIS     #7740,(R3)       ;ARE THERE ERROR ADDRESS BITS?
1751 005532  032713  007740                BIT     #7740,(R3)
1752 005536  001404                        BEQ     5$               ;BRANCH IF NO ERROR ADDR. BITS.
1753 005540  004537  005752                JSR     R5,TEST          ;TEST BITS 5->11
1754 005544  007740                        .WORD   7740
1755 005546  000403                        BR      6$               ;SKIP OVER THE INFORMATION REPORTING
1756 005550  042760  000004  002432  5$:   BIC     #BIT2,CSRINFO(R0) ;REPORT THAT THERE ARE NO ERROR ADDRESS BITS.
1757 005556  032760  000002  002432  6$:   BIT     #BIT1,CSRINFO(R0) ;IS THIS CSR AN ECC CSR?
1758 005564  001014                        BNE     7$               ;BRANCH IF NOT
1759 005566  032760  000001  002432        BIT     #BIT0,CSRINFO(R0) ;IS THE EUB BIT SET?
1760 005574  001410                        BEQ     7$               ;BRANCH IF IT IS
1761                                       ;WE MUST NOW TEST FOR MS11-M ON THE UNIBUS
1762 005576  012713  007760                MOV     #7760,(R3)       ;PUT PATTERN & SBE BIT INTO BITS 4->11
1763 005602  022713  007760                CMP     #7760,(R3)       ;ARE THEY STILL THERE?
1764 005606  001403                        BEQ     7$               ;YES - BRANCH FOR MF11S-K CSR
1765 005610  042760  000001  002432        BIC     #BIT0,CSRINFO(R0) ;NO - SET EUB BIT FOR MS11-M
1766 005616  005013                  7$:   CLR     (R3)             ;CLEAR CSR
1767 005620  022760  000040  002432        CMP     #40,CSRINFO(R0)  ;IS THIS A LEGAL CONFIGURATION?
1768 005626  100004                        BPL     10$              ;BRANCH IF IT'S LEGAL
1769 005630  016037  002432  002050        MOV     CSRINFO(R0),BAD
1770 005636  104021                        ERROR   +21              ;ILLEGAL TYPE ERROR
1771 005640  032760  000004  002432  10$:  BIT     #BIT2,CSRINFO(R0) ;DOES THIS CSR HAVE ERROR BITS
1772 005646  001016                        BNE     2$               ;BRANCH IF TRUE
1773 005650  032760  000002  002432        BIT     #BIT1,CSRINFO(R0) ;ARE THE OTHER 2 BITS SET?
1774 005656  001404                        BEQ     11$              ;BRANCH IF NOT
1775 005660  032760  000001  002432        BIT     #BIT0,CSRINFO(R0)
1776 005666  001006                        BNE     2$
1777 005670  016037  002432  002050  11$:  MOV     CSRINFO(R0),BAD
1778 005676  104021                        ERROR   +21              ;ILLEGAL TYPE ERROR
1779 005700  005060  002432                CLR     CSRINFO(R0)      ;CLEAR THE CSR INFO-IT WILL NOT EXIST IN THE PROGRAM
1780 005704  062700  000002          2$:   ADD     #2,R0            ;INCREMENT TO NEXT CSR TABLE
1781 005710  062703  000002                ADD     #2,R3            ;INCREMENT TO NEXT CSR
1782 005714  006305                        ASL     R5
1783 005716  103001                        BCC     8$               ;IS THERE A CSR 0?
1784 005720  005204                        INC     R4               ;YES - SET CSR PRESENT FLAG
1785 005722  022700  000040          8$:   CMP     #40,R0           ;ARE WE DONE?
1786 005726  001221                        BNE     1$               ;BRANCH IF MORE TO DO
1787 005730  000241                        CLC
1788 005732  006005                        ROR     R5               ;RESYNC R5
1789 005734  005704                        TST     R4               ;WAS THERE A CSR 0?
1790 005736  001402                        BEQ     9$               ;BRANCH IF NOT
1791 005740  052705  100000                BIS     #BIT15,R5        ;YES - SET IN THE CSR TABLE
1792 005744  010537  002216          9$:   MOV     R5,TOTCSRS       ;STORE R5 IN TOTCSRS
1793 005750  000436                        BR      ANA2             ;JUMP OVER SUBROUTINE
1794                                       ;THIS SUBROUTINE TESTS THE CSR BITS
1795 005752  012501                TEST:   MOV     (R5)+,R1         ;GET THE BIT TO TEST
1796 005754  050113                        BIS     R1,(R3)          ;SET THAT IN THE CSR UNDER TEST
1797 005756  030113                        BIT     R1,(R3)          ;IS THE BIT STILL THERE?
1798 005760  001013                        BNE     1$               ;BRANCH IF STILL THERE
1799 005762  011337  002144                MOV     (R3),CSR         ;READ CSR
1800 005766  010137  002042                MOV     R1,GOOD
1801 005772  032713  100020                BIT     #BIT15.BIT4,(R3);IS IT BECAUSE OF A SBE OR DBE?
1802 005776  001004                        BNE     1$               ;BRANCH IF IT IS
1803 006000  104035                        ERROR   +35              ;BIT SET ERROR
```

```
1804 006002  042760  000010  002432          BIC     #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT
1805 006010  040113                  1$:     BIC     R1,(R3)           ;CLEAR THE SELECTED BIT
1806 006012  030113                          BIT     R1,(R3)           ;IS IT CLEARED?
1807 006014  001413                          BEQ     2$                ;BRANCH IF IT IS CLEARED
1808 006016  011337  002144                  MOV     (R3),CSR          ;READ CSR
1809 006022  010137  002042                  MOV     R1,GOOD
1810 006026  032713  100020                  BIT     #BIT15!BIT4,(R3);IS IT BECAUSE OF A SBE OR DBE?
1811 006032  001004                          BNE     2$                ;BRANCH IF TRUE
1812 006034  104010                          ERROR   +10               ;BIT CLEAR ERROR
1813 006036  042760  000010  002432          BIC     #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT
1814 006044  000205                  2$:     RTS     R5
1815
```

```
1817 006046                    ANA2:   SUBTST  <<MATCH ALL CSR'S WITH MEMORY>>
                               ;*****************************************************************************
                               ;*SUBTEST       MATCH ALL CSR'S WITH MEMORY
                               ;*****************************************************************************
1818                           ;* THE SECOND PART OF THE ANALYSIS MATCHES UP THE CSR'S WITH THE MEMORY, AND
1819                           ;* INSTALLS ALL THE INFORMATION FOUND IN THE CONFIGURATION TABLE.  FOR ECC,
1820                           ;* THIS IS DONE BY TAKING EACH CSR FOUND IN THE PREVIOUS SECTION SEQUENTIALLY
1821                           ;* AND CHECKING THROUGH ALL OF MEMORY, ONE BANK AT AT TIME, TO SEE WHICH BANKS
1822                           ;* RESPOND TO THE CSR IN QUESTION. THE FIRST DOUBLE WORD PAIR IN EACH BANK ARE
1823                           ;* WRITTEN WITH DATA AND DIAGNOSTIC CHECK MODE SET IN THE CSR IN ORDER TO AC-
1824                           ;* COMPLISH THIS. ALL POSSIBLE CONFIGURATIONS OF DOUBLE WORD PAIRS (NON-INTER-
1825                           ;* LEAVED, 64K INTERLEAVED, OR 128K INTERLEAVED) ARE CHECKED FOR EACH BANK
1826                           ;* THROUGH USE OF TESTADD AND KERNEL INSTRUCTION PAGE ADDRESS REGISTERS 4 AND
1827                           ;* 5. IF WE GET THE PROPER CHECKBITS BACK, WE HAVE A MATCH. IF NOT, THE ROUT-
1828                           ;* INE CHECKS FOR SINGLE OR DOUBLE BIT ERRORS.
1829                           ;*                 IF ONE OR THE OTHER IS FOUND, THE ERROR ADDRESS IS CHECKED
1830                           ;* TO SEE IF IT IS THAT BANK.  IF IT IS, WE HAVE A MATCH.  AT THE END OF EACH
1831                           ;* BANK PASS, FOR EACH CSR PASS, THE PROGRAM COMES UP WITH A NUMBER, STORED IN
1832                           ;* "I", WHICH DENOTES THE FOLLOWING:
1833                           ;*
1834                           ;*      I       MEMORY DESCRIPTION
1835                           ;*      -       ------------------
1836                           ;*      0       NON-EXISTANT MEMORY
1837                           ;*      1       NON-INTERLEAVED MEMORY
1838                           ;*      2       64K INTERLEAVED, A1 NOT ASSERTED MEMORY
1839                           ;*      3       128K INTERLEAVED, A1 NOT ASSERTED MEMORY
1840                           ;*      4       64K INTERLEAVED, A1 ASSERTED MEMORY
1841                           ;*      5       128K INTERLEAVED, A1 ASSERTED MEMORY
1842                           ;*
1843                           ;*  NOTE - I-2 THROUGH I=5 CAN ONLY OCCUR WITH MS11-M MEMORY.
1844                           ;*
1845                           ;* NOTE THAT PARITY MEMORY WRITES WRONG PARITY TO THE DOUBLE WORDS, THEN LOOKS
1846                           ;* FOR THE PARITY ERROR BIT TO BE SET.  IF THE BIT IS SET, WE HAVE A MATCH.
1847                           ;*
1848 006046  005037  002274              CLR     CHECK
1849 006052  000241                      CLC
1850 006054  013703  002216              MOV     TOTCSRS,R3
1851 006060  006303              70$:    ASL     R3
1852 006062  103403                      BCS     71$
1853 006064  005237  002274              INC     CHECK
1854 006070  000773                      BR      70$
1855 006072  006337  002274      71$:    ASL     CHECK
1856 006076                              SET4    #6$                     ;NE MEMORY TRAPS GO TO 4
1857 006104  005037  002502              CLR     PGMCSR                  ;SET PROGRAM CSR POINTER TO ZERO
1858 006110  012701  002362              MOV     #TESTADD,R1             ;SET UP THE VIRTUAL ADDR. POINTER
1859 006114  013703  002216              MOV     TOTCSRS,R3              ;MOVE CSR MAP INTO R3
1860 006120  005000                      CLR     R0                      ;CLEAR THE CSR POINTER
1861 006122  005005                      CLR     R5                      ;CLEAR THE PROGRAM CSR STATUS POINTER
1862 006124  005737  002424              TST     NO22BIT                 ;IS THIS AN 11/44?
1863 006130  001403                      BEQ     7$                      ;BRANCH IF IT IS
1864 006132  005037  002530              CLR     LASTBLOCK               ;ADJUST LASTBLOCK INDICATOR FOR 124K MACHINE
1865 006136  000413                      BR      1$                      ;BRANCH OVER NEXT PIECE OF CODE
1866 006140  022737  000167  002526  7$: CMP     #167,LASTBANK           ;IS THERE UNIBUS MEMORY ABOVE 17000000?
1867 006146  001407                      BEQ     1$                      ;BRANCH IF NOT
1868 006150  013702  002526              MOV     LASTBANK,R2             ;SET UP A NEW LAST BLOCK INDICATOR
1869 006154  005202                      INC     R2
1870 006156  072227  000011              ASH     #9.,R2
```

```
1871 006162  010237  002530                    MOV     R2,LASTBLOCK
1872 006166  012702  000004            1$:     MOV     #4,R2                   ;R2 IS INDEX FOR CONFIG TABLE
1873 006172  012737  001000  172350            MOV     #1000,KIPAR4            ;SET KIPAR4 FOR BANK 1
1874 006200  012737  001000  172352            MOV     #1000,KIPAR5            ;SET KIPAR5 FOR BANK 1
1875 006206  006303                    2$:     ASL     R3                      ;DOES THIS CSR EXIST?
1876 006210  103420                            BCS     3$                      ;BRANCH IF IT DOES EXIST
1877 006212  062700  000002                    ADD     #2,R0                   ;INCREMENT THE CSR POINTER
1878 006216  010037  002146                    MOV     R0,CSRNO                ;STORE IT IN CSRNO ALSO
1879 006222  005703                            TST     R3                      ;ARE THERE ANY MORE CSR'S TO DO?
1880 006224  001370                            BNE     2$                      ;BRANCH IF ALL CSRS NOT DONE
1881 006226  012737  001000  172350            MOV     #1000,KIPAR4            ;RESTORE KIPAR4
1882 006234  012737  001200  172352            MOV     #1200,KIPAR5            ;RESTORE KIPAR5
1883 006242  013706  002534                    MOV     KSTACK,SP               ;RESTORE STACK
1884 006246  000137  007550                    JMP     SUBAAS                  ;JUMP TO SUBAAS IF ALL CSR'S ARE DONE
1885 006252  010037  002146            3$:     MOV     R0,CSRNO                ;MAKE SURE CSRNO IS UPDATED
1886 006256  022705  000003                    CMP     #3,R5                   ;PROGRAM CSR FOUND?
1887 006262  001413                            BEQ     13$                     ;BRANCH IF TRUE
1888 006264  022705  000001                    CMP     #1,R5                   ;INTERLEAVED PROGRAM CSR HALF FOUND?
1889 006270  001403                            BEQ     14$                     ;BRANCH IF TRUE
1890 006272  110037  002502                    MOVB    R0,PGMCSR               ;SET UP PROGRAM CSR POINTER
1891 006276  000405                            BR      13$
1892 006300  110037  002503            14$:    MOVB    R0,PGMCSR+1             ;SET UP INTERLEAVED PROGRAM CSR POINTER
1893 006304  052737  100000  002502            BIS     #BIT15,PGMCSR           ;SET INTERLEAVED PROGRAM CSR FLAG
1894 006312  104424                    13$:    CACHOFF                         ;TURN THE CACHE OFF
1895 006314  000240                            NOP
1896 006316  012737  100000  002362    45$:    MOV     #100000,TESTADD         ;SET UP VIRTUAL ADDRESS TO KIPAR4
1897 006324  012737  120002  002364            MOV     #120002,TESTADD+2       ;SET UP VIRTUAL ADDRESS TO KIPAR5
1898 006332  032762  000040  002624            BIT     #BIT5,CONFIG(R2)        ;IS THIS A BANK TO SKIP?
1899 006340  001402                            BEQ     43$                     ;NO - BRANCH AROUND NEXT INSTRUCTION
1900 006342  000137  007474                    JMP     6$                      ;YES - GO TO END OF BANK
1901 006346  005037  002422            43$:    CLR     I                       ;CLEAR THE MEMORY CONFIGURATION COUNTER
1902 006352  005771  000000            4$:     TST     @(R1)                   ;TEST TO SEE THAT THERE IS MEMORY PRESENT
1903 006356  005237  002422                    INC     I
1904 006362                                    PUSH    @(R1),@2(R1)            ;SAVE THE LOCATIONS UNDER TEST
1905 006372  032760  000002  002432            BIT     #BIT1,CSRINFO(R0)       ;IS THIS PARITY MEMORY?
1906 006400  001414                            BEQ     34$                     ;NO - BRANCH
1907 006402  052760  000004  172100            BIS     #BIT2,CSRADD(R0)        ;SET WRITE WRONG PARITY
1908 006410  012771  123456  000000            MOV     #123456,@(R1)           ;SET THE FIRST LOCATION UNDER TEST
1909 006416  012771  123456  000002            MOV     #123456,@2(R1)          ;SET THE SECOND LUT
1910 006424  005060  172100                    CLR     CSRADD(R0)              ;CLEAR THE CSR
1911 006430  000411                            BR      41$                     ;TEST LOCATIONS
1912 006432  012771  123456  000000  34$:     MOV     #123456,@(R1)           ;SET THE FIRST LOCATION UNDER TEST
1913 006440  012771  123456  000002            MOV     #123456,@2(R1)          ;SET THE SECOND LUT
1914 006446  104503                            CLR1CSR                         ;RESET CSR
1915 006450  104475                            CB1CSR                          ;SET DIAG. CHECK MODE IN CSR UNDER TEST
1916 006452  000240                    40$:    NOP
1917 006454  005771  000000            41$:    TST     @(R1)                   ;READ THE FIRST LUT TO WRITE CKBITS. INTO CSR
1918 006460  104426                            READCSR                         ;READ THE CSR UNDER TEST
1919 006462  000240                            NOP
1920 006464  013704  002144                    MOV     CSR,R4                  ;GET THE CHECKBITS FROM THE CSR
1921 006470  000240                            NOP
1922 006472  010437  002404                    MOV     R4,TEMP                 ;SAVE IN TEMP FOR LATER
1923 006476  104503                            CLR1CSR                         ;RESET CSR
1924 006500                                    POP     @2(R1),@(R1)            ;RESTORE LOCATIONS UNDER TEST
1925 006510  032760  000002  002432            BIT     #BIT1,CSRINFO(R0)       ;IS THIS PARITY MEMORY?
1926 006516  001404                            BEQ     42$                     ;NO - BRANCH
1927 006520  005704                            TST     R4                      ;DID WE GET A PARITY ERROR?
```

```
1928 006522  100414                        BMI     25$                     ;YES - FILL IN CONFIG TABLE
1929 006524  000137  007474                JMP     6$                      ;NO - JUMP TO END OF BANK
1930 006530  072427  177773          42$:  ASH     #-5,R4                  ;MANIPULATE THE CSR BITS
1931 006534  042704  177600                BIC     #^C177,R4               ;INTO A USABLE FORM.
1932 006540  000240                        NOP
1933 006542  022704  000157                CMP     #157,R4                 ;DO THE CHECKBITS COMPARE TO WHAT WAS WRITTEN?
1934 006546  001402                        BEQ     25$                     ;BRANCH IF THERE IS A MATCH
1935 006550  000137  007154                JMP     22$                     ;ELSE BRANCH IF NOT THE SAME
1936                                        ;*
1937                                        ;* WE COME HERE IF THERE IS A MATCH
1938                                        ;*
1939 006554  010004                  25$:  MOV     R0,R4                   ;GET THE CSR NUMBER
1940 006556  006204                        ASR     R4                      ;SET IT UP FOR USE IN THE
1941 006560  000304                        SWAB    R4                      ;CONFIGURATION TABLE.
1942 006562  042704  170377                BIC     #170377,R4              ;CLEAR OFF EXTRANEOUS BITS
1943 006566  032737  000004  002422        BIT     #BIT2,I                 ;INTERLEAVED A1 ASSERTED MEMORY FOUND?
1944 006574  001402                        BEQ     15$                     ;BRANCH IF NOT
1945 006576  072427  000004                ASH     #4,R4                   ;PUT CSR NUMBER IN INTERLEAVED CSR SLOT
1946 006602  050462  002624          15$:  BIS     R4,CONFIG(R2)           ;PUT CSR NUMBER IN CONFIG. TABLE
1947 006606  016004  002432                MOV     CSRINFO(R0),R4          ;GET MEMORY TYPE
1948 006612  042704  177770                BIC     #^C7,R4                 ;CLEAR OFF THE EXTRANEOUS BITS
1949 006616  000304                        SWAB    R4                      ;MOVE INTO PROPER POSITION
1950 006620  050462  002626                BIS     R4,CONFIG+2(R2)         ;SET IT INTO THE CONFIG TABLE
1951 006624  022737  000001  002422        CMP     #1,I                    ;WAS THIS NON-INTERLEAVED MEMORY?
1952 006632  001431                        BEQ     24$                     ;BRANCH IF IT WAS
1953 006634  052762  010000  002626        BIS     #BIT12,CONFIG+2(R2)     ;SET THE INTERLEAVED BIT
1954 006642  010204                        MOV     R2,R4                   ;SAVE THE CURRENT BANK INDFX
1955 006644  032737  000001  002422        BIT     #BIT0,I                 ;WAS THIS 128K INTERLEAVED?
1956 006652  001006                        BNE     5$                      ;BRANCH IF TRUE
1957 006654  052762  004000  002626        BIS     #BIT11,CONFIG+2(R2)     ;SET 64K INTERLEAVED FLAG IN CONFIG
1958 006662  062704  000020                ADD     #20,R4                  ;SET NEW BANK POINTER TO 4 BANKS AHEAD
1959 006666  000402                        BR      16$                     ;JUMP OVER NEXT INSTRUCTION
1960 006670  062704  000040          5$:   ADD     #40,R4                  ;SET NEW BANK POINTER 8 BANKS AHEAD
1961 006674  052764  000040  002624  16$:  BIS     #BIT5,CONFIG(R4)        ;SET SKIP ECC LOGIC TESTS FLAG
1962 006702  056264  002624  002624        BIS     CONFIG(R2),CONFIG(R4)   ;SET OTHER INFO INTO THAT BANK
1963 006710  056264  002626  002626        BIS     CONFIG+2(R2),CONFIG+2(R4)
1964                                        ;*
1965                                        ;* THIS SECTION IS EXECUTED ONLY WHEN THE BANK=1
1966                                        ;*
1967 006716  022737  001000  172350  24$:  CMP     #1000,KIPAR4            ;IS THIS BANK 1?
1968 006724  001402                        BEQ     30$                     ;BRANCH IF TRUE
1969 006726  000137  007334                JMP     33$                     ;ELSE JUMP TO END OF THIS BANK
1970 006732  032737  100020  002404  30$:  BIT     #BIT15!BIT4,TEMP        ;WAS THERE A SBE OR DBE?
1971 006740  001417                        BEQ     44$                     ;BRANCH IF NOT
1972 006742  013704  002404                MOV     TEMP,R4                 ;GET CSR CONTENTS
1973 006746  072427  177767                ASH     #-9.,R4                 ;MAKE ERROR ADDRESS INTO BANK #
1974 006752  022704  000001                CMP     #1,R4                   ;ERROR IN BANKS 0 OR 1?
1975 006756  003010                        BGT     44$                     ;BRANCH IF NOT
1976 006760  052762  000001  002624        BIS     #BIT0,CONFIG(R2)        ;SET ERROR FLAG IN CONFIG TABLE
1977 006766  105262  002626                INCB    CONFIG+2(R2)            ;ADD ONE TO BANK ERROR COUNT
1978 006772                                SET     CONFGERROR              ;PRINT CONFIG TABLE
1979 007000  022737  000001  002422  44$:  CMP     #1,I                    ;WAS THIS NON-INTERLEAVED MEMORY?
1980 007006  001003                        BNE     9$                      ;BRANCH IF NOT
1981 007010  012705  000003                MOV     #3,R5                   ;SET PGM CSR FLAG TO DONE
1982 007014  000420                        BR      10$                     ;BRANCH OVER NEXT STUFF
1983 007016  032737  000004  002422  9$:   BIT     #BIT2,I                 ;A1 ASSERTED INTERLEAVED MEMORY?
1984 007024  001010                        BNE     11$                     ;BRANCH IF TRUE
```

```
1985 007026  110037  002503                    MOVB    R0,PGMCSR+1             ;SET PGM CSR IN HI BYTE
1986 007032  052737  100000  002502            BIS     #BIT15,PGMCSR           ;SET INTERLEAVED PGM CSR FLAG
1987 007040  052705  000002                    BIS     #2,R5                   ;SET PGM CSR FLAG TO HI BYTE DONE
1988 007044  000404                            BR      10$                     ;BRANCH OVER NEXT STUFF
1989 007046  110037  002502            11$:    MOVB    R0,PGMCSR               ;SET PGM CSR TO LO BYTE
1990 007052  052705  000001                    BIS     #1,R5                   ;SET INTERLEAVE PGM CSR FLAG
1991 007056  053737  002630  002624    10$:    BIS     CONFIG+4,CONFIG         ;SET UP INFORMATION IN BANK ZERO
1992 007064  053737  002632  002626            BIS     CONFIG+6,CONFIG+2
1993 007072  000240                            NOP
1994 007074  022737  000001  002422            CMP     #1,I                    ;WAS THIS NON-INTERLEAVED MEMORY
1995 007102  001002                            BNE     46$                     ;NO - BRANCH OVER NEXT STMT.
1996 007104  000137  007474                    JMP     6$                      ;YES - JUMP TO END OF THIS BANK
1997 007110  012704  000020            46$:    MOV     #20,R4                  ;SET UP COUNTER FOR 64K INTERLEAVED
1998 007114  032737  000001  002422            BIT     #BIT0,I                 ;WAS IT 128K INTERLEAVED?
1999 007122  001402                            BEQ     26$                     ;BRANCH IF NOT
2000 007124  062704  000020                    ADD     #20,R4                  ;SET UP COUNTER FOR 128K INTERLEAVED
2001 007130  053764  002624  002624    26$:    BIS     CONFIG,CONFIG(R4)       ;SET OTHER BANK WITH SAME INFORMATION
2002 007136  053764  002626  002626            BIS     CONFIG+2,CONFIG+2(R4)   ;AS IN BANK 0
2003 007144  052764  000040  002624            BIS     #BIT5,CONFIG(R4)        ;SET SKIP ECC LOGIC TESTS FLAG
2004 007152  000470                            BR      33$                     ;BRANCH
2005                                           ;*
2006                                           ;* IF CHECKBITS DID NOT MATCH, WE COME HERE
2007                                           ;*
2008 007154  032737  100020  002144    22$:    BIT     #BIT15!BIT4,CSR         ;SBE OR DBE FLAGS SET?
2009 007162  001001                            BNE     8$                      ;BRANCH IF TRUE
2010 007164  000463                            BR      33$                     ;CHECK TO SEE IF IT IS MS11-M
2011 007166  013704  002146            8$:     MOV     CSRNO,R4                ;GET CSRNO
2012 007172  042764  000006  172100            BIC     #6,CSRADD(R4)           ;TURN OFF DIAG CHECK & ECC DISABLE
2013 007200                                    PUSH    R0,R1                   ;SAVE R0 & R1
2014 007204  016401  172100                    MOV     CSRADD(R4),R1           ;GET CSR INFORMATION
2015 007210  072127  177773                    ASH     #-5,R1                  ;SET UP ERROR ADDRESS
2016 007214  042701  177600                    BIC     #^C177,R1
2017 007220  005737  002424                    TST     NO22BIT                 ;IS THIS AN 11/44
2018 007224  001015                            BNE     27$                     ;BRANCH IF NOT
2019 007226  052764  040000  172100            BIS     #BIT14,CSRADD(R4)       ;GET EXTENDED ERROR ADDRESS BITS
2020 007234  016400  172100                    MOV     CSRADD(R4),R0           ;READ FROM CSR
2021 007240  042764  040000  172100            BIC     #BIT14,CSRADD(R4)       ;TURN OFF EUB BIT
2022 007246  042700  177037                    BIC     #^C740,R0               ;SET UP EXTENDED BITS
2023 007252  006300                            ASL     R0
2024 007254  006300                            ASL     R0
2025 007256  060001                            ADD     R0,R1                   ;SET UP TOTAL ERROR ADDRESS
2026 007260  010104                    27$:    MOV     R1,R4                   ;SAVE IN R4
2027 007262                                    POP     R1,R0                   ;RESTORE R0 & R1
2028 007266  072427  000005                    ASH     #5,R4                   ;SET ERROR ADDRESS UP IN PAR NOTATION
2029 007272  020437  172350                    CMP     R4,KIPAR4               ;DOES IT EQUAL KIPAR4?
2030 007276  001001                            BNE     28$                     ;BRANCH IF FALSE
2031 007300  000403                            BR      35$                     ;YES - MARK INFO IN CONFIG TABLE
2032 007302  020437  172352            28$:    CMP     R4,KIPAR5               ;DOES IT EQUAL KIPAR5?
2033 007306  001012                            BNE     33$                     ;BRANCH IF FALSE
2034 007310  052762  000001  002624    35$:    BIS     #BIT0,CONFIG(R2)        ;SET BANK ERROR FLAG
2035 007316  105262  002626                    INCB    CONFIG+2(R2)            ;INCREMENT BANK ERROR COUNTER
2036 007322                                    SET     CONFGERROR              ;PRINT CONFIG TABLE
2037 007330  000137  006554                    JMP     25$                     ;YES - MARK INFO IN CONFIG TABLE
2038                                           ;*
2039                                           ;* THIS SECTION SETS UP ALL THE POSSIBLE CONFIGURATIONS OF
2040                                           ;* MS11-M MEMORY.
2041                                           ;*
```

```
2042 007334  032760  000003  002432  33$:    BIT     #BIT0.BIT1.CSRINFO(R0)  ;IS THIS MS11-M MEMORY?
2043 007342  001054                          BNE     6$                      ;NO - GO TO END OF BANK
2044 007344  032760  000004  002432          BIT     #BIT2,CSRINFO(R0)
2045 007352  001450                          BEQ     6$
2046 007354  022737  000001  002422          CMP     #1,I                    ;IS THIS 1ST TIME THROUGH?
2047 007362  103410                          BLO     18$                     ;BRANCH IF NOT
2048 007364  162737  000002  002364          SUB     #2,TESTADD+2            ;TRY AS 64K INTERLEAVED
2049 007372  062737  004000  172352          ADD     #4000,KIPAR5            ;A1 NON-ASSERTED MEMORY
2050 007400  000137  006352                  JMP     4$                      ;TRY TO MATCH AGAIN
2051 007404  022737  000004  002422  18$:    CMP     #4,I                    ;4TH TIME THROUGH?
2052 007412  001404                          BEQ     20$                     ;YES - BRANCH
2053 007414  022737  000002  002422          CMP     #2,I                    ;2ND TIME THROUGH
2054 007422  103405                          BLO     12$                     ;NO - BRANCH
2055 007424  062737  004000  172352  20$:    ADD     #4000,KIPAR5            ;TRY AS 128K INTERLEAVED
2056 007432  000137  006352                  JMP     4$                      ;TRY TO MATCH AGAIN
2057 007436  022737  000003  002422  12$:    CMP     #3,I                    ;THIRD TIME THROUGH?
2058 007444  103413                          BLO     6$                      ;NO - BRANCH
2059 007446  062737  000002  002362          ADD     #2,TESTADD              ;TRY TESTING THE BANK
2060 007454  062737  000002  002364          ADD     #2,TESTADD+2            ;AS A1 ASSERTED
2061 007462  162737  004000  172352          SUB     #4000,KIPAR5            ;64K INTERLEAVED MEMORY
2062 007470  000137  006352                  JMP     4$                      ;TRY TO MATCH AGAIN
2063                                         ;*
2064                                         ;*END OF BANK ROUTINE
2065                                         ;*
2066 007474  104503                  6$:     CLR1CSR                         ;CLEAR THE CSR UNDER TEST
2067 007476  062702  000004                  ADD     #4,R2                   ;UPDATE CONFIGURATION POINTER
2068 007502  062737  001000  172350          ADD     #1000,KIPAR4            ;UPDATE KIPAR4 TO NEXT BANK
2069 007510  013737  172350  172352          MOV     KIPAR4,KIPAR5           ;AND UPDATE KIPAR5
2070 007516  000240                          NOP
2071 007520  023737  002530  172350          CMP     LASTBLOCK,KIPAR4        ;HAVE WE DONE THE WHOLE MEMORY SPACE?
2072 007526  002002                          BGE     19$                     ;BRANCH IF DONE
2073 007530  000137  006316                  JMP     45$                     ;JUMP IF NOT DONE
2074 007534  062700  000002          19$:    ADD     #2,R0                   ;INCREMENT CSR POINTER
2075 007540  000240                          NOP
2076 007542  104423                          CACHON                          ;TURN ON THE CACHE
2077 007544  000137  006166                  JMP     1$                      ;JUMP TO TRY NEXT CSR
```

```
                                SUBAAS: CACHON          ;MAKE SURE THE CACHE IS ON
                                        ECCINIT         ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
                                        NEWTST  <<TEST BANK 0 ACCESSES>>
                                ;*****************************************************************************
                                ;*TEST 2         TEST BANK 0 ACCESSES
                                ;*****************************************************************************
     007554  000004             TST2:   SCOPE
                                        ;THIS DOES A 'TST' INSTRUCTION ON EVERY LOCATION IN BANK #0 TO SEE
                                        ;IF IT GETS ANY PARITY TRAPS.
                                        ;SINCE EVERY LOCATION IS EITHER LOADED OR WRITTEN INTO BY THE PROGRAM
                                        ;PRIOR TO THIS POINT - THEN A PARITY ERROR IMPLIES THAT THERE IS A
                                        ;HARDWARE FAILURE IN THE MEMORY.
                                        ;THESE ERRORS ARE COUNTED AND A FATAL ACTION IS TAKEN
 007556  005037  002070               CLR     PARCNT                  ;CLEAR PARITY ERROR COUNTER
 007562  012737  000001  002074       MOV     #1,NOPAR                ;SET THE NO PARITY ERROR FLAG
 007570  005037  002066               CLR     NEMCNT                  ;CLEAR NON-EXISTANT MEMORY ERROR COUNTER
 007574  012737  000001  002076       MOV     #1,NONEM                ;SET THE NON-EXISTANT MFMORY ERROR MODE TO COUNT
 007602                               SET4    #NONEXIST               ;TRAPS TO 4 GOTO NONEXIST
 007610  005000                       CLR     R0
 007612  012701  040000               MOV     #SIZE,R1
 007616  104424                       CACHOFF                         ;TURN CACHE OFF
 007620  005720             1$:       TST     (R0)+                   ;SEE IF I CAN DO A READ ACCESS WITHOUT A PARITY TRAP
 007622  077102                       SOB     R1,1$
 007624  104423                       CACHON                          ;TURN CACHE ON
                                      ;SEE IF ANY FAILURES
 007626  005737  002070               TST     PARCNT                  ;ANY PARITY ERRORS?
 007632  001403                       BEQ     2$                      ;NO - SKIP
 007634                               FATAL   3
 007642  005737  002066       2$:     TST     NEMCNT                  ;ANY NON-EXISTANT MEMORY (HOLES)?
 007646  001406                       BEQ     3$                      ;SKIP IF EQUAL
 007650  162737  000002  002032       SUB     #2,ADDRESS              ;UPDATE 1ST ADDRESS FAILURE FROM AUTO INCREMENT #
 007656                               FATAL   4
 007664  053737  002104  002624 3$:   BIS     CPUBIT,CONFIG           ;SET CORRECT ACCESSED BIT ON BANK 0
 007672                               RES4                            ;RESET TRAPS TO 4 TO DEFAULT

 007712                               SUBTST  <<ENABLE ECC FOR CORRECT TRAPS>>
                                ;*****************************************************************************
                                ;*SUBTEST        ENABLE ECC FOR CORRECT TRAPS
                                ;*****************************************************************************
 007712                               IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE
 007730  104506                         ENASBE                        ;TRAP ON SINGLE BIT ERRORS
 007732                               ELSE
 007734  104472                         ECCINIT                       ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
 007736                               END ;OF IF #SW0
```

```
 2118 007736                                    NEWTST  <<TEST BANKS 1-200 (OCTAL) FOR ZEROS & ONES>>
                                        ;*****************************************************************************
                                        ;*TEST 3         TEST BANKS 1-200 (OCTAL) FOR ZEROS & ONES
                                        ;*****************************************************************************
      007736  000004                    TST3:   SCOPE
 2119                                           ;EACH BANK IS TESTED FOR EXISTANCE AND IF IT EXISTS
 2120                                           ;THEN IS IS TESTED FOR ZEROS & ONES.
 2121                                           ;EXCEPT -
 2122                                           ;       PROTECTED BANKS (WHERE THE PROGRAM IS) ARE ONLY TESTED BY
 2123                                           ;       "TST" INSTRUCTIONS LIKE BANK #0
 2124                                           ;ANY BAD BANKS ARE LOGGED IN THE CONFIGURATION TABLE.
 2125                                           ;THIS ROUTINE IS ONLY DOING A SMART SIZE - NOT ACTUAL TESTING!
 2126 007740  005037  002100                    CLR     BANK
 2127 007744  012737  000001  002074            MOV     #1,NOPAR                ;SET NO PARITY ERROR FLAG
 2128 007752  012737  000002  002076            MOV     #2,NONEM                ;SET NON-EXISTANT MEMORY MODE TO EXIT TEST LOOP
 2129 007760                                    SET4    #NONEXIST       ;TRAPS TO 4 GOTO NONEXIST
 2130 007766  005737  002424                    TST     NO22BIT         ;IS THIS AN 11/44?
 2131 007772  001407                            BEQ     1$              ;BRANCH IF TRUE
 2132 007774  012737  010564  002472            MOV     #MTST3+4,LINK1  ;SET UP LINKS
 2133 010002  012737  010566  002474            MOV     #MTST3+6,LINK2
 2134 010010  000411                            BR      TAG9$
 2135 010012                            1$:     BMOV    MTST3                   ;PUT IN FAST MEMORY
 2136 010020  012737  177644  002472            MOV     #UIPAR2,LINK1   ;SET UP LINKS
 2137 010026  012737  177646  002474            MOV     #UIPAR3,LINK2
 2138 010034  005237  002100            TAG9$:  INC     BANK
 2139 010040  023737  002526  002100            CMP     LASTBANK,BANK           ;DONE?
 2140 010046  103456                            BLO     TAG2$                   ;YES - SKIP TO NEXT TEST
 2141 010050  013701  002100                    MOV     BANK,R1
 2142 010054  006301                            ASL     R1
 2143 010056  006301                            ASL     R1              ;BANK * 4
 2144 010060  010137  002102                    MOV     R1,BANKINDEX
 2145 010064  005037  002072                    CLR     PATERR                  ;CLEAR PATTERN ERROR COUNTER
 2146 010070  005037  002070                    CLR     PARCNT                  ;CLEAR PARITY ERROR COUNTER
 2147 010074  005037  002066                    CLR     NEMCNT                  ;CLEAR NON-EXISTANT MEMORY COUNTER (HOLES)
 2148 010100                                    MAP     BANK                    ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
 2149 010114  105761  002624                    TSTB    CONFIG(R1)              ;IS THIS BANK PROTECTED?
 2150 010120  100552                            BMI     TSTBANK                 ;YES - GO TEST BANK SPECIAL
 2151 010122  012777  000207  172342    WARN1:  MOV     #207,@LINK1             ;PUT "RETURN" INSTRUCTION AFTER WRITE ROUTINE
 2152 010130  012700  060000                    MOV     #FIRST,R0
 2153 010134  010004                            MOV     R0,R4
 2154 010136  012701  040000                    MOV     #SIZE,R1
 2155 010142  010103                            MOV     R1,R3
 2156 010144  005002                            CLR     R2                      ;DATA IS ZEROS
 2157 010146  104424                            CACHOFF                 ;TURN CACHE OFF
 2158 010150                                    TESTAREA                        ;ENTER SUPERVISOR MODE
 2159 010156  005737  002424                    TST     NO22BIT         ;IS THIS AN 11/44?
 2160 010162  001403                            BEQ     1$              ;BRANCH IF TRUE
 2161 010164  004737  010560                    CALL    MTST3
 2162 010170  000402                            BR      2$
 2163 010172  004737  177640            1$:     CALL    FASTCITY                ;CALL TO THE USER INSTRUCTION PAR'S
 2164 010176  104417                    2$:     KERNEL                  ;ENTER KERNEL MODE
 2165 010200  104423                            CACHON                  ;TURN CACHE ON
 2166 010202  000415                            BR      TAG3$                   ;SKIP NEXT INSTRUCTION
 2167 010204  005037  002100            TAG2$:  CLR     BANK
 2168 010210                                    RES4                    ;RESET TRAPS TO 4 TO DEFAULT
 2169 010230  005037  002074                    CLR     NOPAR                   ;INDICATE DEFAULT PARITY ACTION
 2170 010234  000563                            BR      SUBAAI
```

```
2171 010236 005737 002066          TAG3$:  TST     NEMCNT                  ;ANY TRAPS?
2172 010242 001401                         BEQ     1$                      ;NO - SKIP
2173 010244 000673                         BR      TAG9$                   ;NOW - TRY NEXT BANK
2174 010246 104424                 1$:     CACHOFF                 ;TURN CACHE OFF
2175 010250                                TESTAREA                        ;ENTER SUPERVISOR MODE
2176 010256 004777 172212                  CALL    @LINK2                  ;FINISH PATTERN
2177 010262 104417                         KERNEL          ;ENTER KERNEL MODE
2178 010264 104423                         CACHON                  ;TURN CACHE ON
2179 010266 005737 002072                  TST     PATERR                  ;ANY PATTERN ERRORS
2180 010272 001037                         BNE     2$                      ;YES - SKIP
2181 010274 005737 002070                  TST     PARCNT                  ;ANY PARITY ERRORS
2182 010300 001034                         BNE     2$                      ;YES - SKIP
2183 010302 005737 002066                  TST     NEMCNT                  ;ANY NON EXISTANT MEMORY
2184 010306 001031                         BNE     2$                      ;YES - SKIP
2185 010310 012700 060000                  MOV     #FIRST,R0
2186 010314 010004                         MOV     R0,R4
2187 010316 012701 040000                  MOV     #SIZE,R1
2188 010322 010103                         MOV     R1,R3
2189 010324 013702 002554                  MOV     ONES,R2                 ;DATA IS ONES
2190 010330 012777 000240 172134           MOV     #000240,@LINK1          ;PUT 'NOP'' INSTRUCTION BACK IN SUBROUTINE
2191 010336 104424                         CACHOFF                 ;TURN CACHE OFF
2192 010340                                TESTAREA                        ;ENTER TEST MODE
2193 010346 005737 002424                  TST     NO22BIT                 ;IS THIS AN 11/44?
2194 010352 001403                         BEQ     5$                      ;BRANCH IF IT IS
2195 010354 004737 010560                  CALL    MTST3                   ;DO IN MEMORY IF NOT
2196 010360 000402                         BR      6$                      ;JUMP OVER NEXT INSTRUCTION
2197 010362 004737 177640          5$:     CALL    FASTCITY                ;CALL TO THE USER INSTRUCTION PAR'S
2198 010366 104417                 6$:     KERNEL          ;ENTER KERNEL MODE
2199 010370 104423                         CACHON                  ;TURN CACHE ON
2200 010372 013700 002102          2$:     MOV     BANKINDEX,R0
2201 010376 005737 002072                  TST     PATERR                  ;ANY PATTERN ERRORS?
2202 010402 001006                         BNE     3$                      ;YES - SKIP
2203 010404 005737 002070                  TST     PARCNT                  ;ANY PARITY ERRORS?
2204 010410 001003                         BNE     3$                      ;YES - SKIP
2205 010412 005737 002066                  TST     NEMCNT                  ;ANY HOLES?
2206 010416 001406                         BEQ     4$                      ;NONE - SKIP
2207 010420 052760 000001 002624   3$:     BIS     #BIT0,CONFIG(R0)        ;SET ERROR BIT IN THIS BANK
2208 010426                                SET     CONFGERROR              ;FORCE PRINTING OF CONFIGURATION TABLE
2209 010434 053760 002104 002624   4$:     BIS     CPUBIT,CONFIG(R0)       ;SET ACCESSED BIT
2210 010442 000137 010034                  JMP     TAG9$
2211
2212                                       ;TEST A PROTECTED BANK
2213 010446                        TSTBANK:PUSH    R1
2214 010450 012737 000001 002076           MOV     #1,NONEM                ;SET NON-EXISTANT MEMORY TO COUNT
2215 010456 012700 060000                  MOV     #FIRST,R0
2216 010462 012701 020000                  MOV     #20000,R1
2217 010466 104424                         CACHOFF                 ;TURN CACHE OFF
2218 010470                                TESTAREA                        ;ENTER TEST MODE
2219 010476 005720                 4$:     TST     (R0)+
2220 010500 077102                         SOB     R1,4$
2221 010502 104417                         KERNEL                  ;ENTER KERNEL MODE
2222 010504 104423                         CACHON                  ;TURN CACHE ON
2223 010506 012737 000002 002076           MOV     #2,NONEM                ;RESET NON-EXISTANT MEMORY TO EXIT TEST LOOP
2224 010514                                POP     R1
2225 010516                                IF PARCNT NE #0
2226 010524 052761 000001 002624             BIS   #BIT0,CONFIG(R1)        ;ERROR BANK
2227 010532                                  SET   CONFGERROR
```

```
                                                    END ;OF IF PARCNT
                                                    IF NEMCNT EQ #0
2230 010546  053761  002104  002624                    BIS   CPUBIT,CONFIG(R1)      ;ACCESSED BANK
                                                    END ;OF IF NEMCNT
2232 010554  000137  010034                         JMP   TAG9$
2233 010560  010220                        MTST3:   MOV   R2,(R0)+               ;V177640
2234 010562  077102                                 SOB   R1,MTST3                ;V177642
2235 010564  000240                                 NOP                           ;V177644
2236 010566  012401                        2$:      MOV   (R4)+,R1               ;V177646
2237 010570  020102                                 CMP   R1,R2                   ;V177650
2238 010572  001402                                 BEQ   3$                      ;V177652
2239 010574  005237  002072                         INC   PATERR                  ;V177654
2240 010600  077306                        3$:      SOB   R3,2$                   ;V177660
2241 010602  000207                                 RETURN                        ;V177662
```

```
2243 010604                          SUBAAI: SUBTST  <<FIND SHADOW INHIBIT MODE POINTERS>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       FIND SHADOW INHIBIT MODE POINTERS
                                     ;*****************************************************************************
2244                                 ;* THIS SECTION LOOKS FOR INTERLEAVED MS11-M MEMORIES AND FIGURES OUT
2245                                 ;* WHERE THE SHADOW INHIBIT MODE POINTERS ARE LOCATED.  THESE AREAS
2246                                 ;* ARE THEN MARKED AS PROGRAM SPACE.
2247 010604  005037  002100                CLR     BANK                    ;RESET BANK TO ZERO
2248 010610  004737  042700          SHADL1: CALL    EXBANK                  ;SET BANK PARAMETERS
2249 010614  013700  002102                MOV     BANKINDEX,R0
2250 010620                                IF ACFLAG IS TRUE AND INTFLAG IS TRUE
2251 010634                                  IF INT64K IS TRUE
2252 010642  062700  000020                    ADD #20,R0                  ;POINT TO BANKINDEX + 4
2253 010646  062737  000010  002100            ADD #10,BANK                ;POINT TO BANK + 8
2254 010654                                  ELSE
2255 010656  062702  000040                    ADD #40,R2                  ;POINT TO BANKINDEX + 8
2256 010662  062737  000020  002100            ADD #20,BANK                ;POINT TO BANK + 16
2257 010670                                  END; OF IF INT64K
2258 010670  052760  000200  002624          BIS   #BIT7,CONFIG(R0)        ;MAKE NEW BANK PROGRAM SPACE
2259 010676                                ELSE
2260 010700  005237  002100                  INC   BANK                    ;GO TO NEXT BANK
2261 010704                                END; OF IF ACFLAG
2262 010704  023737  002526  002100        CMP     LASTBANK,BANK           ;HAVE WE DONE ALL THE BANKS?
2263 010712  002336                        BGE     SHADL1                  ;BRANCH IF NOT
```

```
 2266 010714                            NEWTST  <<ECC INHIBIT MODE POINTER TEST>>
                                ;*****************************************************************************
                                ;*TEST 4        ECC INHIBIT MODE POINTER TEST
                                ;*****************************************************************************
      010714  000004            TST4:   SCOPE
 2267                                   ;THE MS11-M OR MF11S-K INHIBIT ECC DISABLE AND DIAGNOSTIC CHECK MODE
 2268                                   ;ON THE BOTTOM  FIRST OR SECOND 16K WORDS CONTROLLED BY A CSR.  THIS
 2269                                   ;IS CONSIDERED  TO BE A  PROTECTED BANK  BY THE  PROGRAM.  IT MAY BE
 2270                                   ;QUITE  COMPLEX TO DETERMINE ON A  GIVEN SYSTEM  CONFIGURATION WHICH
 2271                                   ;BANKS CAN BE PROTECTED;
 2272                                   ;SO
 2273                                   ;THIS ROUTINE ATTEMPS TO  CREATE A DOUBLE BIT ERROR IN ADDRESS 0 & 2
 2274                                   ;OF  EVERY ECC BANK.  ECC HARDWARE WILL PREVENT THIS  FROM HAPPENING
 2275                                   ;IN  PROTECTED BANKS  WHICH SHOULD ALWAYS  INCLUDE BANK ZERO - WHERE
 2276                                   ;THE PROGRAM IS.
 2277                                   ;
 2278                                   ;
 2279                                   ;WARNING:!!!.!!.!.!
 2280                                   ;  IN CASE OF HARDWARE FAILURE IT IS COMMON  THAT A DOUBLE BIT ERROR
 2281                                   ;  WILL BE  CREATED ON  THE KERNEL  STACK & ''CRASH''  THE  DIAGNOSTIC
 2282                                   ;DURING  THIS ROUTINE.  YOUR ONLY CLUE IS THAT YOU CAN GET AS FAR AS
 2283                                   ;THIS ROUTINE BUT NOT PAST IT!
 2284 010716  104424                    CACHOFF                                 ;TURN CACHE OFF
 2285 010720  012737  177777  002150     MOV     #-1,OLDCSR
 2286 010726                            FOR BANK := #0 TO LASTBANK
 2287 010732  012701  060000              MOV   #FIRST,R1                       ;SET UP VIRT ADDR POINTER
 2288 010736  004737  042700              CALL  EXBANK
 2289 010742  013700  002102              MOV   BANKINDEX,R0
 2290 010746                              IF ACFLAG IS TRUE
 2291 010754                                IF MKFLAG IS TRUE
 2292 010762                                IF SKIPMK IS FALSE
 2293 010770                                  IF INTFLAG IS TRUE
 2294 010776  012703  040000                    MOV     #40000,R3              ;SET INDEX COUNTER
 2295 011002  012737  000001  002232            MOV     #1,SPLTCSR             ;MAP AS INTERLEAVED BANK
 2296 011010                                  ELSE
 2297 011012  012703  000002                    MOV     #2,R3                  ;SET INDEX COUNTER
 2298 011016                                  END; OF IF INTFLAG
 2299 011016  116002  002625                  MOVB    CONFIG+1(R0),R2
 2300 011022  006302                          ASL     R2
 2301 011024  042702  177741                  BIC     #^C36,R2
 2302 011030  010237  002146                  MOV     R2,CSRNO
 2303 011034                                  IF CSRNO NE OLDCSR
 2304 011044  013737  002146  002150            MOV     CSRNO,OLDCSR
 2305 011052                                    IF PFLAG IS FALSE
 2306 011060  052760  000100  002624              BIS   #BIT6,CONFIG(R0)
 2307 011066                                    END; OF IF PFLAG
 2308 011066  004737  011222                    CALL    IMPTEST
```

```
2310 011072                                   IF INTFLAG IS TRUE
2311 011100  116002  002625                     MOVB  CONFIG+1(R0),R2
2312 011104  072227  177775                     ASH   #-3,R2
2313 011110  042702  177741                     BIC   #^C36,R2
2314 011114  010237  002146                     MOV   R2,CSRNO
2315 011120  062701  000002                     ADD   #2,R1           ;FIX POINTER FOR A1 ASSERTED HALF
2316 011124  004737  011222                     CALL  IMPTEST
2317 011130  005037  002232                     CLR   SPLTCSR
2318 011134                                   END; OF IF INTFLAG
2319 011134                                 END; OF IF CSRNO
2320 011134                               END; OF IF SKIPMK
2321 011134                               END; OF IF MKFLAG
2322 011134                             END; OF IF ACFLAG
2323 011134                           END; OF FOR BANK
2324 011150                           MAP                     ;MAP TEST SPACE TO BANK 0
2325 011164  005037  002100           CLR     BANK
2326 011170                           IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE
2327 011206  104506                     ENASBE               ;TRAP ON SINGLE BIT ERRORS
2328 011210                           ELSE
2329 011212  104472                     ECCINIT              ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
2330 011214                           END; OF IF #SW0
2331 011214  104423                   CACHON                 ;TURN THE CACHE BACK ON
2332 011216  000137  011464           JMP     SUBAAR         ;JUMP OVER THE SUBROUTINE
```

```
2334 011222  005004                    IMPTEST:CLR     R4
2335 011224                                    MAP BANK                                ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
2336 011240  005005                            CLR     R5
2337 011242  012737  020000  002144            MOV     #BIT13,CSR
2338 011250                                    TESTAREA                                ;ENTER TEST MODE
2339 011256                                    PUSH    (R1)                            ;SAVE TEST LOCATION
2340 011260  060301                            ADD     R3,R1                   ;INDEX TO NEXT LOCATION
2341 011262                                    PUSH    (R1)                    ;SAVE TEST LOCATION
2342 011264  104505                            CHK1DIS                         ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
2343 011266  010411                            MOV     R4,(R1)                 ;WRITE CHECKBITS (ALL ZEROS)
2344 011270  160301                            SUB     R3,R1
2345 011272  010411                            MOV     R4,(R1)
2346 011274  104503                            CLR1CSR                                 ;CLEAR CSR
2347 011276  005711                            TST     (R1)                            ;READ CHECKBITS INTO REAL CSR
2348 011300  104501                            WAS1DBE                                 ;WAS THERE A DOUBLE BIT ERROR
2349
2350                                           ;THIS MAKES SURE THAT SBE'S DON'T LOOK LIKE PROTECTED AREAS
2351
2352 011302                                    ON.NOERROR ;1
2353 011304  012737  020000  002144              MOV   #BIT13,CSR
2354 011312  104505                              CHK1DIS                               ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
2355 011314  013711  002554                      MOV   ONES,(R1)
2356 011320  060301                              ADD   R3,R1
2357 011322  013711  002554                      MOV   ONES,(R1)
2358 011326  160301                              SUB   R3,R1
2359 011330  104503                              CLR1CSR                               ;CLEAR CSR
2360 011332  005711                              TST   (R1)
2361 011334  104501                              WAS1DBE                               ;WAS THERE A DOUBLE BIT ERROR
2362 011336                                      ON.NOERROR ;2
2363 011340  012737  027400  002144                MOV #27400,CSR
2364 011346  104505                                CHK1DIS                             ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
2365 011350  010411                                MOV R4,(R1)
2366 011352  060301                                ADD R3,R1                   ;ADD INDEX TO GET TO SECOND WORD
2367 011354  010411                                MOV R4,(R1)
2368 011356  160301                                SUB R3,R1                   ;SUBTRACT INDEX TO FIRST WORD
2369 011360  104503                                CLR1CSR                             ;CLEAR CSR
2370 011362  005711                                TST (R1)
2371 011364  104501                                WAS1DBE                             ;WAS THERE A DOUBLE BIT ERROR
2372 011366                                        ON.NOERROR ;3
2373 011370  012737  074000  002144                  MOV       #74000,CSR
2374 011376  104505                                  CHK1DIS                           ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
2375 011400  010411                                  MOV       R4,(R1)
2376 011402  060301                                  ADD       R3,R1           ;INDEX TO SECOND WORD
2377 011404  010411                                  MOV       R4,(R1)
2378 011406  104503                                  CLR1CSR                           ;CLEAR CSR
2379 011410  160301                                  SUB       R3,R1           ;GO BACK TO FIRST WORD
2380 011412  005711                                  TST       (R1)
2381 011414  104501                                  WAS1DBE                           ;WAS THERE A DOUBLE BIT ERROR
2382 011416                                        END ;OF ON.NOERROR ;3
2383 011416                                      END ;OF ON.NOERROR ;2
2384 011416                                    END ;OF ON.NOERROR ;1
2385 011416                                    ON.ERROR
2386 011420  005205                              INC   R5                              ;IDENTIFY AS BAD BANK
2387 011422                                    END ;OF ON.ERROR
2388 011422  104471                            ECC1DIS                                 ;DISABLE ERROR CORRECTION
2389 011424  010411                            MOV     R4,(R1)                 ;CLEAR OUT DOUBLE BIT ERROR.
2390 011426  060301                            ADD     R3,R1                   ;INDEX TO SECOND WORD
```

```
2391 011430  010411                        MOV     R4,(R1)             ;CLEAR OUT DOUBLE BIT ER, R!
2392 011432  104503                        CLR1CSR
2393 011434  005705                        TST     R5
2394 011436  001405                        BEQ     1$
2395 011440  050560  002624                BIS     R5,CONFIG(R0)
2396 011444  105260  002626                INCB    CONFIG+2(R0)
2397 011450  104036                        ERROR   +36
2398 011452                          1$:   POP     (R1)                ;RESTORE TEST LOCATION (2ND WORD)
2399 011454  160301                        SUB     R3,R1               ;GO BACK TO FIRST WORD
2400 011456                                POP     (R1)                ;RESTORE TEST LOCATION (1ST WORD)
2401 011460  104417                        KERNEL
2402 011462  000207                        RETURN
```

```
2746
2747 011464                    SUBAAR: SET     STOPOK          ;PROGRAM CAN NOW BE HALTED
```

```
2750 011472                                    SUBTST  <<LEGAL CONFIGURATION CHECK>>
                                       ;*****************************************************************************
                                       ;*SUBTEST       LEGAL CONFIGURATION CHECK
                                       ;*****************************************************************************
2751 011472  012700  000020            MOV     #16.,R0
2752 011476  012701  002432            MOV     #CSRINFO,R1
2753 011502  005021              1$:   CLR     (R1)+
2754 011504  077002                    SOB     R0,1$
2758 011506                              FOR BANK := #0 TO LASTBANK
2759 011512  004737  042700                CALL        EXBANK
2760 011516  013700  002102                MOV BANKINDEX,R0
2782
2783 011522                                IF ACFLAG IS TRUE
2784 011530  116003  002625                 MOVB       CONFIG+1(R0),R3
2785 011534  042703  177760                 BIC        #^C17,R3
2786 011540  006303                        ASL        R3
2787 011542  005263  002432                INC        CSRINFO(R3)
2788 011546                                IF MKFLAG IS TRUE
2789                                       ;MAKE SURE THAT EACH BANK HAS NO MORE THAN 2 CSRS
2790 011554                                  BEGIN   LEGALCSR
2791 011554                                    IF INTFLAG IS TRUE
2792 011562  116003  002625                     MOVB       CONFIG+1(R0),R3
2793 011566  010304                           MOV R3,R4
2794 011570  042703  177760                   BIC #^C17,R3
2795 011574  072427  177774                   ASH #-4,R4
2796 011600  042704  177760                   BIC #^C17,R4
2797 011604                                   IF R3 EQ R4
2798 011610  042760  014000  002626              BIC        #BIT11!BIT12,CONFIG+2(R0)
2799 011616  042760  170000  002624              BIC        #170000,CONFIG(R0)
2800 011624                                     LEAVE LEGALCSR
2801 011626                                   END; OF IF R3
2802 011626                                   IF KFLAG IS FALSE
2803 011634                                     LEAVE     LEGALCSR
2804 011636                                   END; OF IF KFLAG
2805 011636  006303                           ASL R3
2806 011640  006304                           ASL R4
2807 011642  005263  002432                   INC CSRINFO(R3)
2808 011646  005264  002432                   INC CSRINFO(R4)
2809 011652                                  ELSE
2810 011654                                    LEAVE       LEGALCSR
2811 011656                                  END; OF IF INTFLAG
2812 011656                                TYPE    MSG124                    ;# OF CSR'S IS WRONG
2813 011662                                TYPOCS  BANK,<TYPE BANK #>,3
2814 011672                                SET     CONFGERROR
2815 011700                                END     LEGALCSR
2816 011700                              END ;OF IF MKFLAG
2817 011700                            END ;OF IF ACFLAG
2818 011700                          END; OF FOR BANK
2819 011714                        PUSH    R5,R0                             ;SAVE CONTENTS OF R5, R0
2820 011720  005000                CLR     R0                                ;CLEAR REGISTERS
2821 011722  005001                CLR     R1
2822 011724  005005                CLR     R5
2823 011726  005037  002274        CLR     CHECK                             ;CLEAR ERROR INDICATOR
2824 011732  022761  000010  002432  2$:  CMP     #10,CSRINFO(R1)           ;IS CURRENT CSR <= 10
2825 011740  002043                BGE     5$                                ;BRANCH IF SO
2826 011742  022761  000020  002432       CMP     #20,CSRINFO(R1)           ;IS CURRENT CSR < 20
2827 011750  002003                BGE     3$                                ;BRANCH IF SO
```

```
2828 011752  004737  012102                CALL    ILLCSR                          ;CALL ERROR ROUTINE
2829 011756  000434                        BR      5$                              ;TRY NEXT CSR
2830 011760  016005  002624          3$:   MOV     CONFIG(R0),R5                   ;MOVE LOW WORD TO R5
2831 011764  032705  000002                BIT     #BIT1,R5                        ;DOES MEMORY EXIST HERE?
2832 011770  001415                        BEQ     4$                              ;BRANCH IF NOT
2833 011772  042705  170377                BIC     #^C7400,R5                      ;ISOLATE CSR NUMBER IN
2834 011776  072527  177771                ASH     #-7,R5                          ;REGISTER 5
2835 012002  020501                        CMP     R5,R1                           ;IS IT THE CURRENT CSR?
2836 012004  001007                        BNE     4$                              ;TRY NEXT WORD OF CONFIG IF NOT
2837 012006  032760  010000  002626        BIT     #BIT12,CONFIG+2(R0)             ;IS IT INTERLEAVED?
2838 012014  001003                        BNE     4$                              ;BRANCH IF SO
2839 012016  012737  000001  002274        MOV     #1,CHECK                        ;SET ERROR INDICATOR
2840 012024  062700  000004          4$:   ADD     #4,R0                           ;UPDATE CONFIG COUNTER
2841 012030  022700  000340                CMP     #340,R0                         ;CONFIG TABLE ALL DONE?
2842 012034  001351                        BNE     3$                              ;BRANCH IF NOT
2843 012036  005737  002274                TST     CHECK                           ;ERRORS FOUND?
2844 012042  001402                        BEQ     5$                              ;TRY NEXT CSR IF NOT
2845 012044  004737  012102                CALL    ILLCSR                          ;CALL ERROR ROUTINE
2846 012050  005000                  5$:   CLR     R0                              ;REINITIALIZE CONFIG COUNTER
2847 012052  005037  002274                CLR     CHECK                           ;CLEAR ERROR INDICATOR
2848 012056  062701  000002                ADD     #2,R1                           ;UPDATE CSR COUNTER
2849 012062  022701  000040                CMP     #40,R1                          ;ALL CSR'S DONE?
2850 012066  001321                        BNE     2$                              ;BRANCH IF NOT
2851 012070                                POP     R0,R5                           ;RESTORE REGISTERS
2852 012074  005037  002274                CLR     CHECK                           ;RESET CHECK
2856 012100  000421                        BR      SUBAAP          ;BRANCH TO NEXT SUBTEST
2857 012102  010102                ILLCSR: MOV     R1,R2           ;R2 HAS CSR NUMBER
2858 012104  006202                        ASR     R2              ;MAKE ACCEPTABLE FOR PRINTING
2859 012106  022702  0000'2                CMP     #10.,R2
2860 012112  100002                        BPL     1$
2861 012114  062702  000007                ADD     #7,R2
2862 012120  062702  000060          1$:   ADD     #60,R2
2863 012124  110237  077470                MOVB    R2,MSGA122      ;PUT NUMBER INTO ERROR MESSAGE
2864 012130                                TYPE    MSG122
2865 012134                                SET     CONFGERROR
2866 012142  000207                        RETURN
```

```
2869 012144                          SUBAAP: SUBTST  <<PRINT CONFIGURATION DETAILS>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       PRINT CONFIGURATION DETAILS
                                     ;*****************************************************************************
2870 012144                                  CLEAR   BSIZE,KSIZE,LSIZE,MSIZE,PSIZE
2871 012170  013702  002526                  MOV     LASTBANK,R2
2872 012174  006302                          ASL     R2
2873 012176  006302                          ASL     R2
2874 012200                                  FOR R1 := #0 TO R2 BY #4
2875 012202                                   IF CPUBIT SET.IN CONFIG(R1)
2876 012212                                     IF #BIT10 SET.IN CONFIG+2(R1)
2877 012222                                       IF #BIT8 SET.IN CONFIG+2(R1)
2878 012232                                         IF #BIT9 SET.IN CONFIG+2(R1)
2879 012242                                           LET PSIZE := PSIZE + #1
2880 012246                                         ELSE
2881 012250                                           LET KSIZE := KSIZE + #1
2882 012254                                         END;IF BIT9
2883 012254                                       ELSE
2884 012256                                         IF #BIT9 SET.IN CONFIG+2(R1)
2885 012266                                           LET LSIZE := LSIZE + #1
2886 012272                                         ELSE
2887 012274                                           LET MSIZE := MSIZE + #1
2888 012300                                         END; IF BIT9
2889 012300                                     END;IF BIT8
2890 012300                                   ELSE
2891 012302                                     IF #BIT9 SET.IN CONFIG+2(R1)
2892 012312                                       IF #BIT8 SET.IN CONFIG+2(R1)
2893 012322                                         LET BSIZE := BSIZE + #1
2894 012326                                       END; OF IF #BIT8
2895 012326                                     END; OF IF #BIT9
2896 012326                                   END;IF BIT10
2897 012326                                  END; OF IF CPUBIT
2898 012326                                 END ;OF FOR ALL BANKS IN TABLE
2899
2900 012336  005037  002422                  CLR     I
2901 012342                                  FOR R1 := #0 TO #10 BY #2
2902 012344  006361  002344                    ASL   BSIZE(R1)
2903 012350  006361  002344                    ASL   BSIZE(R1)
2904 012354  006361  002344                    ASL   BSIZE(R1)
2905 012360  006361  002344                    ASL   BSIZE(R1)                ;BSIZE(R1) := BSIZE(R1) * 16.
2906 012364  066137  002344  002422           ADD   BSIZE(R1),I              ;I <- I + BSIZE(R1)
2907 012372                                  END; FOR R1
2908 012404                                  FOR R1 := #0 TO #200 BY #4
2909 012406                                     IF CPUBIT SET.IN CONFIG(R1)
2910 012416                                       LET UNITOP := UNITOP + #1
2911 012422                                     END; OF IF CPUBIT
2912 012422                                  END; OF FOR R1
2913 012434  006337  002366                  ASL     UNITOP
2914 012440  006337  002366                  ASL     UNITOP
2915 012444  006337  002366                  ASL     UNITOP
2916 012450  006337  002366                  ASL     UNITOP                 ;UNITOP := UNITOP * 16.
2917 012454                                  IF I LT UNITOP THEN LET I := UNITOP
2918 012472                                  TYPE    $CRLF
2919 012476  005737  002344                  TST     BSIZE
2920 012502  001405                          BEQ     1$
2921 012504                                  TYPDEC  BSIZE
2922 012512                                  TYPE    MSG071
```

```
 2923 012516  005737  002346          1$:     TST     KSIZE
 2924 012522  001405                          BEQ     2$
 2925 012524                                  TYPDEC  KSIZE
 2926 012532                                  TYPE    MSG072
 2927 012536  005737  002350          2$:     TST     LSIZE
 2928 012542  001405                          BEQ     3$
 2929 012544                                  TYPDEC  LSIZE
 2930 012552                                  TYPE    MSG112
 2931 012556  005737  002352          3$:     TST     MSIZE
 2932 012562  001405                          BEQ     4$
 2933 012564                                  TYPDEC  MSIZE
 2934 012572                                  TYPE    MSG113
 2935 012576  005737  002354          4$:     TST     PSIZE
 2936 012602  001405                          BEQ     5$
 2937 012604                                  TYPDEC  PSIZE
 2938 012612                                  TYPE    MSG114
 2939 012616                          5$:     TYPDEC  I
 2940 012624                                  TYPE    MSG070
 2941 012630                                  IF #SW6 OFF.IN @SWR
 2942 012640  004737  035254                    CALL  PCONFIG
 2943 012644                                  END; OF IF #SW6
```

```
2946 012644                                     SUBTST  <<CHECK APT SIZING>>
                                               ;*****************************************************************************
                                               ;*SUBTEST       CHECK APT SIZING
                                               ;*****************************************************************************
2947 012644                                            IF APTFLAG IS TRUE AND APTSIZE IS TRUE
2948 012660  005037  002404                              CLR    TEMP
2949 012664  012700  061306                              MOV    #$MAMS1,R0
2950 012670                                              FOR R2 := #0 TO #4
2951 012672                                                IFB 1(R0) NE #0
2952 012700  111001                                          MOVB       (R0),R1
2953 012702  042701  177400                                  BIC        #177400,R1
2954 012706                                                  IF 2(R0) LT #0
2955 012714  000261                                            SEC
2956 012716                                                  ELSE
2957 012720  000241                                            CLC
2958 012722                                                  END ;OF IF 2(R0)
2959 012722  006101                                          ROL        R1
2960 012724  005201                                          INC        R1              ;TO COMPENSATE FOR 4 BANKS BEING (0-3)
2961 012726  006301                                          ASL        R1
2962 012730  006301                                          ASL        R1
2963 012732  006301                                          ASL        R1
2964 012734  006301                                          ASL        R1
2965 012736  163701  002404                                  SUB        TEMP,R1
2966 012742  010137  002404                                  MOV        R1,TEMP
2967 012746                                                  IFB 1(R0) EQ #3
2968 012756  060137  002372                                    ADD      R1,APTPAR
2969 012762                                                  END ;OF IFB 1(R0)
2970 012762                                                  IFB 1(R0) EQ #4
2971 012772  060137  002374                                    ADD      R1,APTECC
2972 012776                                                  END ;OF IFB 1(R0)
2973 012776  062700  000004                                  ADD        #4,R0
2974 013002                                                IEND ;OF IFB 1(R0)
2975 013002                                              END ;OF FOR R2
2976 013012                                              IF APTPAR NE LSIZE OR APTECC NE MSIZE
2977 013032  104046                                        ERROR        +46
2978 013034                                              END ;OF IF APTPAR
2990 013034                                     END ;OF IF APTFLAG
```

```
 2992 013034                            LOOP:   NEWTST  <<DIAGNOSTIC MODE DISPATCH ROUTINE>>
                                        ;*****************************************************************************
                                        ;*TEST 5         DIAGNOSTIC MODE DISPATCH ROUTINE
                                        ;*****************************************************************************
      013034  000004                    TST5:   SCOPE
 2993 013036  005037  002214                    CLR     CONTFLAG
 2994 013042  017700  167530                    MOV     @SWR,R0         ;GET SWITCHES
 2995 013046  042700  177761                    BIC     #^C16,R0        ;MASK TO ONLY MODE BITS
 2996 013052  004770  013062                    CALL    @DISPTBL(R0)    ;DISPATCH TO ROUTINE THROUGH NEXT TABLE
 2997 013056  000137  013102                    JMP     MEMDONE         ;GO TO NEXT TEST
 2998 013062  013546                    DISPTBL:BAFPAF  ;MODE 0;BANKS FORWARD, PATTERNS FORWARD
 2999 013064  013654                            BAFPAR  ;MODE 1;BANKS FORWARD, PATTERNS REVERSE
 3000 013066  013762                            BAWPAF  ;MODE 2;BANKS WORST FIRST, PATTERNS FORWARD
 3001 013070  014112                            BAWPAR  ;MODE 3;BANKS WORST FIRST, PATTERNS REVERSE
 3002 013072  014242                            PAFBAF  ;MODE 4;PATTERNS FORWARD, BANKS FORWARD
 3003 013074  014372                            PAFBAW  ;MODE 5;PATTERNS FORWARD, BANKS WORST FIRST
 3004 013076  014544                            PARBAF  ;MODE 6;PATTERNS REVERSE, BANKS FORWARD
 3005 013100  014674                            PARBAW  ;MODE 7;PATTERNS REVERSE, BANKS WORST FIRST
 3006
 3007 013102  004737  013446            MEMDONF:CALL    DOBACK                          ;CHECK BACKGROUND PATTERN
 3008 013106                                    IF SELONLY IS TRUE
 3009 013114  005137  002560                      COM   SOFTPAT                         ;COMPLIMENT BACKGROUND PATTERN
 3010 013120                                    END ;OF IF SELONLY
 3011
 3012 013120                                    NEWTST<<UNIQUE BANK TEST>>
                                        ;*****************************************************************************
                                        ;*TEST 6         UNIQUE BANK TEST
                                        ;*****************************************************************************
      013120  000004                    TST6:   SCOPE
 3013                                           ;MAKE SURE THAT EACH BANK CAN HAVE UNIQUE DATA
 3014                                           ;WRITE AND READ THE BANK NUMBER IN EACH BANK (EXCEPT WHERE THE PROGRAM IS)
 3015 013122                                    IF SELONLY IS FALSE
 3016 013130                                      SET   HEADER,MUT
 3017 013144  004737  022740                      CALL  MT0027
 3018 013150                                      SET   HEADER
 3019 013156  005037  002106                      CLR   MUT
 3020 013162                                    END ;OF IF SELONLY
 3021 013162  004737  013446                    CALL    DOBACK          ;RESTORE BACKROUND PATTERN
 3025
 3026 013166                            FLUSH:  SUBTST  <<FLUSH OUT DBE'S>>
                                        ;*****************************************************************************
                                        ;*SUBTEST        FLUSH OUT DBE'S
                                        ;*****************************************************************************
 3027 013166  004737  023412                    CALL    MT0030
```

```
                                        .SBTTL  END OF PASS ROUTINE
                                ;*******************************************************************************
                                ;*INCREMENT THE PASS NUMBER ($PASS)
                                ;*INDICATE END-OF-PROGRAM AFTER EACH PASSES THRU THE PROGRAM
                                ;*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
                                ;*IF THERES A MONITOR GO TO IT
                                ;*IF THERE ISN'T JUMP TO LOOP
013172  005037  002412          $EOP:   CLR     FSINFLAG
013176  012700  002626                  MOV     #CONFIG+2,R0    ;MOVE 2ND WORD OF CONFIG TO R0
013202  042710  020000          1$:     BIC     #BIT13,(R0)     ;CLEAR BACKGROUND VALID BIT
013206  062700  000004                  ADD     #4,R0           ;INCREMENT TO NEXT BANK
013212  020027  003620                  CMP     R0,#3620        ;DONE?
013216  003771                          BLE     1$              ;NO - BRANCH
013220  013737  002570  002014          MOV     $ERTTL,LASTERROR
013226  005237  061264                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
013232  042737  100000  061264          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
013240                                  TYPE    MSG077          ;;TYPE 'END PASS #'
013244                                  IF #SW11 SET.IN @SWR OR QVFLAG IS TRUE
013262                                    TYPE  MSG035          ;QV
013266  005037  002316                    CLR   QVFLAG
013272                                  END ;OF IF SW11
013272                                  TYPDEC  $PASS
013300  013700  000042                  MOV     42,R0           ;;GET MONITOR ADDRESS
013304  001456                          BEQ     $DOAGAIN        ;;BRANCH IF NO MONITOR
013306  022700  002000          $ZAP42: CMP     #STACK,R0       ;ARE WE UNDER RT11
013312  001453                          BEQ     $DOAGAIN        ;YES - BRANCH
                                        ;WE ARE UNDER (HEAVEN HELP US) XXDP!
013314                                  PUSH    R0
013316  004737  043566                  CALL    SHUTUP
013322                                  POP     R0
013324  000005                          RESET                   ;;CLEAR THE WORLD
013326  004710                  $ENDAD: CALL    (R0)            ;;GO TO MONITOR
013330  000240                          NOP                     ;;SAVE ROOM
013332  000240                          NOP                     ;;FOR
013334  000240                          NOP                     ;;ACT11
013336                          $DOAGN: ;UNDO SHUTUP STUFF
                                        ;       RESTORE STACK
                                        ;       ENERGIZE UNIBUS MAP & 22 BIT ADDRESSING
                                        ;       ENERGIZE MEMORY MANAGEMENT
                                        ;       PUT LOADERS BACK HOME
013336  013706  002534                  MOV     KSTACK,SP
013342  005737  002424                  TST     N022BIT
013346  001003                          BNE     1$
013350  052737  000060  172516          BIS     #BIT5!BIT4,MMR3
013356  104420                  1$:     ENERGIZE                ;TURN ON MEMORY MANAGEMENT
013360  013700  002536                  MOV     LOADHOME,R0             ;DESTINATION BANK
013364  012701  000001                  MOV     #1,R1                   ;SOURCE BANK
013370  004737  042350                  CALL    BANKMOV
013374                                  IF APTFLAG IS TRUE
013402                                    IF $USWR EQ $PASS
013412  012701  000050          APTHANG:    MOV #50,R1
013416  077001                  2$:         SOB R0,2$
013420  062737  000001  061266            ADD #1,$DEVCT
013426  005537  061270                    ADC $UNIT
013432  077107                            SOB R1,2$
013434  005237  061264                    INC $PASS
013440  000764                            BR  APTHANG
```

```
3087 013442                                    END ;OF IF $USWR
3088 013442                                  END ;OF IF APTFLAG
3089 013442  000137  013034          $DOAGAIN: JMP   LOOP              ;RETURN
```

```
3092 013446                          DOBACK: SUBTST  <<WRITE BACKGROUND PATTERNS>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       WRITE BACKGROUND PATTERNS
                                     ;*****************************************************************************
3093 013446  005037  002110                 CLR     PATTERN
3094 013452                                 FOR BANK := #0 TO LASTBANK
3095 013456  004737  042700                   CALL  EXBANK
3096 013462                                   IF ACFLAG IS TRUE AND RRFLAG IS FALSE
3097 013476                                     SET         HEADER,MUT
3098 013512  004737  016354                     CALL        MKTEST          ;CALL   MJTEST WOULD ALSO WORK
3099 013516  005037  002106                     CLR         MUT
3100 013522                                     SET         HEADER
3101 013530                                   END ;OF IF ACFLAG
3102 013530                                 END ;OF FOR BANK
3103 013544  000207                         RETURN
```

```
                                        .SBTTL  MTEST MODES
                                ;
013546                          BAFPAF: SUBTST  <<BANKS FORWARD,PATTERNS FORWARD      **RECURSIVE**>>
                                ;*******************************************************************************
                                ;*SUBTEST        BANKS FORWARD,PATTERNS FORWARD      **RECURSIVE**
                                ;*******************************************************************************
013546  005037  002100                CLR     BANK            ;SET BANK TO 0
                                        ;START OF BANK LOOP
013552  004737  042700          1$:   CALL    EXBANK          ;EXAMINE BANK
013556  005737  002114                TST     ACFLAG          ;CAN WE ACCESS THIS BANK?
013562  001412                        BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
013564  005737  002122                TST     RRFLAG          ;RELOCATION REQUIRED?
013570  001007                        BNE     4$              ;YES - GO TO BANK LOOP TERMINATION
013572  005037  002110                CLR     PATTERN         ;SET PATTERN TO 0
                                        ;START OF PATTERN LOOP
013576  004737  015046          2$:   CALL    MTEST           ;GO TEST CORRECT MEMORY
                                        ;TERMINATION OF PATTERN LOOP
013602  004737  043366                CALL    INCPAT          ;GO SEE IF THIS IS THE LAST PATTERN
013606  001373                        BNE     2$              ;NO - LOOP ON THIS PATTERN
                                        ;TERMINATION OF BANK LOOP
013610  005037  002214          4$:   CLR     CONTFLAG
013614  004737  043412                CALL    INCBNK          ;NEXT HIGHER BANK
013620  002354                        BGE     1$              ;IF NOT DONE - LOOP ON THIS BANK
                                        ;END OF LOOPS
013622  005737  002124                TST     RLFLAG          ;HAVE WE BEEN RELOCATED?
013626  001401                        BEQ     5$              ;NO - SKIP
013630  000207                        RETURN                  ;YES - RETURN
013632  004737  041154          5$:   CALL    RELOCATE        ;MOVE & MAP PROGRAM
013636                                ON.ERROR THEN $RETURN
                                        ;**NOTE** RECURSIVE CALL
013642  004737  013546                CALL    BAFPAF          ;CALL SELF
013646  004737  042042                CALL    UNRELOCATE      ;UNMOVE & UNMAP PROGRAM
013652  000207                        RETURN
```

```
3138 013654                        BAFPAR: SUBTST  <<BANKS FORWARD,PATTERNS REVERSE      **RECURSIVE**>>
                                   ;*****************************************************************************
                                   ;*SUBTEST       BANKS FORWARD,PATTERNS REVERSE      **RECURSIVE**
                                   ;*****************************************************************************
3139 013654  005037  002100                CLR     BANK                ;SET BANK TO 0
3140                                       ;START OF BANK LOOP
3141 013660  004737  042700          1$:   CALL    EXBANK              ;EXAMINE BANK
3142 013664  005737  002114                TST     ACFLAG              ;CAN WE ACCESS THIS BANK?
3143 013670  001412                        BEQ     4$                  ;NO - GO TO BANK LOOP TERMINATION
3144 013672  005737  002122                TST     RRFLAG              ;RELOCATION REQUIRED?
3145 013676  001007                        BNE     4$                  ;YES - GO TO BANK LOOP TERMINATION
3146 013700  004737  043402                CALL    SETPAT              ;SET HIGH PATTERN FOR CORRECT MEMORY
3147                                       ;START OF PATTERN LOOP
3148 013704  004737  015046          2$:   CALL    MTEST               ;GO TEST CORRECT MEMORY
3149                                       ;TERMINATION OF PATTERN LOOP
3150 013710  005337  002110                DEC     PATTERN             ;IS THIS THE LAST PATTERN?
3151 013714  100373                        BPL     2$                  ;NO - LOOP ON THIS PATTERN
3152                                       ;TERMINATION OF BANK LOOP
3153 013716  005037  002214          4$:   CLR     CONTFLAG
3154 013722  004737  043412                CALL    INCBNK              ;NEXT HIGHER BANK
3155 013726  002354                        BGE     1$                  ;IF NOT DONE - LOOP ON THIS BANK
3156                                       ;END OF LOOPS
3157 013730  005737  002124                TST     RLFLAG              ;HAVE WE BEEN RELOCATED?
3158 013734  001401                        BEQ     5$                  ;NO - SKIP
3159 013736  000207                        RETURN                      ;YES - RETURN
3160 013740  004737  041154          5$:   CALL    RELOCATE            ;MOVE & MAP PROGRAM
3161 013744                                ON.ERROR THEN $RETURN
3162                                       ;**NOTE** RECURSIVE CALL
3163 013750  004737  013654                CALL    BAFPAR              ;CALL SELF
3164 013754  004737  042042                CALL    UNRELOCATE          ;UNMOVE & UNMAP PROGRAM
3165 013760  000207                        RETURN
```

```
3168 013762                        BAWPAF: SUBTST  <<BANKS WORST FIRST,PATTERNS FORWARD  **RECURSIVE**>>
                                   ;*****************************************************************************
                                   ;*SUBTEST       BANKS WORST FIRST,PATTERNS FORWARD  **RECURSIVE**
                                   ;*****************************************************************************
3169 013762  005037  002100               CLR     BANK            ;SET BANK TO 0
3170                               ;START OF BANK LOOP
3171 013766  004737  042700        1$:     CALL    EXBANK          ;EXAMINE BANK
3172 013772  005737  002114               TST     ACFLAG          ;CAN WE ACCESS THIS BANK?
3173 013776  001415                       BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3174 014000  005737  002126               TST     BMFLAG          ;IS THIS BAD MEMORY (WORST FIRST)?
3175 014004  001412                       BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3176 014006  005737  002122               TST     RRFLAG          ;RELOCATION REQUIRED?
3177 014012  001007                       BNE     4$              ;YES - GO TO BANK LOOP TERMINATION
3178 014014  005037  002110               CLR     PATTERN         ;SET PATTERN TO 0
3179                               ;START OF PATTERN LOOP
3180 014020  004737  015046        2$:     CALL    MTEST           ;GO TEST CORRECT MEMORY
3181                               ;TERMINATION OF PATTERN LOOP
3182 014024  004737  043366               CALL    INCPAT          ;GO SEE IF THIS IS THE LAST PATTERN
3183 014030  001373                       BNE     2$              ;NO - LOOP ON THIS PATTERN
3184                               ;TERMINATION OF BANK LOOP
3185 014032  005037  002214        4$:     CLR     CONTFLAG
3186 014036  004737  043412               CALL    INCBNK          ;NEXT HIGHER BANK
3187 014042  002351                       BGE     1$              ;IF NOT DONE - LOOP ON THIS BANK
3188                               ;END OF LOOPS
3189 014044  005137  002540               COM     WORST           ;IS THIS AN EVEN NUMBERED PASS?
3190 014050  001003                       BNE     5$              ;YES - SKIP
3191                               ;**NOTE** RECURSIVE CALL
3192 014052  004737  013762               CALL    BAWPAF          ;CALL SELF
3193 014056  000207                       RETURN
3194 014060  005737  002124        5$:     TST     RLFLAG          ;HAVE WE BEEN RELOCATED?
3195 014064  001401                       BEQ     6$              ;NO - SKIP
3196 014066  000207                       RETURN                  ;YES - RETURN
3197 014070  004737  041154        6$:     CALL    RELOCATE        ;MOVE & MAP PROGRAM
3198 014074                               ON.ERROR THEN $RETURN
3199                               ;**NOTE** RECURSIVE CALL
3200 014100  004737  013762               CALL    BAWPAF          ;CALL SELF
3201 014104  004737  042042               CALL    UNRELOCATE      ;UNMOVE & UNMAP PROGRAM
3202 014110  000207                       RETURN
```

```
3205 014112                        BAWPAR: SUBTST  <<BANKS WORST FIRST,PATTERNS REVERSE  **RECURSIVE**>>
                                   ;*****************************************************************************
                                   ;*SUBTEST       BANKS WORST FIRST,PATTERNS REVERSE  **RECURSIVE**
                                   ;*****************************************************************************
3206 014112  005037  002100                CLR     BANK            ;SET BANK TO 0
3207                               ;START OF BANK LOOP
3208 014116  004737  042700        1$:     CALL    EXBANK          ;EXAMINE BANK
3209 014122  005737  002114                TST     ACFLAG          ;CAN WE ACCESS THIS BANK?
3210 014126  001415                        BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3211 014130  005737  002126                TST     BMFLAG          ;IS THIS BAD MEMORY (WORST FIRST)
3212 014134  001412                        BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3213 014136  005737  002122                TST     RRFLAG          ;RELOCATION REQUIRED?
3214 014142  001007                        BNE     4$              ;YES - GO TO BANK LOOP TERMINATION
3215 014144  004737  043402                CALL    SETPAT          ;SET HIGH PATTERN FOR CORRECT MEMORY
3216                               ;START OF PATTERN LOOP
3217 014150  004737  015046        2$:     CALL    MTEST           ;GO TEST CORRECT MEMORY
3218                               ;TERMINATION OF PATTERN LOOP
3219 014154  005337  002110                DEC     PATTERN         ;IS THIS THE LAST PATTERN?
3220 014160  100373                        BPL     2$              ;NO - LOOP ON THIS PATTERN
3221                               ;TERMINATION OF BANK LOOP
3222 014162  005037  002214        4$:     CLR     CONTFLAG
3223 014166  004737  043412                CALL    INCBNK          ;NEXT HIGHER BANK
3224 014172  002351                        BGE     1$              ;IF NOT DONE - LOOP ON THIS BANK
3225                               ;END OF LOOPS
3226 014174  005137  002540                COM     WORST           ;IS THIS AN EVEN NUMBERED PASS?
3227 014200  001003                        BNE     5$              ;YES - SKIP
3228                               ;**NOTE** RECURSIVE CALL
3229 014202  004737  014112                CALL    BAWPAR          ;CALL SELF
3230 014206  000207                        RETURN
3231 014210  005737  002124        5$:     TST     RLFLAG          ;HAVE WE BEEN RELOCATED?
3232 014214  001401                        BEQ     6$              ;NO - SKIP
3233 014216  000207                        RETURN                  ;YES - RETURN
3234 014220  004737  041154        6$:     CALL    RELOCATE        ;MOVE & MAP PROGRAM
3235 014224                                ON.ERROR THEN $RETURN
3236                               ;**NOTE** RECURSIVE CALL
3237 014230  004737  014112                CALL    BAWPAR          ;CALL SELF
3238 014234  004737  042042                CALL    UNRELOCATE      ;UNMOVE & UNMAP PROGRAM
3239 014240  000207                        RETURN
```

```
3242 014242                          PAFBAF: SUBTST  <<PATTERNS FORWARD,BANKS FORWARD      **RECURSIVE**>>
                                     ;*************************************************************************
                                     ;*SUBTEST       PATTERNS FORWARD,BANKS FORWARD      **RECURSIVE**
                                     ;*************************************************************************
3243 014242  005037  002110                     CLR     PATTERN          ;SET PATTERN TO 0
3244                                             ;START OF PATTERN LOOP
3245 014246  005037  002100          1$:        CLR     BANK             ;SET BANK TO 0
3246                                             ;START OF BANK LOOP
3247 014252  004737  042700          2$:        CALL    EXBANK           ;EXAMINE BANK
3248 014256  004737  043350                     CALL    BANKOK           ;CORRECT MEMORY FOR THIS BANK?
3249 014262  001010                             BNE     4$               ;NO - GO TO BANK LOOP TERMINATOR
3250 014264  005737  002114                     TST     ACFLAG           ;CAN WE ACCESS THIS BANK?
3251 014270  001405                             BEQ     4$               ;NO - GO TO BANK LOOP TERMINATION
3252 014272  005737  002122                     TST     RRFLAG           ;RELOCATION REQUIRED?
3253 014276  001002                             BNE     4$               ;YES - GO TO BANK LOOP TERMINATION
3254 014300  004737  015046                     CALL    MTEST            ;GO TEST CORRECT MEMORY
3255                                             ;TERMINATION OF BANK LOOP
3256 014304  005037  002214          4$:        CLR     CONTFLAG
3257 014310  004737  043412                     CALL    INCBNK           ;NEXT HIGHER BANK
3258 014314  002356                             BGE     2$               ;IF NOT DONE - LOOP ON THIS BANK
3259                                             ;TERMINATION OF PATTERN LOOP
3260 014316  004737  043366                     CALL    INCRPT           ;NEXT HIGHER PATTERN
3261 014322  001351                             BNE     1$               ;OK - LOOP; ELSE CONTINUE
3262                                             ;END OF LOOPS
3263 014324  005137  002132                     COM     TMFLAG           ;COMPLEMENT TYPE OF MEMORY
3264                                                                      ;IS THIS AN EVEN NUMBER PASS?
3265 014330  001403                             BEQ     5$               ;YES - SKIP
3266                                             ;**NOTE** RECURSIVE CALL
3267 014332  004737  014242                     CALL    PAFBAF           ;CALL SELF
3268 014336  000207                             RETURN
3269 014340  005737  002124          5$:        TST     RLFLAG           ;HAVE WE BEEN RELOCATED?
3270 014344  001401                             BEQ     6$               ;NO - SKIP
3271 014346  000207                             RETURN                   ;YES - RETURN
3272 014350  004737  041154          6$:        CALL    RELOCATE         ;MOVE & MAP PROGRAM
3273 014354                                     ON.ERROR THEN $RETURN
3274                                             ;**NOTE** RECURSIVE CALL
3275 014360  004737  014242                     CALL    PAFBAF           ;CALL SELF
3276 014364  004737  042042                     CALL    UNRELOCATE       ;UNMOVE & UNMAP PROGRAM
3277 014370  000207                             RETURN
```

```
3280 014372                        PAFBAW: SUBTST  <<PATTERNS FORWARD,BANKS WORST FIRST  **RECURSIVE**>>
                                   ;*************************************************************************
                                   ;*SUBTEST       PATTERNS FORWARD,BANKS WORST FIRST  **RECURSIVE**
                                   ;*************************************************************************
3281 014372  005037  002110               CLR     PATTERN          ;SET PATTERN TO 0
3282                               ;START OF PATTERN LOOP
3283 014376  005037  002100        1$:     CLR     BANK             ;SET BANK TO 0
3284                               ;START OF BANK LOOP
3285 014402  004737  042700        2$:     CALL    EXBANK           ;EXAMINE BANK
3286 014406  004737  043350               CALL    BANKOK           ;CORRECT MEMORY FOR THIS BANK?
3287 014412  001013                       BNE     4$               ;NO - GO TO BANK LOOP TERMINATOR
3288 014414  005737  002114               TST     ACFLAG           ;CAN WE ACCESS THIS BANK?
3289 014420  001410                       BEQ     4$               ;NO - GO TO BANK LOOP TERMINATION
3290 014422  005737  002126               TST     BMFLAG           ;IS THIS BAD MEMORY (WORST FIRST)
3291 014426  001405                       BEQ     4$               ;NO - GO TO BANK LOOP TERMINATION
3292 014430  005737  002122               TST     RRFLAG           ;RELOCATION REQUIRED?
3293 014434  001002                       BNE     4$               ;YES - GO TO BANK LOOP TERMINATION
3294 014436  004737  015046               CALL    MTEST            ;GO TEST CORRECT MEMORY
3295                               ;TERMINATION OF BANK LOOP
3296 014442  005037  002214        4$:     CLR     CONTFLAG
3297 014446  004737  043412               CALL    INCBNK           ;NEXT HIGHER BANK
3298 014452  002353                       BGE     2$               ;IF NOT DONE - LOOP ON THIS BANK
3299                               ;TERMINATION OF PATTERN LOOP
3300 014454  004737  043366               CALL    INCRPT           ;NEXT HIGHER PATTERN
3301 014460  001346                       BNE     1$               ;OK - LOOP; ELSE CONTINUE
3302                               ;END OF LOOPS
3303 014462  005137  002132               COM     TMFLAG           ;COMPLEMENT TYPE OF MEMORY
3304                                                                ;IS THIS AN EVEN NUMBER PASS?
3305 014466  001403                       BEQ     5$               ;YES - SKIP
3306                               ;**NOTE** RECURSIVE CALL
3307 014470  004737  014372               CALL    PAFBAW           ;CALL SELF
3308 014474  000207                       RETURN
3309 014476  005137  002540        5$:     COM     WORST            ;4TH PASS?
3310 014502  001003                       BNE     6$               ;YES - SKIP
3311                               ;**NOTE** RECURSIVE CALL
3312 014504  004737  014372               CALL    PAFBAW           ;CALL SELF
3313 014510  000207                       RETURN
3314 014512  005737  002124        6$:     TST     RLFLAG           ;HAVE WE BEEN RELOCATED?
3315 014516  001401                       BEQ     7$               ;NO - SKIP
3316 014520  000207                       RETURN                   ;YES - RETURN
3317 014522  004737  041154        7$:     CALL    RELOCATE         ;MOVE & MAP PROGRAM
3318 014526                               ON.ERROR THEN $RETURN
3319                               ;**NOTE** RECURSIVE CALL
3320 014532  004737  014372               CALL    PAFBAW           ;CALL SELF
3321 014536  004737  042042               CALL    UNRELOCATE       ;UNMOVE & UNMAP PROGRAM
3322 014542  000207                       RETURN
```

```
3325 014544                          PARBAF: SUBTST  <<PATTERNS REVERSE,BANKS FORWARD      **RECURSIVE**>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       PATTERNS REVERSE,BANKS FORWARD      **RECURSIVE**
                                     ;*****************************************************************************
3326 014544  004737  043402                  CALL    HIPAT           ;SET HIGHTEST PATTERNS
3327                                         ;START OF PATTERN LOOP
3328 014550  005037  002100          1$:     CLR     BANK            ;SET BANK TO 0
3329                                         ;START OF BANK LOOP
3330 014554  004737  042700          2$:     CALL    EXBANK          ;EXAMINE BANK
3331 014560  004737  043350                  CALL    BANKOK          ;CORRECT MEMORY FOR THIS BANK?
3332 014564  001010                          BNE     4$              ;NO - GO TO BANK LOOP TERMINATOR
3333 014566  005737  002114                  TST     ACFLAG          ;CAN WE ACCESS THIS BANK?
3334 014572  001405                          BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3335 014574  005737  002122                  TST     RRFLAG          ;RELOCATION REQUIRED?
3336 014600  001002                          BNE     4$              ;YES - GO TO BANK LOOP TERMINATION
3337 014602  004737  015046                  CALL    MTEST           ;GO TEST CORRECT MEMORY
3338                                         ;TERMINATION OF BANK LOOP
3339 014606  00'037  002214          4$:     CLR     CONTFLAG
3340 014612  004737  043412                  CALL    INCBNK          ;NEXT HIGHER BANK
3341 014616  002356                          BGE     2$              ;IF NOT DONE - LOOP ON THIS BANK
3342                                         ;TERMINATION OF PATTERN LOOP
3343 014620  005337  002110                  DEC     PATTERN         ;NEXT LOWER PATTERN
3344 014624  100351                          BPL     1$              ;OK - LOOP; ELSE CONTINUE
3345                                         ;END OF LOOPS
3346 014626  005137  002132                  COM     TMFLAG          ;COMPLEMENT TYPE OF MEMORY
3347                                                                 ;IS THIS AN EVEN NUMBER PASS?
3348 014632  001403                          BEQ     5$              ;YES - SKIP
3349                                         ;**NOTE** RECURSIVE CALL
3350 014634  004737  014544                  CALL    PARBAF          ;CALL SELF
3351 014640  000207                          RETURN
3352 014642  005737  002124          5$:     TST     RLFLAG          ;HAVE WE BEEN RELOCATED?
3353 014646  001401                          BEQ     6$              ;NO - SKIP
3354 014650  000207                          RETURN                  ;YES - RETURN
3355 014652  004737  041154          6$:     CALL    RELOCATE        ;MOVE & MAP PROGRAM
3356 014656                                  ON.ERROR THEN $RETURN
3357                                         ;**NOTE** RECURSIVE CALL
3358 014662  004737  014544                  CALL    PARBAF          ;CALL SELF
3359 014666  004737  042042                  CALL    UNRELOCATE      ;UNMOVE & UNMAP PROGRAM
3360 014672  000207                          RETURN
```

```
3363 014674                         PARBAW: SUBTST  <<PATTERNS REVERSE,BANKS WORST FIRST  **RECURSIVE**>>
                                    ;*************************************************************************
                                    ;*SUBTEST       PATTERNS REVERSE,BANKS WORST FIRST  **RECURSIVE**
                                    ;*************************************************************************
3364 014674  004737  043402                CALL    HIPAT           ;SET HIGHTEST PATTERN
3365                                       ;START OF PATTERN LOOP
3366 014700  005037  002100         1$:    CLR     BANK            ;SET BANK TO 0
3367                                       ;START OF BANK LOOP
3368 014704  004737  042700         2$:    CALL    EXBANK          ;EXAMINE BANK
3369 014710  004737  043350                CALL    BANKOK          ;CORRECT MEMORY FOR THIS BANK?
3370 014714  001013                       BNE     4$              ;NO - GO TO BANK LOOP TERMINATOR
3371 014716  005737  002114                TST     ACFLAG          ;CAN WE ACCESS THIS BANK?
3372 014722  001410                       BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3373 014724  005737  002126                TST     BMFLAG          ;IS THIS BAD MEMORY (WORST FIRST)
3374 014730  001405                       BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3375 014732  005737  002122                TST     RRFLAG          ;RELOCATION REQUIRED?
3376 014736  001002                       BNE     4$              ;YES - GO TO BANK LOOP TERMINATION
3377 014740  004737  015046                CALL    MTEST           ;GO TEST CORRECT MEMORY
3378                                       ;TERMINATION OF BANK LOOP
3379 014744  005037  002214         4$:    CLR CONTFLAG
3380 014750  004737  043412                CALL    INCBNK          ;NEXT HIGHER BANK
3381 014754  002353                       BGE     2$              ;IF NOT DONE - LOOP ON THIS BANK
3382                                       ;TERMINATION OF PATTERN LOOP
3383 014756  005337  002110                DEC     PATTERN         ;NEXT LOWER PATTERN
3384 014762  100346                       BPL     1$              ;OK - LOOP; ELSE CONTINUE
3385                                       ;END OF LOOPS
3386 014764  005137  002132                COM     TMFLAG          ;COMPLEMENT TYPE OF MEMORY
3387                                                               ;IS THIS AN EVEN NUMBER PASS?
3388 014770  001403                       BEQ     5$              ;YES - SKIP
3389                                       ;**NOTE** RECURSIVE CALL
3390 014772  004737  014674                CALL    PARBAW          ;CALL SELF
3391 014776  000207                       RETURN
3392 015000  005137  002540         5$:    COM     WORST           ;4TH PASS?
3393 015004  001003                       BNE     6$              ;YES - SKIP
3394                                       ;**NOTE** RECURSIVE CALL
3395 015006  004737  014674                CALL    PARBAW          ;CALL SELF
3396 015012  000207                       RETURN
3397 015014  005737  002124         6$:    TST     RLFLAG          ;HAVE WE BEEN RELOCATED?
3398 015020  001401                       BEQ     7$              ;NO - SKIP
3399 015022  000207                       RETURN                  ;YES - RETURN
3400 015024  004737  041154         7$:    CALL    RELOCATE        ;MOVE & MAP PROGRAM
3401 015030                               ON.ERROR THEN $RETURN
3402                                       ;**NOTE** RECURSIVE CALL
3403 015034  004737  014674                CALL    PARBAW          ;CALL SELF
3404 015040  004737  042042                CALL    UNRELOCATE      ;UNMOVE & UNMAP PROGRAM
3405 015044  000207                       RETURN
```

```
3408 015046                          MTEST:  SUBTST  <<SUBR  SETUP MEMORY TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       SUBR    SETUP MEMORY TEST
                                     ;*****************************************************************************
                                     ;
3409 015046                                  SET     HEADER                  ;INITIALIZE HEADER MESSAGE TYPEOUT
3410 015054                                  SET     MUT                     ;INDICATE THERE IS A MEMORY UNDER TEST
3411 015062  005037  002256                  CLR     PASFLG
3412 015066  005737  002116                  TST     MKFLAG                  ,ECC?
3413 015072  001413                          BEQ     MT1                     ;NO - SKIP
3414 015074                                  BEGIN   HOLDLOOP
3415 015074                                    IF CONTFLAG IS TRUE THEN LEAVE HOLDLOOP
3416 015102                                    IF SKIPMK IS FALSE
3417 015110  004737  015142                      CALL        MKCONTROL
3418 015114                                    END; OF IF SKIPMK
3419 015114                                  END     HOLDLOOP
3420 015114  004737  016354                  CALL    MKTEST                  ;YES - DO ECC TESTS
3421 015120  000402                          BR      MT2
3422 015122  004737  016574          MT1:    CALL    MJTEST                  ;DO PARITY TESTS
3423 015126  005037  002106          MT2:    CLR     MUT                     ;NOW - NO MEMORY UNDER TEST
3424 015132                                  SET     HEADER                  ;ALLOW HEADERS NORMAL
3425 015140  000207                          RETURN
```

```
3428 015142                                  MKCONTROL:SUBTST        <<SUBR  TEST ECC CSR LOGIC DISPATCH>>
                                             ;*****************************************************************************
                                             ;*SUBTEST       SUBR    TEST ECC CSR LOGIC DISPATCH
                                             ;*****************************************************************************
3429                                                 ;THE NEXT TWO MODULES SOLVE THE PROBLEM OF
3430                                                 ;HOW TO RUN THE CSR TESTS ON EACH ECC MEMORY
3431                                                 ;
3432 015142                                          IF SELONLY IS TRUE THEN $RETURN
3433 015152                                          IF INHECC IS TRUE THEN $RETURN
3434 015162                                          PUSH    BANK,R0,R1,R2,R3
3435 015176  012737  060000  002224                 MOV     #FIRST,CSRFBANK         ;SET FIRST TEST ADDRESS TO FIRST ADDR.
3436 015204  012737  157776  002226                 MOV     #LAST,CSRLBANK
3437 015212  005037  002230                         CLR     CSRINT
3438 015216  005037  002232                         CLR     SPLTCSR
3439 015222  005037  002302                         CLR     CSRLOOP                 ; AND ZERO THE LOOP COUNTER
3440 015226  013700  002102                         MOV     BANKINDEX,R0            ;GET THE BANK INDEX
3441 015232  016001  002624                         MOV     CONFIG(R0),R1           ;GET CSR NUMBER
3442 015236  000301                                 SWAB    R1
3443 015240  042701  177760                         BIC     #^C17,R1
3444 015244  006301                                 ASL     R1
3445 015246  010137  002476                         MOV     R1,CSRHOLD              ;STORE IN THE LOW BYTE
3446 015252  005737  002134                         TST     INTFLAG                 ;IS THIS BANK INTERLEAVED?
3447 015256  001421                                 BEQ     1$                      ;BRANCH IF NOT INTERLEAVED
3448 015260  005237  002232                         INC     SPLTCSR
3449 015264  012737  120000  002226                 MOV     #120000,CSRLBANK
3450 015272  005237  002302                         INC     CSRLOOP                 ;WE MUST LOOP TWICE FOR AN INTERLEAVED BANK
3451 015276  005237  002230                         INC     CSRINT
3452 015302  016001  002624                         MOV     CONFIG(R0),R1           ;GET THE INTERLEAVE CSR NUMBER
3453 015306  072127  177775                         ASH     #-3,R1
3454 015312  042701  160777                         BIC     #^C17000,R1
3455 015316  050137  002476                         BIS     R1,CSRHOLD              ;STORE IT IN CSRHOLD'S UPPER BYTE
3456 015322  005003                        1$:      CLR     R3
3457 015324  116337  002476  002146      MKLOOP:  MOVB    CSRHOLD(R3),CSRNO
3458 015332  042737  177741  002146               BIC     #^C36,CSRNO             ;CLEAR ANY UNNECESSARY BITS
3459 015340                                          FOR R2 := #0 TO CSRINT
3460 015342                                            FOR CSRFIRST := CSRFBANK TO CSRLBANK BY #4000
3461 015350                                            MAP BANK                      ;MAP TEST SPACE TO BANK
3462 015364  104511                                    INVALIDATE                    ;INVALIDATE BACKROUND PATTERN
3463 015366                                              BEGIN CSRSTUFF
3464 015366  005037  002304                               CLR       SUCCESS
3465 015372                                                IF ACFLAG IS TRUE AND RRFLAG IS FALSE
3466 015406  013737  002220  002222                           MOV   CSRFIRST,CSRLAST
3467 015414  062737  004000  002222                           ADD   #4000,CSRLAST
3468 015422                                                    FOR TESTADD := CSRFIRST TO CSRLAST BY #4
3469 015430  013737  002362  002364                             MOV       TESTADD,TESTADD+2
3470 015436  005737  002232                                   TST       SPLTCSR
3471 015442  001404                                           BEQ       1$
3472 015444  062737  040000  002364                           ADD       #40000,TESTADD+2
3473 015452  000403                                           BR        2$
3474 015454  062737  000002  002364                     1$:   ADD       #2,TESTADD+2
3475 015462  004737  015750                             2$:   CALL      SBETEST
3476 015466                                                    ON.NOERROR
3477 015470  104424                                             CACHOFF                 ;TURN CACHE OFF
3478 015472  005037  002074                                     CLR     NOPAR           ;INDICATE PARITY ACTION
3479 015476                                                      FOR I := #0 TO #27
3480 015502                                                        SET   HEADER
3481 015510  005037  002256                                       CLR   PASFLG
```

```
3482 015514                                                   LET R0 := I
3483 015520                                                   PUSH  R3                ;SAVE LOOP COUNTER
3484 015522  010637  002142                                   MOV   SP,CTLKVEC        ;SAVE VECTOR IN CSR OF ^K
3485 015526  162737  000002  002142                           SUB   #2,CTLKVEC
3486 015534  004737  016254                                   CALL CSRCASE
3487 015540                                                   POP   R3                ;RESTORE LOOP COUNTER
3488 015542                                                 END ;OF FOR I
3489 015556  104423                                         CACHON                    ;TURN CACHE ON
3490 015560                                                 SET     SUCCESS
3491 015566                                                 LEAVE CSRSTUFF
3492 015570                                               END ;OF ON.NOERROR
3493 015570                                             END ;OF FOR TESTADD
3494 015606                                           END ;OF IF
3495 015606                                         END CSRSTUFF
3496 015606                                         IF SUCCESS IS FALSE
3497 015614                                           TYPE    MSGA34
3498 015620                                           TYPOCS  BANK,<TYPES BANK NUMBER>,3
3499 015630                                           TYPE    MSGB34
3500 015634                                         END ;OF IF SUCCESS
3501 015634                                       END; OF FOR CSRFIRST
3502 015652  005237  002232                       INC  SPLTCSR
3503 015656                                     END; OF FOR R2
3504 015666  062737  000002  002224           ADD     #2,CSRFBANK
3505 015674  012737  000001  002232           MOV     #1,SPLTCSR
3506 015702  005203                           INC     R3
3507 015704  020337  002302                   CMP     R3,CSRLOOP
3508 015710  003002                           BGT     1$
3509 015712  000137  015324                   JMP     MKLOOP
3510 015716  104472                   1$:     ECCINIT         ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
3511 015720                                   SET     CONTFLAG
3512 015726  005037  002232                   CLR     SPLTCSR
3513 015732                                   POP     R3,R2,R1,R0,BANK
3514 015746  000207                           RETURN
```

```
 3517 015750                    SBETEST:SUBTST  <<CHECK FOR SBE FREE LOCATIONS>>
                                ;*****************************************************************************
                                ;*SUBTEST       CHECK FOR SBE FREE LOCATIONS
                                ;*****************************************************************************
 3518                                   ;IN ORDER TO DETERMINE IF A LOCATION IS SBE FREE I DO THIS
 3519                                   ;
 3520                                   ;WRITE ZEROS WITH ECC DISABLE
 3521                                   ;READ ZEROS BACK
 3522                                   ;IF NOT ZEROS THEN RETURN ERROR
 3523                                   ;
 3524                                   ;WRITE ZEROS WITH ECC ENABLED BUT TRAPS DISABLED
 3525                                   ;READ ZEROS BACK
 3526                                   ;IF NOT ZEROS THEN RETURN ERROR
 3527                                   ;
 3528                                   ;TEST THE LOCATION FROM THE PAR'S (WITH NO PROGRAM FETCHES)
 3529                                   ;IF THERE WERE ANY SBE'S OR DBE'S THEN RETURN ERROR
 3530                                   ;
 3531                                   ;COMPLIMENT ZEROS TO ONES WITH ECC DISABLE
 3532                                   ;READ ONES BACK
 3533                                   ;IF NOT ONES THEN RETURN ERROR
 3534                                   ;
 3535                                   ;WRITE 100,,100000,00000 (CHECKBITS COMPLIMENT OF BEFORE)
 3536                                   ;  WITH ECC ENABLED BUT TRAPS DISABLED
 3537                                   ;TEST THE LOCATION FROM THE PAR'S (WITH NO PROGRAM FETCHES)
 3538                                   ;IF THERE WERE ANY SBE'S OR DBE'S THEN RETURN ERROR
 3539                                   ;
 3540                                   ;IF NONE OF THE ABOVE FORCES A RETURN ERROR THEN RETURN NO.ERROR
 3541                                   .ENABL  LSB
 3542 015750                    PUSH    R0,R1,R4                ;PUSH R0,R1,R4 ONTO STACK
 3543 015756  013701  002362    MOV     TESTADD,R1
 3544 015762  013704  002364    MOV     TESTADD+2,R4
 3545 015766                    TESTAREA                        ;ENTER TEST MODE
 3546 015774  104424            CACHOFF                         ;TURN CACHE OFF
 3547 015776  104471            ECC1DIS                         ;DISABLE ECC ON 1 SELECTED CSR
 3548 016000                    CLEAR   (R1),(R4)
 3549 016004  005711            TST     (R1)
 3550 016006  001107            BNE     SBENT
 3551 016010  005714            TST     (R4)
 3552 016012  001105            BNE     SBENT
 3553
 3554 016014  104503            CLR1CSR                         ;CLEAR 1 SELECTED CSR
 3555 016016                    CLEAR   (R1),(R4)
 3556 016022  005711            TST     (R1)
 3557 016024  001100            BNE     SBENT
 3558 016026  005714            TST     (R4)
 3559 016030  001076            BNE     SBENT
 3560
 3561 016032  104510            TSTREAD                 ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
 3562 016034                    IF #BIT15!BIT4 SET.IN CSR
 3563 016044                      SET SKPERR            ;DISABLE ERRGEN'S ERROR PRINTOUT
 3564 016052  104512              ERRGEN
 3565 016054  013700  002430      MOV   ERRADD,R0
 3566 016060  072027  177774      ASH   #-4,R0
 3567 016064  042700  177600      BIC   #^C177,R0
 3568 016070                      IF BANK EQ R0 THEN GOTO SBENT
 3569 016076                    END; OF IF #BIT15
 3570 016076  104471            ECC1DIS                         ;DISABLE ECC ON 1 SELECTED CSR
```

```
3571 016100  005111                      COM     (R1)
3572 016102  005114                      COM     (R4)
3573 016104  023711  002554              CMP     ONES,(R1)
3574 016110  001046                      BNE     SBENT
3575 016112  023714  002554              CMP     ONES,(R4)
3576 016116  001043                      BNE     SBENT
3577
3578 016120  104503                      CLR1CSR                         ;CLEAR 1 SELECTED CSR
3579 016122  005011                      CLR     (R1)
3580 016124  012714  100000              MOV     #BIT15,(R4)
3581 016130  005711                      TST     (R1)
3582 016132  001035                      BNE     SBENT
3583 016134  022714  100000              CMP     #BIT15,(R4)
3584 016140  001032                      BNE     SBENT
3585
3586 016142  104510                      TSTREAD                 ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
3587 016144                              IF #BIT15!BIT4 SET.IN CSR
3588 016154                                SET SKPERR                    ;DISABLE ERRGEN'S ERROR PRINTOUT
3589 016162  104512                       ERRGEN
3590 016164  013700  002430                MOV   ERRADD,R0
3591 016170  072027  177774                ASH   #-4,R0
3592 016174  042700  177600                BIC   #^C177,R0
3593 016200                                IF BANK EQ R0 THEN GOTO SBENT
3594 016206                              END; OF IF #BIT15
3595
3596 016206  104417                      KERNEL                          ;ENTER KERNEL MODE
3597 016210  104473                      ECC1INIT                        :INITIALIZE 1 SELECTED CSR
3598 016212  104423                      CACHON                          ;TURN CACHE ON
3599 016214                              POP     R4,R1,R0                ;POP R0,R1 & R4 FROM STACK
3600 016222                              $RETURN NOERROR
3601
3602 016226  104503              SBENT:  CLR1CSR                         ;CLEAR 1 SELECTED CSR
3603 016230                              CLEAR   (R1),(R4)
3604 016234  104417                      KERNEL                          ;ENTER KERNEL MODE
3605 016236  104473                      ECC1INIT                        ;INITIALIZE 1 SELECTED CSR
3606 016240  104423                      CACHON                          ;TURN CACHE ON
3607 016242                              POP     R4,R1,R0                ;POP R0,R1 & R4 FROM STACK
3608 016250                              $RETURN ERROR
3609                                     .DSABL  LSB
```

```
3612 016254                     CSRCASE:SUBTST  <<CSR PATTERN CASE STATEMENT>>
                                ;*****************************************************************************
                                ;*SUBTEST       CSR PATTERN CASE STATEMENT
                                ;*****************************************************************************
3613 016254                             CASE R0
3614                                      ;WARNING IF YOU CHANGE THIS TABLE ALSO
3615                                      ;CHANGE '$DDW0' - '$DDW5' (THE PATTERN BIT MAP)
3616
3617                                      ;PAT  TIME              DISCRIPTION
3618 016264  020014                MKCSRT:   MT0006          ;<1 SEC         INITIAL DATA TEST
3619 016266  020112                          MT0010          ;<1 SEC         BYTE ADDRESSING TEST
3620 016270  022402                          MT0025          ;<1 SEC         INTERRUPT ENABLE TEST
3621 016272  020146                          MT0011          ;<2 SEC         CREATE SINGLE BIT ERROR TEST
3622 016274  020214                          MT0012          ;<1 SEC         WRITE BYTE CLEARS SBE TEST
3623 016276  020310                          MT0013          ; 1 SEC         CREATE DOUBLE BIT ERROR TEST
3624 016300  020364                          MT0014          ; 1 SEC         WRITE INHIBIT DURING DATIP WITH DBE
3625 016302  020442                          MT0015          ; 1 SEC         WRITE INHIBIT OF BYTE WITH DBE
3626 016304  020510                          MT0016          ;<1 SEC         WRITE INHIBIT OF WORD WITH DBE
3627 016306  025074                          MT0999          ; 0 SEC         NULL TEST
3628 016310  025074                          MT0999          ; 0 SEC         NULL TEST
3629 016312  025074                          MT0999          ; 0 SEC         NULL TEST
3630 016314  025074                          MT0999          ; 0 SEC         NULL TEST
3631 016316  025074                          MT0999          ; 0 SEC         NULL TEST
3632 016320  025074                          MT0999          ; 0 SEC         NULL TEST
3633 016322  025074                          MT0999          ; 0 SEC         NULL TEST
3634 016324  025074                          MT0999          ; 0 SEC         NULL TEST
3635 016326  025074                          MT0999          ; 0 SEC         NULL TEST
3636 016330  025074                          MT0999          ; 0 SEC         NULL TEST
3637 016332  025074                          MT0999          ; 0 SEC         NULL TEST
3638 016334  025074                          MT0999          ; 0 SEC         NULL TEST
3639 016336  025074                          MT0999          ; 0 SEC         NULL TEST
3640 016340  025074                          MT0999          ; 0 SEC         NULL TEST
3641 016342  025074                          MT0999          ; 0 SEC         NULL TEST
3642 016344                     END ;OF CASE R0
3643 016352  000207              RETURN
```

```
                                        MKTEST: SUBTST  <<SUBR  ECC TEST DISPATCH>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       SUBR    ECC TEST DISPATCH
                                        ;*****************************************************************************
                                                IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE
                                                  ECCDIS                        ;DISABLE ERROR CORRECTION
                                                ELSE
                                                  CLRCSR                        ;CLEAR ALL CSR'S
                                                END ;OF IF
016400  012737  000002  002074                  MOV     #2,NOPAR              ;INDICATE PARITY ACTION
016406  012737  000002  002276                  MOV     #2,PCBUMP             ;TRAPS ADD 2 TO PC
016414  013700  002110                          MOV     PATTERN,R0            ;GET PATTERN NUMBER
016420  006300                                  ASL     R0                    ;MAKE IT A WORD ADDRESS
016422                                          IF MKPAT(R0) NE #MT0034 AND MKPAT(R0) NE #MT0999
016442  104511                                    INVALIDATE                  ;INVALIDATE BACKGROUND PATTERN ON 'BANK''
016444                                          END ;OF IF MKPAT(R0)
016444  010637  002142                          MOV     SP,CTLKVEC            ;SAVE VECTOR IN CASE OF ^K
016450  162737  000002  002142                  SUB     #2,CTLKVEC
016456  004770  016514                          CALL    @MKPAT(R0)            ;INDEX OFF TABLE
016462                                          IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE
016500  104506                                    ENASBE                      ;TRAP ON SINGLE BIT ERRORS
016502                                          ELSE
016504  104472                                    ECCINIT                     ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
016506                                          END ;OF IF #SW0
016506  005037  002074                          CLR     NOPAR                 ;INDICATE PARITY ACTION
016512  000207                                  RETURN

                                        ;WARNING IF YOU CHANGE THIS TABLE ALSO
                                        ;CHANGE '$DDW0' - '$DDW5' (THE PATTERN BIT MAP)
                                        ;PAT    TIME            DISCRIPTION
016514                                  MKPAT:  ;NOTE MT0034 MUST BE FIRST & LAST
016514  024616                                  MT0034  ;<1 SEC         ;SOFT ERROR - BACKGROUND PATTERN TEST
016516  020556                                  MT0017  ;<1 SEC         ;HOLDING 1'S & 0'S TEST
016520  020050                                  MT0007  ;<1 SEC         ;ADDRESS BIT TEST
016522  017036                                  MT0001  ;<1 SEC         ;ADDRESS TEST
016524  017154                                  MT0002  ;<1 SEC         ;COMPLEMENT ADDRESS TEST
016526  017542                                  MT0004  ; 1 SEC         ;ROTATING ZEROS TEST
016530  017662                                  MT0005  ; 1 SEC         ;ROTATING ONES TEST
016532  021630                                  MT0021  ; 1 SEC         ;MARCHING 0'S & 1'S TEST
016534  020600                                  MT0020  ;<1 SEC         ;MARCHING 1'S & 0'S IN CHECK BITS
016536  022052                                  MT0022  ;10 SEC         ;REFRESH & SHIFTING DIAGONAL  TEST
016540  022450                                  MT0026  ;<1 SEC         ;RANDOM DATA TEST
016542  022150                                  MT0024  ;20 SEC         ;FAST GALLOPING PATTERN TEST
016544  023712                                  MT0031  ; 3 SEC         ;SOB-A-LONG TEST
016546  024102                                  MT0032  ;<1 SEC         ;WRITE RECOVERY TEST
016550  024430                                  MT0033  ;35 SEC         ;BRANCH GOBBLE TEST
016552  024616                                  MT0034  ;<1 SEC         ;SOFT ERROR - BACKGROUND PATTERN TEST
                                                ;NOTE MT0034 MUST BE FIRST & LAST
016554  025074                                  MT0999  ; 0 SEC         ;NULL TEST
016556  025074                                  MT0999  ; 0 SEC         ;NULL TEST
016560  025074                                  MT0999  ; 0 SEC         ;NULL TEST
016562  025074                                  MT0999  ; 0 SEC         ;NULL TEST
016564  025074                                  MT0999  ; 0 SEC         ;NULL TEST
016566  025074                                  MT0999  ; 0 SEC         ;NULL TEST
016570  025074                                  MT0999  ; 0 SEC         ;NULL TEST
016572  025074                                  MT0999  ; 0 SEC         ;NULL TEST
```

```
3704 016574                          MJTEST: SUBTST  <<SUBR  PARITY TEST DISPATCH>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       SUBR    PARITY TEST DISPATCH
                                     ;*****************************************************************************
3708 016574  012737  000002  002074          MOV     #2,NOPAR                ;INDICATE PARITY ACTION
3709 016602  012737  000002  002276          MOV     #2,PCBUMP               ;TRAPS ADD 2 TO PC
3710 016610  012737  060000  002362          MOV     #FIRST,TESTADD
3711 016616  012737  060002  002364          MOV     #FIRST+2,TESTADD+2
3712 016624  013700  002110                  MOV     PATTERN,R0              ;GET PATTERN NUMBER
3713 016630  006300                          ASL     R0                      ;MAKE IT A WORD ADDRESS
3714 016632                                  IF MJPAT(R0) NE #MT0034 AND MJPAT(R0) NE #MT0999
3715 016652  104511                            INVALIDATE                    ;INVALIDATE BACKGROUND PATTERN ON 'BANK'
3716 016654                                  END ;OF IF MJPAT(R0)
3717 016654  010637  002142                  MOV     SP,CTLKVEC              ;SAVE VECTOR IN CASE OF ^K
3718 016660  162737  000002  002142          SUB     #2,CTLKVEC
3719 016666  004770  016700                  CALL    @MJPAT(R0)              ;INDEX OFF TABLE
3720 016672  005037  002074                  CLR     NOPAR                   ;IND'CATE PARITY ACTION
3721 016676  000207                          RETURN
3722
3723                                         ;WARNING IF YOU CHANGE THIS TABLE ALSO
3724                                         ;CHANGE '$DDW0' - '$DDW5' (THE PATTERN BIT MAP)
3725
3726                                         ;PAT    TIME            DISCRIPTION
3727 016700                          MJPAT:  ;NOTE MT0034 MUST BE FIRST & LAST
3728 016700  024616                          MT0034  ;<1 SEC         ;SOFT ERROR - BACKGROUND PATTERN TEST
3729 016702  020014                          MT0006  ;<1 SEC         ;INITIAL DATA TEST
3730 016704  020556                          MT0017  ;<1 SEC         ;HOLDING 1'S & 0'S TEST
3731 016706  020050                          MT0007  ;<1 SEC         ;ADDRESS BIT TEST
3732 016710  017036                          MT0001  ;<1 SEC         ;ADDRESS TEST
3733 016712  017154                          MT0002  ;<1 SEC         ;COMPLEMENT ADDRESS TEST
3734 016714  017312                          MT0003  ; 1 SEC         ;3 XOR 9 WORST CASE NOISE TEST
3735 016716  017542                          MT0004  ; 1 SEC         ;ROTATING ZEROS TEST
3736 016720  017662                          MT0005  ; 1 SEC         ;ROTATING ONES TEST
3737 016722  021630                          MT0021  ; 1 SEC         ;MARCHING 0'S & 1'S TEST
3738 016724  024762                          MT0035  ;<1 SEC         ;WORSE CASE NOISE PARITY TEST
3739 016726  022052                          MT0022  ;10 SEC         ;REFRESH TEST
3740 016730  022104                          MT0023  ;10 SEC         ;SHIFTING DIAGONAL  TEST
3741 016732  022450                          MT0026  ;<1 SEC         ;RANDOM DATA TEST
3742 016734  022150                          MT0024  ;20 SEC         ;FAST GALLOPING PATTERN TEST
3743 016736  023712                          MT0031  ; 3 SEC         ;SOB-A-LONG TEST
3744 016740  024102                          MT0032  ;<1 SEC         ;WRITE RECOVERY TEST
3745 016742  024430                          MT0033  ;35 SEC         ;BRANCH GOBBLE TEST
3746 016744  024616                          MT0034  ;<1 SEC         ;SOFT ERROR - BACKGROUND PATTERN TEST
3747                                         ;NOTE MT0034 MUST BE FIRST & LAST
3748 016746  025074                          MT0999  ; 0 SEC         ;NULL TEST
3749 016750  025074                          MT0999  ; 0 SEC         ;NULL TEST
3750 016752  025074                          MT0999  ; 0 SEC         ;NULL TEST
3751 016754  025074                          MT0999  ; 0 SEC         ;NULL TEST
3752 016756  025074                          MT0999  ; 0 SEC         ;NULL TEST
```

```
                                        .SBTTL  PATTERNS
                                        .SBTTL  MEMORY TEST SETUP ROUTINES
                                MT0000: SUBTST  <<MT0000        SETUP DATA PATTERN TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MT0000  SETUP DATA PATTERN TEST
                                ;*****************************************************************************
016760  005037  002260                  CLR     REALPAT         ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
016764  012700  060000                  MOV     #FIRST,R0
016770  012701  040000                  MOV     #SIZE,R1
016774  004737  035016                  CALL    REGCOPY
017000  005737  002424                  TST     NO22BIT         ;ARE WE ON AN 11/44?
017004  001406                          BEQ     1$              ;BRANCH IF YES
017006  012737  025514  002254          MOV     #MTP000,SUPDOADD ;ELSE DO PATTERN IN MAIN MEMORY
017014  004737  025322                  CALL    SUPD03
017020  000207                          RETURN
017022                          1$:     BMOV    MTP000
017030  004737  025144                  CALL    SUPD01                  ;DO IT IN SUPERVISOR MODE
017034  000207                          RETURN
017036                          MT0001: SUBTST  <<MT0001        SETUP ADDRESS TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MT0001  SETUP ADDRESS TEST
                                ;*****************************************************************************
017036  012737  000001  002260          MOV     #1,REALPAT              ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
017044  012700  060000                  MOV     #FIRST,R0
017050  012701  040000                  MOV     #SIZE,R1
017054  005737  002426                  TST     NOSUPER
017060  001005                          BNE     2$
017062  023737  172252  172254          CMP     SIPAR5,SIPAR6
017070  001007                          BNE     4$
017072  000404                          BR      3$
017074  023737  177652  177654  2$:     CMP     UIPAR5,UIPAR6
017102  001002                          BNE     4$
017104  012701  030000          3$:     MOV     #30000,R1
017110  005002                  4$:     CLR     R2
017112  004737  035016                  CALL    REGCOPY
017116  005737  002424                  TST     NO22BIT                 ;IS THIS AN 11/44?
017122  001406                          BEQ     1$                      ;BRANCH IF IT IS
017124  012737  025540  002254          MOV     #MTP001,SUPDOADD        ;SET UP CALLING ADDRESS
017132  004737  025322                  CALL    SUPD03
017136  000207                          RETURN
017140                          1$:     BMOV    MTP001
017146  004737  025144                  CALL    SUPD01                  ;DO IT IN SUPERVISOR MODE
017152  000207                          RETURN
017154                          MT0002: SUBTST  <<MT0002        SETUP COMPLEMENT ADDRESS TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MT0002  SETUP COMPLEMENT ADDRESS TEST
                                ;*****************************************************************************
017154  012737  000002  002260          MOV     #2,REALPAT              ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
017162  012700  160000                  MOV     #LAST+2,R0
017166  012701  040000                  MOV     #SIZE,R1
017172  012704  060000                  MOV     #FIRST,R4
017176  012705  100001                  MOV     #100001,R5
017202  005737  002426                  TST     NOSUPER
017206  001005                          BNE     2$
017210  023737  172252  172254          CMP     SIPAR5,SIPAR6
017216  001013                          BNE     4$
```

```
3802 017220  000404                          BR      3$
3803 017222  023737  177652  177654  2$:     CMP     UIPAR5,UIPAR6
3804 017230  001006                          BNE     4$
3805 017232  012701  030000          3$:     MOV     #30000,R1
3806 017236  012700  140000                  MOV     #140000,R0
3807 017242  012705  120001                  MOV     #120001,R5
3808 017246  012702  000001          4$:     MOV     #1,R2
3809 017252  010103                          MOV     R1,R3
3810 017254  005737  002424                  TST     N022BIT                 ;IS THIS AN 11/44?
3811 017260  001406                          BEQ     1$                      ;BRANCH IF TRUE
3812 017262  012737  025572  002254          MOV     #MTP002,SUPDOADD        ;SET UP CALLING ADDRESS
3813 017270  004737  025322                  CALL    SUPD03
3814 017274  000207                          RETURN
3815 017276                          1$:     BMOV    MTP002
3816 017304  004737  025144                  CALL    SUPD01
3817 017310  000207                          RETURN
```

```
3820 017312                                    MT0003: SUBTST  <<MT0003        SETUP 3 XOR 9 WORST CASE NOISE TEST>>
                                                ;*****************************************************************************
                                                ;*SUBTEST       MT0003  SETUP 3 XOR 9 WORST CASE NOISE TEST
                                                ;*****************************************************************************
3821 017312                                             IF EUFLAG IS TRUE THEN $RETURN
3822 017322  012737  000003  002260                     MOV     #3,REALPAT              ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3823 017330  005037  002276                     CLR     PCBUMP                  ;TRAPS DO NOT ADD TO PC
3824 017334  004737  035026             1$:     CALL    FLIPWARN                ;SETUP WARNING CONSTANTS & R2
3825 017340  012701  060000             2$:     MOV     #FIRST,R1               ;R1 <-- STARTING ADDRESS
3826 017344  012703  020000                     MOV     #20000,R3
3827 017350  072327  177770                     ASH     #-8.,R3                 ;R3 <-- R3 / 256.
3828 017354  012702  000004                     MOV     #4,R2                   ;SMALL LOOP SIZE
3829 017360  012705  000100                     MOV     #64.,R5                 ;MEDIUM LOOP SIZE
3830 017364  005737  002424                     TST     NO22BIT                 ;IS THIS AN 11/44?
3831 017370  001415                            BEQ     3$                      ;BRANCH IF IT IS
3832 017372  104415                            SAVREG
3833 017374  012737  025624  002254                     MOV     #MTPA03,SUPDOADD
3834 017402  004737  025322                     CALL    SUPD03                  ;DO IT IN MAIN MEMORY
3835 017406  104416                            RESREG
3836 017410  012737  025664  002254                     MOV     #MTPB03,SUPDOADD
3837 017416  004737  025336                     CALL    SUPD04
3838 017422  000442                            BR      4$
3839 017424                             3$:     BMOV    MTPA03
3840 017432  104415                            SAVREG
3841 017434  004737  025144                     CALL    SUPD01
3842 017440                                     BMOV    MTPB03
3843 017446                                     BMOV    MTPC03,KDPAR0,8.
3844 017460                                     BMOV    MTPD03,SDPAR0,8.
3845 017472  012737  172360  177642                     MOV     #KDPAR0,UIPAR1          ;SET UP PAR LINKS
3846 017500  012737  172260  172374                     MOV     #SDPAR0,KDPAR6
3847 017506  012737  177644  172276                     MOV     #UIPAR2,SDPAR7
3848 017514  012737  001032  172272                     MOV     #1032,SDPAR5            ;CHANGE INST TO BR .+66 (BR TO KDPAR1)
3849 017522  104416                            RESREG
3850 017524  004737  025160                     CALL    SUPD02
3851 017530  022737  000003  002556     4$:     CMP     #3,FLIPLOC              ;DONE WITH 4 PATTERNS
3852                                                                            ;[(0,177777);(177777,0);(401,177777);(177777,401)]?
3853 017536  001276                            BNE     1$                      ;NO - LOOP
3854 017540  000207                            RETURN
3855
3856 017542                                    MT0004: SUBTST  <<MT0004        SETUP ROTATING ZEROS TEST>>
                                                ;*****************************************************************************
                                                ;*SUBTEST       MT0004  SETUP ROTATING ZEROS TEST
                                                ;*****************************************************************************
3857 017542  012737  000004  002260                     MOV     #4,REALPAT              ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3858 017550  012737  000004  002276                     MOV     #4,PCBUMP               ;TRAPS ADD 4 TO PC
3859 017556  013702  002554                     MOV     ONES,R2
3860 017562  004737  035156                     CALL    BACKGND                 ;WRITE BACKGROUND OF ONES
3861 017566  012700  060000                     MOV     #FIRST,R0
3862 017572  012701  040000                     MOV     #SIZE,R1
3863 017576  005737  002424                     TST     NO22BIT                 ;IS THIS AN 11/44?
3864 017602  001406                            BEQ     1$                      ;BRANCH IF IT IS
3865 017604  012737  025762  002254                     MOV     #MTPA04,SUPDOADD        ;SET UP LINKS
3866 017612  004737  025336                     CALL    SUPD04
3867 017616  000207                            RETURN
3868 017620                             1$:     BMOV    MTPA04
3869 017626                                     BMOV    MTPB04,KDPAR0,8.
3870 017640  012737  172360  177652                     MOV     #KDPAR0,UIPAR5
```

```
 3871 017646  012737  177654  172376          MOV     #UIPAR6,KDPAR7
 3872 017654  004737  025160                  CALL    SUPD02
 3873 017660  000207                          RETURN
 3874 017662                          MT0005: SUBTST  <<MT0005        SETUP ROTATING ONES TEST>>
                                      ;*****************************************************************************
                                      ;*SUBTEST        MT0005  SETUP ROTATING ONES TEST
                                      ;*****************************************************************************
 3875 017662  012737  000005  002260          MOV     #5,REALPAT              ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
 3876 017670  012737  000004  002276          MOV     #4,PCBUMP               ;TRAPS ADD 4 TO PC
 3877 017676  005002                          CLR     R2
 3878 017700  004737  035156                  CALL    BACKGND                 ;WRITE BACKGROUND OF ZEROS
 3879 017704  012700  060000                  MOV     #FIRST,R0
 3880 017710  012701  040000                  MOV     #SIZE,R1
 3881 017714  005737  002424                  TST     NO22BIT                 ;IS THIS AN 11/44?
 3882 017720  001414                          BEQ     1$                      ;BRANCH IF IT IS
 3883 017722  012737  026036  002254          MOV     #MTP005,SUPD0ADD        ;SET UP LINKS
 3884 017730  012737  026052  026034          MOV     #MTP005+14,MTPB04+16
 3885 017736  004737  025336                  CALL    SUPD04
 3886 017742  012737  025776  026034          MOV     #MTPA04+14,MTPB04+16    ;RESET TEST'S ORIGINAL VALUE
 3887 017750  000207                          RETURN
 3888 017752                          1$:     BMOV    MTP005
 3889 017760                                  BMOV    MTPB04,KDPAR0,8.
 3890 017772  012737  172360  177652          MOV     #KDPAR0,UIPAR5
 3891 020000  012737  177654  172376          MOV     #UIPAR6,KDPAR7
 3892 020006  004737  025160                  CALL    SUPD02
 3893 020012  000207                          RETURN
```

```
3896 020014                          MT0006: SUBTST  <<MT0006        SETUP INITIAL DATA TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MT0006  SETUP INITIAL DATA TEST
                                     ;*****************************************************************************
3897 020014  012737  000006  002260          MOV     #6,REALPAT              ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3898 020022  012737  000004  002276          MOV     #4,PCBUMP               ;TRAPS ADD 4 TO PC
3899 020030  012701  002362                  MOV     #TESTADD,R1
3900 020034  012737  026072  002254          MOV     #MTP006,SUPDOADD
3901 020042  004737  025322                  CALL    SUPD03                  ;DO IT IN SUPERVISOR MODE
3902 020046  000207                          RETURN
3903 020050                          MT0007: SUBTST  <<MT0007        SETUP ADDRESS BIT TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MT0007  SETUP ADDRESS BIT TEST
                                     ;*****************************************************************************
3904 020050  012737  000007  002260          MOV     #7,REALPAT              ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3905 020056  005002                          CLR     R2
3906 020060  004737  035156                  CALL    BACKGND                 ;OF ZEROS
3907 020064  012701  060000                  MOV     #FIRST,R1
3908 020070  012702  000001                  MOV     #1,R2
3909 020074  050201                          BIS     R2,R1
3910 020076  012737  026272  002254          MOV     #MTP007,SUPDOADD
3911 020104  004737  025322                  CALL    SUPD03                  ;DO IT IN SUPERVISOR MODE
3912 020110  000207                          RETURN
3913 020112                          MT0010: SUBTST  <<MT0010        SETUP BYTE ADDRESSING TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MT0010  SETUP BYTE ADDRESSING TEST
                                     ;*****************************************************************************
3914 020112  012737  000010  002260          MOV     #10,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3915 020120  012737  000004  002276          MOV     #4,PCBUMP               ;TRAPS ADD 4 TO PC
3916 020126  013704  002362                  MOV     TESTADD,R4
3917 020132  012737  026372  002254          MOV     #MTP010,SUPDOADD
3918 020140  004737  025322                  CALL    SUPD03                  ;DO IT IN SUPERVISOR MODE
3919 020144  000207                          RETURN
```

```
3922 020146                          MT0011: SUBTST  <<MT0011        SETUP CREATE SINGLE BIT ERROR TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MT0011  SETUP CREATE SINGLE BIT ERROR TEST
                                     ;*****************************************************************************
3923 020146                                  IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
3924 020162                                    IF $PASS NE #0 THEN $RETURN
3925 020172                                  END; OF IF ACTFLAG
3926 020172  012737  000011  002260          MOV     #11,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3927 020200  012737  026500  002254          MOV     #MTP011,SUPDOADD
3928 020206  004737  025322                  CALL    SUPDO3                  ;DO IT IN SUPERVISOR MODE
3929 020212  000207                          RETURN
3930 020214                          MT0012: SUBTST  <<MT0012        SETUP WRITE BYTE CLEARS SBE TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MT0012  SETUP WRITE BYTE CLEARS SBE TEST
                                     ;*****************************************************************************
3931 020214                                  IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
3932 020230                                    IF $PASS NE #0 THEN $RETURN
3933 020240                                  END; OF IF ACTFLAG
3934 020240  012737  000012  002260          MOV     #12,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3935 020246  013700  002102                  MOV     BANKINDEX,R0
3936 020252                                  IF #BIT12 SET.IN CONFIG+2(R0)
3937 020262  012705  040000                    MOV   #40000,R5
3938 020266                                  ELSE
3939 020270  012705  000002                    MOV   #2,R5
3940 020274                                  END; OF IF #BIT12
3941 020274  012737  027244  002254          MOV     #MTP012,SUPDOADD
3942 020302  004737  025322                  CALL    SUPDO3                  ;DO IT IN SUPERVISOR MODE
3943 020306  000207                          RETURN
3944 020310                          MT0013: SUBTST  <<MT0013        SETUP CREATE DOUBLE BIT ERROR TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MT0013  SETUP CREATE DOUBLE BIT ERROR TEST
                                     ;*****************************************************************************
3945 020310                                  IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
3946 020324                                    IF $PASS NE #0 THEN $RETURN
3947 020334                                  END; OF IF ACTFLAG
3948 020334  012737  000013  002260          MOV     #13,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3949 020342  012737  027632  002254          MOV     #MTP013,SUPDOADD
3950 020350  012737  000003  002074          MOV     #3,NOPAR                ;INDICATE PARITY ACRION
3951 020356  004737  025322                  CALL    SUPDO3                  ;DO IT IN SUPERVISOR MODE
3952 020362  000207                          RETURN
3953 020364                          MT0014: SUBTST  <<MT0014        SETUP WRITE INHIBIT DURING DATIP WITH DBE>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MT0014  SETUP WRITE INHIBIT DURING DATIP WITH DBE
                                     ;*****************************************************************************
3954 020364                                  IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
3955 020400                                    IF $PASS NE #0 THEN $RETURN
3956 020410                                  END; OF IF ACTFLAG
3957 020410                                  IF KFLAG IS FALSE THEN $RETURN
3958 020420  012737  000014  002260          MOV     #14,REALPAT             ;SFTUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3959 020426  012737  030346  002254          MOV     #MTP014,SUPDOADD
3960 020434  004737  025322                  CALL    SUPDO3                  ;DO IT IN SUPERVISOR MODE
3961 020440  000207                          RETURN
```

```
3964 020442                                MT0015: SUBTST  <<MT0015        SETUP WRITE INHIBIT OF BYTE WITH DBE>>
                                           ;*********************************************************************************
                                           ;*SUBTEST       MT0015  SETUP WRITE INHIBIT OF BYTE WITH DBE
                                           ;*********************************************************************************
3965 020442                                        IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
3966 020456                                          IF $PASS NE #0 THEN $RETURN
3967 020466                                        END ;OF IF ACTFLAG
3968 020466  012737  000015  002260                MOV     #15,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3969 020474  012737  031130  002254                MOV     #MTP015,SUPDOADD
3970 020502  004737  025322                        CALL    SUPD03                  ;DO IT IN SUPERVISOR MODE
3971 020506  000207                                RETURN
3972 020510                                MT0016: SUBTST  <<MT0016        SETUP WRITE INHIBIT OF WORD WITH DBE>>
                                           ;*********************************************************************************
                                           ;*SUBTEST       MT0016  SETUP WRITE INHIBIT OF WORD WITH DBE
                                           ;*********************************************************************************
3973 020510                                        IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
3974 020524                                          IF $PASS NE #0 THEN $RETURN
3975 020534                                        END ;OF IF ACTFLAG
3976 020534  012737  000016  002260                MOV     #16,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3977 020542  012737  031674  002254                MOV     #MTP016,SUPDOADD
3978 020550  004737  025322                        CALL    SUPD03                  ;DO IT IN SUPERVISOR MODE
3979 020554  000207                                RETURN
3980 020556                                MT0017: SUBTST  <<MT0017        SETUP HOLDING 1'S & 0'S>>
                                           ;*********************************************************************************
                                           ;*SUBTEST       MT0017  SETUP HOLDING 1'S & 0'S
                                           ;*********************************************************************************
3981 020556  012737  000017  002260                MOV     #17,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3982 020564  012737  032456  002254                MOV     #MTP017,SUPDOADD
3983 020572  004737  025322                        CALL    SUPD03                  ;DO IT IN SUPERVISOR MODE
3984 020576  000207                                RETURN
```

```
 3987 020600                                MT0020: SUBTST  <<MT0020        SETUP MARCHING 0'S & 1'S IN CHECKBITS TEST>>
                                            ;*****************************************************************************
                                            ;*SUBTEST       MT0020  SETUP MARCHING 0'S & 1'S IN CHECKBITS TEST
                                            ;*****************************************************************************
 3988 020600                                        IF ACTFLAG IS TRUE OR ACTFLAG IS TRUE
 3989 020614                                          IF $PASS NE #0 THEN $RETURN
 3990 020624                                        END ;OF IF ACTFLAG
 3991 020624  012737  000020  002260                MOV     #20,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
 3992 020632  012737  000003  002074                MOV     #3,NOPAR                ;INDICATE PARITY ACTION
 3993 020640  005001                                CLR     R1                      ;CLEAR LOOP COUNTER
 3994 020642  005004                                CLR     R4                      ;CLEAR INTERLEAVE ODD/EVEN FLAG
 3995 020644  012700  060000                        MOV     #FIRST,R0
 3996 020650  013702  002102                        MOV     BANKINDEX,R2            ;SET BANK INDEX
 3997 020654                                MTL020: IF INTFLAG IS FALSE
 3998 020662                                          BEGIN MTB020
 3999 020662                                          IF NO22BIT IS TRUE
 4000 020670                                            IF LASTBANK EQ BANK
 4001 020700  012737  140000  002362                      MOV       #140000,TESTADD ;SET UP 12K NON-INTERLEAVED VIRT ADDR
 4002 020706  012705  140002                            MOV       #140002,R5
 4003 020712                                            LEAVE     MTB020
 4004 020714                                          END; OF IF LASTBANK
 4005 020714                                        END; OF IF NO22BIT
 4006 020714  012737  160000  002362                MOV   #LAST+2,TESTADD
 4007 020722  012705  160002                        MOV   #LAST+4,R5
 4008 020726                                        END   MTB020
 4009 020726  010537  002364                        MOV   R5,TESTADD+2              ;SET UP NON-INTERLEAVED VIRT. ADDR.
 4010 020732                                      ELSE
 4011 020734  005737  002312                        TST   SKIPMK                    ;IS THIS BANK IN SKIP RANGE?
 4012 020740  001401                                BEQ   1$                        ;BANK IS OUT OF RANGE - DO TEST
 4013 020742  000207                                RETURN                          ;LEAVE TEST-BANK'S ALREADY TESTED
 4014 020744  012737  120000  002362  1$:           MOV   #120000,TESTADD           ;SET UP 1ST INTERLEAVED VIRT. ADDR.
 4015 020752  012705  160000                        MOV   #LAST+2,R5                ;SET UP END OF BANK FLAG
 4016 020756  010537  002364                        MOV   R5,TESTADD+2              ;SET UP 2ND INT'L. VIRT. ADDR.
 4017 020762  005237  002232                        INC   SPLTCSR                   ;FLAG THE MAPPING ROUTINE FOR INTERLEAVING
 4018 020766  005201                                INC   R1                        ;SET LOOP COUNTER FOR INTERLEAVING
 4019 020770  005204                                INC   R4                        ;SET ODD/EVEN FLAG
 4020 020772                                      END; OF IF INTFLAG
 4021 020772  016203  002624                      MOV     CONFIG(R2),R3           ;SET UP CSR NUMBER
 4022 020776                                      IF R4 EQ #2                     ;IF THE SECOND TIME AROUND
 4023 021004  060437  002362                        ADD   R4,TESTADD
 4024 021010  060437  002364                        ADD   R4,TESTADD+2            ;TEST THE A1 ASSERTED ADDRESSES
 4025 021014  060400                                ADD   R4,R0
 4026 021016  060405                                ADD   R4,R5
 4027 021020  072327  177775                        ASH   #-3,R3                  ;MOVE INTERLEAVED CSR NUMBER
 4028 021024                                      ELSE
 4029 021026  006303                                ASL   R3                      ;MOVE CSR NUMBER
 4030 021030                                      END; IF R4
 4031 021030  000303                              SWAB    R3
 4032 021032  042703  177741                      BIC     #^C36,R3
 4033 021036  010337  002146                      MOV     R3,CSRNO              ;MOVE R3 INTO CSR NUMBER
 4034 021042                                      IF #SW0 SET.IN @SWR
 4035 021052  104506                                ENASBE                      ;TRAP ON SINGLE BIT ERRORS
 4036 021054                                      ELSE
 4037 021056  104472                                ECCINIT                     ;TRAP ON UNCORRECTABLE ERRORS
 4038 021060                                      END; OF IF #SW0
 4039 021060                                      PUSH    R2,R4
 4040 021064                                      FOR MTV020 := #0 TO R1
```

```
 4041 021070                                        PUSH  R1
 4042 021072  005002                                CLR   R2                        ;PATTERN TO WRITE INTO BANK
 4043 021074  004737  035156                        CALL  BACKGND                   ;SET UP ZEROS IN BANK
 4044 021100                                        IF NO22BIT IS TRUE AND MTV020 EQ #1 AND BANK EQ #3
 4045 021126  162737  020000  002362                  SUB #20000,TESTADD            ;SET UP 12K INTERLEAVED BANK
 4046 021134  162705  020000                        SUB #20000,R5
 4047 021140  010537  002364                        MOV R5,TESTADD+2
 4048 021144                                        END; OF IF NO22BIT
 4049 021144  004737  021220                        CALL  MT020Z                    ;START TEST
 4050 021150  005237  002232                        INC   SPLTCSR                   ;UPDATE INTERLEAVED MAPPING FLAG
 4051 021154                                        POP   R1
 4052 021156                                      END; OF FOR MTV020
 4053 021170                                      POP     R4,R2
 4054 021174  005001                              CLR     R1                        ;RESET LOOP FLAG
 4055 021176  005037  002232                      CLR     SPLTCSR                   ;RESET INTERLEAVED MAP FLAG
 4056 021202  022704  000001                      CMP     #1,R4                     ;ODD/EVEN FLAG SET?
 4057 021206  001622                              BEQ     MTL020                    ;BRANCH IF TRUE
 4058 021210  005037  002074                      CLR     NOPAR                     ;INDICATE PARITY ACTION
 4059 021214  000207                              RETURN
 4060 021216  000000                   MTV020:    0                                 ;VARIABLE FOR PAT 20
```

```
 4062 021220  012702  000004          MT020Z: MOV     #4,R2                   ;SET UP WORD INCR/DECR AMOUNT
 4063 021224  012700  060000                  MOV     #FIRST,R0
 4064 021230  013701  002362                  MOV     TESTADD,R1
 4065 021234  013704  002364                  MOV     TESTADD+2,R4
 4066 021240  012703  100000                  MOV     #BIT15,R3
 4067 021244                                  IF #SW11 SET.IN @SWR OR QVFLAG IS TRUE
 4068 021262                                    GOTO  MT020Y
 4069 021264                                  END ;OF IF #SW11
 4070 021264  005737  002424                  TST     NO22BIT                 ;IS THIS AN 11/44?
 4071 021270  001411                          BEQ     1$                      ;BRANCH IF IT IS
 4072 021272  012737  032534  002254          MOV     #MTPA20,SUPDOADD
 4073 021300  012737  032550  002264          MOV     #MTPA20+14,PARTHERE     ;VECTOR FOR TRAPS
 4074 021306  004737  025322                  CALL    SUPD03
 4075 021312  000410                          BR      2$
 4076 021314                          1$:     BMOV    MTPA20
 4077 021322  012737  177654  002264          MOV     #UIPAR6,PARTHERE        ;VECTOR FOR TRAPS
 4078 021330  004737  025144                  CALL    SUPD01
 4079 021334  005737  002424          2$:     TST     NO22BIT                 ;IS THIS AN 11/44?
 4080 021340  001411                          BEQ     4$                      ;BRANCH IF IT IS
 4081 021342  012737  032564  002254          MOV     #MTPB20,SUPDOADD
 4082 021350  012737  032574  002264          MOV     #MTPB20+10,PARTHERE     ;VECTOR FOR TRAPS
 4083 021356  004737  025336                  CALL    SUPD04
 4084 021362  000410                          BR      MT020Y
 4085 021364                          4$:     BMOV    MTPB20
 4086 021372  012737  177650  002264          MOV     #UIPAR4,PARTHERE        ;VECTOR FOR TRAPS
 4087 021400  004737  025160                  CALL    SUPD02
 4088 021404  005737  002134          MT020Y: TST     INTFLAG                 ;ARE WE INTERLEAVED?
 4089 021410  001405                          BEQ     7$                      ;BRANCH IF NOT INTERLEAVED
 4090 021412  162701  040000                  SUB     #40000,R1               ;RESET FIRST WORD TO BEGINNING OF BANK
 4091 021416  162704  040000                  SUB     #40000,R4               ;RESET SECOND WORD TO BFGINNING OF BANK
 4092 021422  000404                          BR      8$
 4093 021424  012701  060000          7$:     MOV     #FIRST,R1               ;RESET FIRST WORD TO BEGINNING OF BANK
 4094 021430  012704  060002                  MOV     #FIRST+2,R4             ;RESET SECOND WORD TO BEGINNING OF BANK
 4095 021434  005737  002424          8$:     TST     NO22BIT                 ;IS THIS AN 11/44?
 4096 021440  001411                          BEQ     1$                      ;BRANCH IF IT IS
 4097 021442  012737  032614  002254          MOV     #MTPC20,SUPDOADD
 4098 021450  012737  032624  002264          MOV     #MTPC20+10,PARTHERE     ;VECTOR FOR TRAPS
 4099 021456  004737  025322                  CALL    SUPD03
 4100 021462  000410                          BR      2$
 41?' 021464                          1$:     BMOV    MTPC20
 4102 021472  012737  177650  002264          MOV     #UIPAR4,PARTHERE        ;VECTOR FOR TRAPS
 4103 021500  004737  025144                  CALL    SUPD01
 4104 021504  005737  002424          2$:     TST     NO22BIT                 ;IS THIS AN 11/44?
 4105 021510  001411                          BEQ     3$                      ;BRANCH IF IT IS
 4106 021512  012737  032644  002254          MOV     #MTPD20,SUPDOADD
 4107 021520  012737  032660  002264          MOV     #MTPD20+14,PARTHERE     ;VECTOR FOR TRAPS
 4108 021526  004737  025336                  CALL    SUPD04
 4109 021532  000410                          BR      4$
 4110 021534                          3$:     BMOV    MTPD20
 4111 021542  012737  177654  002264          MOV     #UIPAR6,PARTHERE        ;VECTOR FOR TRAPS
 4112 021550  004737  025160                  CALL    SUPD02
 4113 021554  005737  002424          4$:     TST     NO22BIT                 ;IS THIS AN 11/44?
 4114 021560  001411                          BEQ     5$                      ;BRANCH IF IT IS
 4115 021562  012737  032674  002254          MOV     #MTPE20,SUPDOADD
 4116 021570  012737  032704  002264          MOV     #MTPE20  0,PARTHERE     ;VECTOR FOR TRAPS
 4117 021576  004737  025336                  CALL    SUPD04
 4118 021602  000410                          BR      6$
```

```
 4119 021604                                5$:     BMOV    MTPE20
 4120 021612  012737  177650  002264               MOV     #UIPAR4,PARTHERF        ;VECTOR FOR TRAPS
 4121 021620  004737  025160                       CALL    SUPD02
 4122 021624  104503                        6$:     CLR1CSR                         ;CLEAR 1 SELECTED CSR
 4123 021626  000207                                RETURN
```

```
4126 021630                                 MT0021: SUBTST  <<MT0021        SETUP MARCHING 0'S & 1'S TEST>>
                                            ;*****************************************************************************
                                            ;*SUBTEST       MT0021  SETUP MARCHING 0'S & 1'S TEST
                                            ;*****************************************************************************
4127 021630                                         SET NOSCOPE
4128 021636  012737  000021  002260                 MOV     #21,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4129 021644  013702  002572                         MOV     BAKPAT,R2
4130 021650  004737  035156                         CALL    BACKGND
4131 021654  010203                                 MOV     R2,R3
4132 021656  000303                                 SWAB    R3
4133 021660  012701  160000                         MOV     #LAST+2,R1
4134 021664  010105                                 MOV     R1,R5
4135 021666  012704  060000                         MOV     #FIRST,R4
4136 021672  005737  002424                         TST     NO22BIT         ;IS THIS AN 11/44?
4137 021676  001435                                 BEQ     1$              ;BRANCH IF IT IS
4138 021700  022737  000007  002100                 CMP     #7,BANK
4139 021706  001003                                 BNE     3$
4140 021710  012701  140000                         MOV     #140000,R1
4141 021714  010105                                 MOV     R1,R5
4142 021716  012737  032720  002254  3$:           MOV     #MTPA21,SUPDOADD
4143 021724  004737  025322                         CALL    SUPD03
4144 021730  012737  032750  002254                 MOV     #MTPB21,SUPDOADD
4145 021736  004737  025336                         CALL    SUPD04
4146 021742  010401                                 MOV     R4,R1
4147 021744  012737  033004  002254                 MOV     #MTPC21,SUPDOADD
4148 021752  004737  025336                         CALL    SUPD04
4149 021756  012737  033040  002254                 MOV     #MTPD21,SUPDOADD
4150 021764  004737  025336                         CALL    SUPD04
4151 021770  000425                                 BR      2$
4152 021772                                 1$:     BMOV    MTPA21
4153 022000  004737  025144                         CALL    SUPD01
4154
4155 022004                                         BMOV    MTPB21
4156 022012  004737  025160                         CALL    SUPD02
4157
4158 022016  010401                                 MOV     R4,R1
4159 022020                                         BMOV    MTPC21
4160 022026  004737  025160                         CALL    SUPD02
4161
4162 022032                                         BMOV    MTPD21
4163 022040  004737  025160                         CALL    SUPD02
4164 022044  005037  002410             2$:         CLR     NOSCOPE
4165 022050  000207                                 RETURN
```

```
4167 022052                                MT0022: SUBTST  <<MT0022        SETUP REFRESH & SHIFTING DIAGONAL TEST>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       MT0022  SETUP REFRESH & SHIFTING DIAGONAL TEST
                                           ;*****************************************************************************
                                           ;
4168 022052  004737  025110                        CALL    KAMITEST                ;CHECK FOR KAMIKAZE MODE
4169 022056                                        ON.ERROR THEN $RETURN           ;IF NOT IN KAMIKAZE MODE RETURN
4170 022062  012737  000022  002260                MOV     #22,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4171 022070  012737  033070  002254                MOV     #MTP022,SUPDOADD
4172 022076  004737  025322                        CALL    SUPDO3                  ;DO IT IN SUPERVISOR MODE
4173 022102  000207                               RETURN
4174
4175 022104                                MT0023: SUBTST  <<MT0023        SHIFTING DIAGONAL TEST>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       MT0023  SHIFTING DIAGONAL TEST
                                           ;*****************************************************************************
                                           ;
4176 022104  004737  025110                        CALL    KAMITEST                ;CHECK FOR KAMIKAZE MODE
4177 022110                                        ON.ERROR THEN $RETURN           ;IF NOT IN KAMIKAZE MODE RETURN
4178 022114  012737  000023  002260                MOV     #23,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4179 022122  012737  033070  002254                MOV     #MTP022,SUPDOADD
4180 022130                                        SET     DIAGFLAG                ;IDENTIFY DIAGONAL TEST TO MTP022
4181 022136  004737  025322                        CALL    SUPDO3                  ;DO IT IN SUPERVISOR MODE
4182 022142  005037  002002                        CLR     DIAGFLAG
4183 022146  000207                               RETURN
```

```
4185 022150                                MT0024: SUBTST  <<MT0024        SETUP FAST GALLOPING PATTERN TEST>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       MT0024  SETUP FAST GALLOPING PATTERN TEST
                                           ;*****************************************************************************
                                           ;
4186 022150  004737  025110                        CALL    KAMITEST                ;CHECK FOR KAMIKAZE MODE
4187 022154                                        ON.ERROR THEN $RETURN           ;IF NOT IN KAMIKAZE MODE RETURN
4188 022160                                        SET     NOSCOPE
4189 022166  012737  000024  002260                MOV     #24,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4190 022174  013702  002572                        MOV     BAKPAT,R2
4191 022200  004737  035156                        CALL    BACKGND
4192 022204  010203                                MOV     R2,R3
4193 022206  010304                                MOV     R3,R4
4194 022210  000304                                SWAB    R4
4195 022212  012701  060000                        MOV     #FIRST,R1
4196 022216  012705  15 '76                        MOV     #LAST,R5
4197 022222  005737  002424                        TST     NO22BIT
4198 022226  001413                                BEQ     1$
4199 022230  022737  000007  002100                CMP     #7,BANK
4200 022236  001002                                BNE     3$
4201 022240  012705  137776                        MOV     #137776,R5
4202 022244  104415                        3$:     SAVREG
4203 022246  012737  033604  002254                MOV     #MTPB24,SUPDOADD
4204 022254  000440                                BR      2$
4205 022256  022737  000177  002526        1$:     CMP     #177,LASTBANK
4206 022264  001002                                BNE     4$
4207 022266  012705  137776                        MOV     #137776,R5
4208 022272  104415                        4$:     SAVREG
4209 022274                                        BMOV    MTPA24
4210 022302                                        BMOV    MTPB24,SDPAR0,8.
4211 022314                                        BMOV    MTPC24,KDPAR0,8.
4212 022326  012737  172260  002254                MOV     #SDPAR0,SUPDOADD
4213 022334  012737  172260  177676                MOV     #SDPAR0,UDPAR7          ;SET UP PAR LINKS
4214 022342  012737  172360  172272                MOV     #KDPAR0,SDPAR5
4215 022350  012737  177660  172374                MOV     #UDPAR0,KDPAR6
4216 022356  004737  025336                2$:     CALL    SUPD04
4217
4218                                               ;DO IT AGAIN FOR COMPLEMENT DATA
4219 022362  104416                                RESREG
4220 022364  000302                                SWAB    R2
4221 022366  000303                                SWAB    R3
4222 022370  004737  025336                        CALL    SUPD04
4223 022374  005037  002410                        CLR     NOSCOPE
4224 022400  000207                                RETURN
4225 022402                                MT0025: SUBTST  <<MT0025        SETUP INTERRUPT ENABLE TEST>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       MT0025  SETUP INTERRUPT ENABLE TEST
                                           ;*****************************************************************************
4226 022402                                        IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
4227 022416                                          IF $PASS NE #0 THEN $RETURN
4228 022426                                        END ;OF IF ACTFLAG
4229 022426  012737  000025  002260                MOV     #25,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4230 022434  012737  033636  002254                MOV     #MTP025,SUPDOADD
4231 022442  004737  025322                        CALL    SUPD03                  ;DO IT IN SUPERVISOR MODE
4232 022446  000207                                RETURN
```

```
4235 022450                                  MT0026: SUBTST  <<MT0026        SETUP RANDOM DATA TEST>>
                                             ;*****************************************************************************
                                             ;*SUBTEST       MT0026  SETUP RANDOM DATA TEST
                                             ;*****************************************************************************
4236 022450  012737  000026  002260                 MOV     #26,REALPAT
4237 022456  005037  002276                         CLR     PCBUMP                  ;TRAPS DO NOT ADD TO THE PC
4238 022462  013703  002544                         MOV     SEEDLO,R3               ;INITIALIZE RANDOM NUMBERS
4239 022466  013702  002542                         MOV     SEEDHI,R2
4240 022472  010305                                 MOV     R3,R5
4241 022474  010204                                 MOV     R2,R4
4242 022476  012701  060000                         MOV     #FIRST,R1
4243 022502  012700  020000                         MOV     #SIZE/2,R0
4244 022506  005737  002424                         TST     NO22BIT                 ;DO WE HAVE AN 11/44?
4245 022512  001433                                 BEQ     1$                      ;BRANCH IF WE DO
4246 022514  022737  000007  002100                 CMP     #7,BANK
4247 022522  001002                                 BNE     3$
4248 022524  012700  014000                         MOV     #14000,R0
4249 022530  104415                        3$:      SAVREG
4250 022532  012737  034310  034410                 MOV     #MTPA26+4,MTPD26+14
4251 022540  012737  034304  002254                 MOV     #MTPA26,SUPDOADD
4252 022546  004737  025322                         CALL    SUPDO3
4253 022552  005037  034334                         CLR     RANODD                  ;FOR ERROR REPORTING
4254 022556  012737  034324  034410                 MOV     #MTPB26+4,MTPD26+14     ;SET UP NEXT LINK
4255 022564  012737  034320  002254                 MOV     #MTPB26,SUPDOADD
4256 022572  104416                                 RESREG
4257 022574  004737  025322                         CALL    SUPDO3
4258 022600  000452                                 BR      2$
4259 022602  022737  000177  002100  1$:      CMP     #177,BANK
4260 022610  001002                                 BNE     4$
4261 022612  012700  014000                         MOV     #14000,R0
4262 022616  104415                        4$:      SAVREG
4263 022620                                 BMOV    MTPA26                  ;WRITE ROUTINE TO FAST MEMORY
4264 022626                                 BMOV    MTPC26,KDPAR0,8.        ;RANDOM SUBPROGRAM TO FAST MEMORY
4265 022640  012737  000730  172376         MOV     #730,KDPAR7             ;WRITES 'BR .-116'' IN (BR SDPAR0)
4266 022646                                 BMOV    MTPD26,SDPAR0,8.        ;RANDOM SUBSUBPROGRAM TO FAST MEMORY
4267 022660  012737  172360  177642         MOV     #KDPAR0,UIPAR1
4268 022666  012737  177644  172274         MOV     #UIPAR2,SDPAR6
4269 022674  004737  025144                 CALL    SUPDC1                  ;WRITE RANDOM DATA
4270 022700  005037  034334                 CLR     RANODD                  ;FOR ERROR REPORTING
4271 022704                                 BMOV    MTPB26                  ;READ ROUTINE TO FAST MEMORY
4272 022712  012737  172360  177642         MOV     #KDPAR0,UIPAR1          ;SET UP PAR LINK
4273 022720  104416                         RESREG
4274 022722  004737  025144                 CALL    SUPDO1                  ;READ RANDOM DATA
4275 022726  010337  002544          2$:    MOV     R3,SEEDLO               ;UPDATE FOR NEW RANDOM NUMBERS
4276 022732  010237  002542                 MOV     R2,SEEDHI
4277 022736  000207                         RETURN
```

```
4280 022740                          MT0027: SUBTST  <<MT0027        UNIQUE BANK TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MT0027  UNIQUE BANK TEST
                                     ;*****************************************************************************
4281                                         ;MAKE SURE THAT EACH BANK CAN HAVE UNIQUE DATA
4282                                         ;WRITE AND READ THE BANK NUMBER IN EACH BANK (EXCEPT WHERE THE PROGRAM IS)
4283 022740  012737  000027  002260          MOV     #27,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4284 022746  104502                          CLRCSR                  ;CLEAR CSRS
4285 022750  005737  002424                  TST     NO22BIT                 ;IS THIS AN 11/44?
4286 022754  001404                          BEQ     1$                      ;BRANCH IF TRUE
4287 022756  012737  025322  002472          MOV     #SUPD03,LINK1           ;SET UP LINK
4288 022764  000414                          BR      STAR27                  ;BRANCH TO RUN
4289 022766                          1$:     BMOV    MTP034
4290 022774  012737  177646  002254  WARN7:  MOV     #UIPAR3,SUPDOADD
4291 023002  012737  025144  002472          MOV     #SUPD01,LINK1           ;SET UP LINK
4292 023010                                  SET NOFSMODE
4293 023016                          STAR27: FOR I := #1 TO #2
4294 023024                                    FOR BANK := #0 TO LASTBANK
4295 023030  004737  042700                     CALL EXBANK
4296 023034                                     IF ACFLAG IS TRUE AND RRFLAG IS FALSE
4297 023050  104511                               INVALIDATE              ;INVALIDATE BACKGROUND PATTERN ON 'BANK''
4298 023052                                       LET R2 := BANK
4299 023056  012700  060000                       MOV     #FIRST,R0
4300 023062  010004                               MOV     R0,R4
4301 023064  012701  040000                       MOV     #SIZE,R1
4302 023070  010103                               MOV     R1,R3
4303 023072                                       IF I EQ #1
4304 023102  005737  002424                         TST     NO22BIT
4305 023106  001403                                BEQ     2$
4306 023110  012737  034630  002254                MOV     #MTP034,SUPDOADD
4307 023116  004777  157350                  2$:    CALL    @LINK1
4308 023122                                       END ;OF IF
4309 023122                                       IF I EQ #2
4310 023132  005737  002424                         TST     NO22BIT
4311 023136  001403                                BEQ     3$
4312 023140  012737  034636  002254                MOV     #MTP034+6,SUPDOADD
4313 023146  004737  025322                  3$:    CALL    SUPD03
4314 023152                                       END ;OF IF
4315 023152                                     END ;OF IF
4316 023152                                   END ;OF FOR BANK
4317 023166                                 END ;OF FOR I
4318 023202                                 IF FS7FLAG IS TRUE
4319 023210  005037  002376                   CLR    NOFSMODE
4320 023214  000207                           RETURN
4321 023216                                 END ;OF IF FS7FLAG
4322 023216                                 FOR I := #1 TO #2
4323 023224                                   FOR BANK := LASTBANK DOWNTO #0
4324 023232  004737  042700                     CALL EXBANK
4325 023236                                     IF ACFLAG IS TRUE AND RRFLAG IS FALSE
4326 023252                                       LET R2 := BANK
4327 023256  005102                               COM     R2
4328 023260  012700  060000                       MOV     #FIRST,R0
4329 023264  010004                               MOV     R0,R4
4330 023266  012701  040000                       MOV     #SIZE,R1
4331 023272  010103                               MOV     R1,R3
4332 023274                                       IF I EQ #1
4333 023304  005737  002424                         TST     NO22BIT
```

```
4334 023310  001403                          BEQ     4$
4335 023312  012737  034630  002254          MOV     #MTP034,SUPDOADD
4336 023320  004777  157146              4$: CALL    @LINK1
4337 023324                                  END ;OF IF
4338 023324                                  IF I EQ #2
4339 023334  005737  002424                    TST     NO22BIT
4340 023340  001403                          BEQ     5$
4341 023342  012737  034636  002254          MOV     #MTP034+6,SUPDOADD
4342 023350  004737  025322              5$: CALL    SUPDO3
4343 023354                                  END ;OF IF
4344 023354                              END ;OF IF
4345 023354                          END ;OF FOR BANK
4346 023370                      END ;OF FOR I
4347 023404  005037  002376      CLR     NOFSMODE
4348 023410  000207              RETURN
```

```
 4351 023412                                MT0030: SUBTST  <<MT0030        SETUP FLUSH OUT DBE'S TEST>>
                                                    ;*****************************************************************************
                                                    ;*SUBTEST       MT0030  SETUP FLUSH OUT DBE'S TEST
                                                    ;*****************************************************************************
 4352 023412  005037  002256                               CLR     PASFLG
 4353 023416                                                SET     FULLREL
 4354 023424  012737  000030  002260  MTA030:      MOV     #30,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
 4355 023432  012737  000001  002074                MOV     #1,NOPAR                ;INDICATE COUNT PARITY ERRORS
 4356 023440  005737  002424                        TST     NO22BIT
 4357 023444  001007                                BNE     4$
 4358 023446                                        BMOV    MTP030
 4359 023454  012737  025144  002472                MOV     #SUPD01,LINK1
 4360 023462  000406                                BR      1$
 4361 023464  012737  025322  002472  4$:           MOV     #SUPD03,LINK1
 4362 023472  012737  034412  002254                MOV     #MTP030,SUPDOADD
 4363 023500  104470                    1$:         ECCDIS                          ;DISABLE ERROR CORRECTION
 4364 023502                                        SET     NOFSMODE,NOSCOPE
 4365 023516                                        FOR BANK := #0 TO LASTBANK
 4366 023522  004737  042700                          CALL  EXBANK
 4367 023526                                          IF MKFLAG IS TRUE
 4368 023534                                            IF ACFLAG IS TRUE AND RRFLAG IS FALSE
 4369 023550  012701  040000                              MOV       #SIZE,R1
 4370 023554  012700  060000                              MOV       #FIRST,R0
 4371 023560  004777  156706                              CALL      @LINK1
 4372 023564                                            END ;OF IF ACFLAG
 4373 023564                                          END ;OF IF MKFLAG
 4374 023564                                        END ;OF FOR
 4375 023600                                        IF PASFLG IS FALSE
 4376 023606                                          SET PASFLG
 4377 023614  104502                                  CLRCSR                          ;CLEAR CSRS
 4378 023616  004737  041154                          CALL  RELOCATE
 4379 023622                                          ON.ERROR
 4380 023624  104472                                    ECCINIT               ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
 4381 023626                                            CLEAR       NOFSMODE,NOSCOPE,FULLREL
 4382 023642  000207                                    RETURN
 4383 023644                                          END ;OF ON.ERROR
 4384 023644  013737  002270  002100                  MOV   NEWBANK,BANK
 4385 023652  004737  042700                          CALL  EXBANK
 4386 023656  004737  023424                          CALL  MTA030
 4387 023662  104472                                  ECCINIT                 ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
 4388 023664  004737  042042                          CALL  UNRELOCATE
 4389 023670  000207                                  RETURN
 4390 023672                                        END ;OF IF PASFLG
 4391 023672  104472                                ECCINIT           ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
 4392 023674                                        CLEAR   NOFSMODE,NOSCOPE,FULLREL
 4393 023710  000207                                RETURN
```

```
                                MT0031: SUBTST  <<MT0031        SETUP SOB-A-LONG TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST       MT0031  SETUP SOB-A-LONG TEST
                                ;*****************************************************************************
4397 023712  004737  025110            CALL    KAMITEST                ;CHECK FOR KAMIKAZE MODE
4398 023716                            ON.ERROR THEN $RETURN           ;IF NOT IN KAMIKAZE MODE RETURN
4399 023722                            SET     NOSCOPE
4400 023730  012737  000031  002260    MOV     #31,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4401 023736  005037  002074            CLR     NOPAR                   ;SETUP PARITY ACTION
4402 023742                            MAP     BANK                    ;MAP FIRST SO BLOCK MOVE WORKS
4403 023756                            TESTAREA                        ;ENTER TEST MODE
4404 023764                            BMOV    MTP031,FIRST,SOBLENGTH/2
4405 023776  104417                    KERNEL                          ;ENTER KERNEL MODE
4406 024000  013702  002532            MOV     SOBK,R2
4407 024004  010200                    MOV     R2,R0
4408 024006  012701  100776            MOV     #100776,R1              ;COMPLEMENT OF INSTRUCTION 'SOB R0,DOT'
4409 024012  012705  060056            MOV     #FIRST+SOBLENGTH,R5
4410 024016  012737  060002  002254    MOV     #FIRST+2,SUPDOADD
4411 024024  012737  160000  002472    MOV     #LAST+2,LINK1
4412 024032  005737  002426            TST     NOSUPER
4413 024036  001005                    BNE     1$
4414 024040  023737  172252  172254    CMP     SIPAR5,SIPAR6
4415 024046  001405                    BEQ     2$
4416 024050  000407                    BR      3$
4417 024052  023737  177652  177654  1$:       CMP     UIPAR5,UIPAR6
4418 024060  001003                    BNE     3$
4419 024062  012737  140000  002472  2$:       MOV     #140000,LINK1
4420 024070  004737  025336          3$:       CALL    SUPDO4
4421 024074  005037  002410                    CLR     NOSCOPE
4422 024100  000207                    RETURN
```

```
4425 024102                          MT0032: SUBTST  <<MT0032        SETUP WRITE RECOVERY TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MT0032  SETUP WRITE RECOVERY TEST
                                     ;*****************************************************************************
4426 024102  004737  025110                  CALL    KAMITEST                ;CHECK FOR KAMIKAZE MODE
4427 024106                                  ON.ERROR THEN $RETURN           ;IF NOT IN KAMIKAZE MODE RETURN
4428 024112                                  SET     NOSCOPE
4429 024120  012737  000032  002260          MOV     #32,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4430 024126  005037  002074                  CLR     NOPAR                   ;SETUP PARITY ACTION
4431 024132                                  MAP     BANK                    ;MAP FIRST SO THAT THE BLOCK MOVE WORKS
4432 024146  012700  010247                  MOV     #10247,R0               ;OP CODE OF INSTRUCTION 'MOV    R2,-(PC)'
4433 024152  012701  177667                  MOV     #177667,R1              ;OP CODE OF COMPLEMENT OF INSTRUCTION ''JMP (R0)'
4434 024156  012702  020000                  MOV     #SIZE/2,R2              ;USED FOR 1/2 BANK LOOP
4435 024162  010237  002472                  MOV     R2,LINK1
4436 024166  012703  060000                  MOV     #FIRST,R3
4437 024172  012704  160000                  MOV     #LAST+2,R4
4438 024176  005037  002474                  CLR     LINK2
4439 024202  005737  002426                  TST     NOSUPER
4440 024206  001005                          BNE     1$
4441 024210  023737  172252  172254          CMP     SIPAR5,SIPAR6
4442 024216  001405                          BEQ     2$
4443 024220  000415                          BR      3$
4444 024222  023737  177652  177654  1$:     CMP     UIPAR5,UIPAR6
4445 024230  001011                          BNE     3$
4446 024232  012704  140000          2$:     MOV     #140000,R4
4447 024236  012702  014000                  MOV     #14000,R2
4448 024242  010237  002472                  MOV     R2,LINK1
4449 024246  012737  000001  002474          MOV     #1,LINK2
4450
4451 024254                          3$:     TESTAREA                        ;ENTER TEST MODE
4452                                         ;MOVE TEST TO MEMORY UNDER TEST
4453 024262  010023                  4$:     MOV     R0,(R3)+
4454 024264  010144                          MOV     R1,-(R4)
4455 024266  077203                          SOB     R2,4$
4456
4457 024270  005737  002424                  TST     NO22BIT
4458 024274  001003                          BNE     5$
4459                                         ;MOVE LAST PART OF TEST TO FASTCITY
4460 024276                                  BMOV    MTP032
4461 024304  104417                  5$:     KERNEL                          ;ENTER KERNEL MODE
4462
4463 024306  012702  005141                  MOV     #5141,R2                ;OP CODE OF INSTRUCTION ''COM    -(R1)''
4464 024312  012700  024426                  MOV     #10$,R0                 ;ADDRESS TO RETURN TO IN R0
4465 024316  012701  160000                  MOV     #LAST+2,R1              ;TOP OF BANK
4466 024322  012737  060000  002254          MOV     #FIRST,SUPDOADD
4467 024330  005737  002474                  TST     LINK2
4468 024334  001402                          BEQ     6$
4469 024336  012701  140000                  MOV     #140000,R1
4470 024342  004737  025336          6$:     CALL    SUPD04
4471 024346  012703  020000                  MOV     #SIZE/2,R3
4472 024352  012705  000110                  MOV     #110,R5
4473 024356  012704  060000                  MOV     #FIRST,R4
4474 024362  005737  002474                  TST     LINK2
4475 024366  001402                          BEQ     7$
4476 024370  012703  014000                  MOV     #14000,R3
4477 024374  005737  002424          7$:     TST     NO22BIT
4478 024400  001406                          BEQ     8$
```

```
 4479 024402  012737  034500  002254          MOV     #MTP032,SUPDOADD
 4480 024410  004737  025336                  CALL    SUPD04
 4481 024414  000402                          BR      9$
 4482 024416  004737  025160          8$:     CALL    SUPD02
 4483 024422  005037  002410          9$:     CLR     NOSCOPE
 4484 024426  000207                  10$:    RETURN                  ;THIS RETURN ACTS AS A NORMAL RETURN FROM MT0032
 4485                                                                 ;ALSO A RETURN FROM THE ''CALL   SUPD04'' ABOVE
 4486
```

```
                                MT0033: SUBTST  <<MT0033        SETUP BRANCH GOBBLE TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MT0033  SETUP BRANCH GOBBLE TEST
                                ;*****************************************************************************
                                ;
 4490 024430  004737  025110            CALL    KAMITEST                ;CHECK FOR KAMIKAZE MODE
 4491 024434                            ON.ERROR THEN $RETURN           ;IF NOT IN KAMIKAZE MODE RETURN
 4492 024440                            SET     NOSCOPE
 4493 024446  012737  000033  002260    MOV     #33,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
 4494 024454  005037  002074            CLR     NOPAR                   ;SETUP PARITY ACTION
 4495 024460                            MAP     BANK                    ;MAP FIRST SO THAT BLOCK MOVE WORKS
 4496
 4497 024474                            TESTAREA                        ;ENTER TEST MODE
 4498 024502                            BMOV    MTP033,FIRST,GBLENGTH/2
 4499 024514  104417                    KERNEL                          ;ENTER KERNEL MODE
 4500
 4501 024516  012705  060076            MOV     #FIRST+GBLENGTH,R5
 4502 024522  012737  060004  002254    MOV     #FIRST+4,SUPDOADD
 4503 024530  012701  060002            MOV     #FIRST+2,R1
 4504 024534  012702  060003            MOV     #FIRST+3,R2
 4505 024540  012737  160000  002472    MOV     #LAST+2,LINK1
 4506 024546  005737  002426            TST     NOSUPER
 4507 024552  001005                    BNE     1$
 4508 024554  023737  172252  172254    CMP     SIPAR5,SIPAR6
 4509 024562  001405                    BEQ     2$
 4510 024564  000407                    BR      3$
 4511 024566  023737  177652  177654  1$:       CMP     UIPAR5,UIPAR6
 4512 024574  001003                    BNE     3$
 4513 024576  012737  140000  002472  2$:       MOV     #140000,LINK1
 4514
 4515 024604  004737  025336          3$:       CALL    SUPDO4
 4516 024610  005037  002410                    CLR     NOSCOPE
 4517 024614  000207                    RETURN
 4518
 4519 024616                            MT0034: SUBTST  <<MT0034        SOFT ERROR - BACKGROUND PATTERN TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MT0034  SOFT ERROR - BACKGROUND PATTERN TEST
                                ;*****************************************************************************
                                ;
 4520 024616  012737  000034  002260            MOV     #34,REALPAT
 4521 024624  012700  060000                    MOV     #FIRST,R0
 4522 024630  012701  040000                    MOV     #SIZE,R1
 4523 024634  013702  002560                    MOV     SOFTPAT,R2
 4524 024640  010103                            MOV     R1,R3
 4525 024642  013705  002102                    MOV     BANKINDEX,R5
 4526 024646  010004                            MOV     R0,R4
 4527 024650  005737  002424                    TST     NO22BIT                 ;IS THIS AN 11/44?
 4528 024654  001006                            BNE     1$                      ;BRANCH IF NOT
 4529 024656                                    BMOV    MTP034
 4530 024664  012737  177646  002254            MOV     #UIPAR3,SUPDOADD
 4531 024672                            1$:     IF #BIT13 SET.IN CONFIG+2(R5)
 4532                                      ;BACKGROUND PATTERN IS VALID
 4533 024702  005737  002424                      TST   NO22BIT
 4534 024706  001403                              BEQ   2$
 4535 024710  012737  034636  002254              MOV   #MTP034+6,SUPDOADD
 4536 024716  004737  025322            2$:       CALL  SUPDO3                  ;READ IT
 4537 024722                                    ELSE
 4538                                      ;BACKGROUND PATTERN HAS BEEN INVALIDATED
 4539 024724  005737  002424                      TST   NO22BIT
```

```
4540 024730  001406                          BEQ   3$
4541 024732  012737  034630  002254          MOV   #MTP034,SUPDOADD
4542 024740  004737  025322                  CALL  SUPD03
4543 024744  000402                          BR    4$
4544 024746  004737  025144          3$:     CALL  SUPD01                    ;WRITE IT
4545 024752  052765  020000  002626  4$:     BIS   #BIT13,CONFIG+2(R5)       ;VALIDATE IT
4546 024760                                 END ;OF IF #BIT13
4547 024760  000207                         RETURN
4548
4549 024762                                 MT0035: SUBTST  <<MT0035        SETUP WORST CASE NOISE PARITY TEST>>
                                            ;*****************************************************************************
                                            ;*SUBTEST       MT0035  SETUP WORST CASE NOISE PARITY TEST
                                            ;*****************************************************************************
4550 024762  012737  000035  002260          MOV     #35,REALPAT             ;SET UP TEST NUMBER FOR DISPLAY
4551 024770  013703  002102                  MOV     BANKINDEX,R3
4552 024774  016301  002624                  MOV     CONFIG(R3),R1
4553 025000  000301                          SWAB    R1
4554 025002  042701  177760                  BIC     #^C17,R1
4555 025006  006301                          ASL     R1
4556 025010  010137  002146                  MOV     R1,CSRNO
4557 025014  023737  002146  002502          CMP     CSRNO,PGMCSR
4558 025022  001001                          BNE     1$
4559 025024  000207                          RETURN
4560 025026  012702  052524          1$:     MOV     #52524,R2
4561 025032  004737  035156                  CALL    BACKGND                 ;WRITE BACKROUND OF ALMOST ALT. 1'S AND 0'S
4562 025036  012737  034654  002254          MOV     #MTP035,SUPDOADD
4563 025044  004737  025322                  CALL    SUPD03
4564 025050                                  IF QVFLAG IS TRUE THEN $RETURN
4565 025060  005102                          COM     R2
4566 025062  004737  035156                  CALL    BACKGND                 ;WRITE COMPLEMENT PATTERN INTO MUT
4567 025066  004737  025336                  CALL    SUPD04
4568 025072  000207                         RETURN
```

```
4571 025074                          MT0999: SUBTST  <<MT0999        SETUP NULL TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MT0999  SETUP NULL TEST
                                     ;*****************************************************************************
4572 025074  005037  002260                 CLR     REALPAT
4573 025100                                 SET     NULLFLAG
4574 025106  000207                         RETURN
4575
4576 025110                          KAMITEST:SUBTST <<CHECK FOR KAMIKAZE MODE>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       CHECK FOR KAMIKAZE MODE
                                     ;*****************************************************************************
4577 025110                                 IF KAMIKAZE IS TRUE OR ACTFLAG IS TRUE OR APTFLAG IS TRUE
4578 025132                                   $RETURN NOERROR               ;RUN THE TEST
4579 025136                                 ELSE
4580 025140                                   $RETURN ERROR                 ;DON'T RUN THE TEST
4581 025144                                 END ;OF IF KAMIKAZE
```

```
                                SUPDO1: SUBTST  <<SUBR  EXECUTE PATTERN IN SUPERVISOR>>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    EXECUTE PATTERN IN SUPERVISOR
                                ;*****************************************************************************
                                        MAP     BANK                            ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
004737  054374                SUPDO2: CALL    GETDIS
                                        PUSH    $LPERR,$LPADR
010037  002152                          MOV     R0,SUPDR0
012700  002154                          MOV     #SUPDR1,R0
010120                                  MOV     R1,(R0)+
010220                                  MOV     R2,(R0)+
010320                                  MOV     R3,(R0)+
010420                                  MOV     R4,(R0)+
010520                                  MOV     R5,(R0)+
010620                                  MOV     SP,(R0)+
013700  002152                          MOV     SUPDR0,R0
012737  025240  002562                  MOV     #TAG4$,$LPADR
013737  002562  002564                  MOV     $LPADR,$LPERR
012700  002170                  TAG4$:  MOV     #SUPDR6+2,R0
014006                                  MOV     -(R0),SP
014005                                  MOV     -(R0),R5
014004                                  MOV     -(R0),R4
014003                                  MOV     -(R0),R3
014002                                  MOV     -(R0),R2
014001                                  MOV     -(R0),R1
014000                                  MOV     -(R0),R0
                                        SUPERVISOR                      ;ENTER SUPERVISOR MODE
012706  000740                  MOV     #SUPSTK,SSP
                                CACHOFF                                 ;TURN CACHE OFF
004737  177640                  CALL    FASTCITY                                ;CALL TO THE USER INSTRUCTION PAR'S
                                CACHON                                  ;TURN CACHE ON
                                KERNEL                                  ;ENTER KERNEL MODE
000004                          SCOPE
                                POP     $LPADR,$LPERR
000207                          RETURN
```

```
4618 025322                                SUPD03: MAP     BANK                    ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
4619 025336  004737  054374                SUPD04: CALL    GETDIS
4620 025342                                        PUSH    $LPERR,$LPADR
4621 025352  010037  002152                        MOV     R0,SUPDR0
4622 025356  012700  002154                        MOV     #SUPDR1,R0
4623 025362  010120                                MOV     R1,(R0)+
4624 025364  010220                                MOV     R2,(R0)+
4625 025366  010320                                MOV     R3,(R0)+
4626 025370  010420                                MOV     R4,(R0)+
4627 025372  010520                                MOV     R5,(R0)+
4628 025374  010620                                MOV     SP,(R0)+
4629 025376  013700  002152                        MOV     SUPDR0,R0
4630 025402  012737  025416  002562                MOV     #TBG4$,$LPADR
4631 025410  013737  002562  002564                MOV     $LPADR,$LPERR
4632 025416  012700  002170                TBG4$:  MOV     #SUPDR6+2,R0
4633 025422  014006                                MOV     -(R0),SP
4634 025424  014005                                MOV     -(R0),R5
4635 025426  014004                                MOV     -(R0),R4
4636 025430  014003                                MOV     -(R0),R3
4637 025432  014002                                MOV     -(R0),R2
4638 025434  014001                                MOV     -(R0),R1
4639 025436  014000                                MOV     -(R0),R0
4640 025440                                        TESTAREA                        ;ENTER SUPERVISOR MODE
4641 025446  005737  002426                        TST     NOSUPER
4642 025452  001403                                BEQ     1$
4643 025454  012706  000700                        MOV     #USESTK,USP
4644 025460  000402                                BR      2$
4645 025462  012706  000740                1$:     MOV     #SUPSTK,SSP
4646 025466  104424                        2$:     CACHOFF                         ;TURN CACHE OFF
4647 025470  004777  154560                        CALL    @SUPDOADD
4648 025474  104423                                CACHON                          ;TURN CACHE ON
4649 025476  104417                                KERNEL                          ;ENTER KERNEL MODE
4650 025500  000004                                SCOPE
4651 025502                                        POP     $LPADR,$LPERR
4652 025512  000207                                RETURN
```

```
                        .SBTTL  MEMORY TEST PATTERN ROUTINES
                ;*******************************************************************************
                ; PATTERN REGISTER CONVENTIONS
                ;       R0      FIRST ADDRESS OF PATTERN (FIRST,LAST+2,ETC)
                ;       R1      NUMBER OF ADDRESSES IN PATTERN (SIZE)
                ;       R2      DATA FOR PATTERN (ONES,52525,ETC)
                ;       R3      COPY OF R1 (IF NECESSARY)
                ;       R4      COPY OF R0 (IF NECESSARY)
                ;       R5      COPY OF R2 (IF NECESSARY)
                ;*******************************************************************************
025514          MTP000: SUBTST  <<MTP000        BASIC DATA TEST>>
                ;*******************************************************************************
                ;*SUBTEST       MTP000  BASIC DATA TEST
                ;*******************************************************************************
025514  010220  1$:     MOV     R2,(R0)+        ;V177640
025516  077102          SOB     R1,MTP000       ;V177642
025520  000240          NOP                     ;V177644
025522  012401  2$:     MOV     (R4)+,R1        ;V177646
025524  020102          CMP     R1,R2           ;V177650
025526  001402          BEQ     3$              ;V177652
025530  104430          PERR02                  ;V177654
025532  000240          NOP                     ;V177656
025534  077306  3$:     SOB     R3,2$           ;V177660
025536  000207          RETURN                  ;V177662
025540          MTP001: SUBTST  <<MTP001        ADDRESS TEST>>
                ;*******************************************************************************
                ;*SUBTEST       MTP001  ADDRESS TEST
                ;*******************************************************************************
025540  010220  3$:     MOV     R2,(R0)+        ;V177640
025542  062702  000002  ADD     #2,R2           ;V177642
025546  077104          SOB     R1,3$           ;V177646
025550  000240          NOP                     ;V177650
025552  012400  1$:     MOV     (R4)+,R0        ;V177652
025554  020005          CMP     R0,R5           ;V177654
025556  001401          BEQ     2$              ;V177656
025560  104427          PERR01                  ;V177660
025562  062705  000002  2$:     ADD     #2,R5           ;V177662
025566  077307          SOB     R3,1$           ;V177666
025570  000207          RETURN                  ;V177672
025572          MTP002: SUBTST  <<MTP002        COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)>>
                ;*******************************************************************************
                ;*SUBTEST       MTP002  COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)
                ;*******************************************************************************
025572  010540  3$:     MOV     R5,-(R0)        ;V177640
025574  062705  000002  ADD     #2,R5           ;V177642
025600  077104          SOB     R1,3$           ;V177646
025602  000240          NOP                     ;V177650
025604  162702  000002  1$:     SUB     #2,R2           ;V177652
025610  012401          MOV     (R4)+,R1        ;V177656
025612  020102          CMP     R1,R2           ;V177660
025614  001401          BEQ     2$              ;V177662
025616  104430          PERR02                  ;V177664
025620  077307  2$:     SOB     R3,1$           ;V177666
025622  000207          RETURN                  ;V177670
```

```
                                MTPA03: SUBTST  <<MTPA03        3 XOR 9 WORST CASE NOISE TEST (WRITE)>>
                                ;*********************************************************************************
                                ;*SUBTEST        MTPA03  3 XOR 9 WORST CASE NOISE TEST (WRITE)
                                ;*********************************************************************************
                                        ;R1 = ADDRESS
                                        ;R2 = SMALL LOOP CONSTANT
                                        ;R3 = NUM OF ADD TO TEST (LARGE LOOP)
                                        ;R4 = GOOD DATA
                                        ;R5 = MEDIUM LOOP CONSTANT
                                        .ENABL  LSB
025624  010421                  1$:    MOV     R4,(R1)+        ;V177640
025626  010421                         MOV     R4,(R1)+        ;V177642
025630  077203                         SOB     R2,1$           ;V177644
025632  005104                         COM     R4              ;V177646
025634  052704                         BIS     (PC)+,R4        ;V177650
025636  000401                  WARN2: 401                     ;V177652 WARNING LOCATION IS MODIFIED BEFORE LOADING
025640  012702  000004                 MOV     #4,R2           ;V177654
025644  077511                         SOB     R5,1$           ;V177660
025646  005104                         COM     R4              ;V177662
025650  052704                         BIS     (PC)+,R4        ;V177664
025652  000401                  WARN3: 401                     ;V177666 WARNING LOCATION IS MODIFIED BEFORE LOADING
025654  012705  000100                 MOV     #64.,R5         ;V177670
025660  077317                         SOB     R3,1$           ;V177674
025662  000207                         RETURN                  ;V177676
                                        .DSABL  LSB

025664                          MTPB03: SUBTST  <<MTPB03        3 XOR 9 WORST CASE NOISE TEST (READ)>>
                                ;*********************************************************************************
                                ;*SUBTEST        MTPB03  3 XOR 9 WORST CASE NOISE TEST (READ)
                                ;*********************************************************************************
                                        .ENABL  LSB
025664  000137  025724          1$:    JMP     @#MTPC03        ;V177640        GO TO V172360
025670  077203                         SOB     R2,1$           ;V177644
025672  005104                         COM     R4              ;V177646
025674  052704                         BIS     (PC)+,R4        ;V177650
025676  000401                  WARN4: 401                     ;V177652 WARNING LOCATION IS MODIFIED BEFORE LOADING
025700  012702  000004                 MOV     #4,R2           ;V177654
025704  077511                         SOB     R5,1$           ;V177660
025706  005104                         COM     R4              ;V177662
025710  052704                         BIS     (PC)+,R4        ;V177664
025712  000401                  WARN5: 401                     ;V177666 WARNING LOCATION IS MODIFIED BEFORE LOADING
025714  012705  000100                 MOV     #64.,R5         ;V177670
025720  077317                         SOB     R3,1$           ;V177674
025722  000207                         RETURN                  ;V177676
                                        .DSABL  LSB
```

```
 4743 025724                          MTPC03: SUBTST  <<MTPC03        TEST DATA SUBPROGRAM>>
                                      ;*********************************************************************************
                                      ;*SUBTEST       MTPC03  TEST DATA SUBPROGRAM
                                      ;*********************************************************************************
 4744 025724  020421                          CMP     R4,(R1)+        ;V172360
 4745 025726  001401                          BEQ     1$              ;V172362
 4746 025730  104431                          PERR03                  ;V172364
 4747 025732  005141                  1$:     COM     -(R1)           ;V172366
 4748 025734  005111                          COM     (R1)            ;V172370
 4749 025736  000137  025742                  JMP     @#MTPD03        ;V172372        GO TO V172260
 4750
 4751 025742                          MTPD03: SUBTST  <<MTPD03        TEST DATA SUBSUBPROGRAM>>
                                      ;*********************************************************************************
                                      ;*SUBTEST       MTPD03  TEST DATA SUBSUBPROGRAM
                                      ;*********************************************************************************
 4752 025742  020421                          CMP     R4,(R1)+        ;V172260
 4753 025744  001401                          BEQ     1$              ;V172262
 4754 025746  104431                          PERR03                  ;V172264
 4755 025750  005127                  1$:     COM     (PC)+           ;V172266
 4756 025752  000000                          0                       ;V172270
 4757 025754  001363                          BNE     MTPC03          ;V172272        GO TO V172360
 4758 025756  000137  025670                  JMP     @#MTPB03+4      ;V172274        GO TO V177644
```

```
                                MTPA04: SUBTST  <<MTPA04        ROTATING ZEROS TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MTPA04  ROTATING ZEROS TEST
                                ;*****************************************************************************
4762 025762  012705  000010    1$:     MOV     #8.,R5          ;V177640
4763 025766  010504            1$:     MOV     R5,R4           ;V177644
4764 025770  000241            1$:     CLC                     ;V177646
4765 025772  000137  026016    1$:     JMP     @#MTPB04        ;V177650
4766 025776  016004  177776    1$:     MOV     -2(R0),R4       ;V177654
4767 026002  103402            1$:     BCS     2$              ;V177660
4768 026004  020204            1$:     CMP     R2,R4           ;V177662
4769 026006  001401            1$:     BEQ     3$              ;V177664
4770 026010  104432            2$:     PERR04                  ;V177666
4771 026012  077115            3$:     SOB     R1,1$           ;V177670
4772 026014  000207            1$:     RETURN                  ;V177672
4773
4774 026016                    MTPB04: SUBTST  <<MTPB04        SUBR    ROTATING BIT>>
                                ;*****************************************************************************
                                ;*SUBTEST        MTPB04  SUBR    ROTATING BIT
                                ;*****************************************************************************
4775 026016  106110            1$:     ROLB    (R0)            ;V172360
4776 026020  077502            1$:     SOB     R5,1$           ;V172362
4777 026022  106120            1$:     ROLB    (R0)+           ;V172364
4778 026024  106110            2$:     ROLB    (R0)            ;V172366
4779 026026  077402            2$:     SOB     R4,2$           ;V172370
4780 026030  106120            1$:     ROLB    (R0)+           ;V172372
4781 026032  000137  025776    1$:     JMP     @#MTPA04+14     ;V172374
4782
4783 026036                    MTP005: SUBTST  <<MTP005        ROTATION ONES TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MTP005  ROTATION ONES TEST
                                ;*****************************************************************************
4784 026036  012705  00001C    1$:     MOV     #8.,R5          ;V177640
4785 026042  010504            1$:     MOV     R5,R4           ;V177644
4786 026044  000261            1$:     SEC                     ;V177646
4787 026046  000137  026016    1$:     JMP     @#MTPB04        ;V177650
4788 026052  016004  177776    1$:     MOV     -2(R0),R4       ;V177654
4789 026056  103002            1$:     BCC     2$              ;V177660 IF THIS HAPPENS THE GOOD & BAD MATCH
4790 026060  020204            1$:     CMP     R2,R4           ;V177662
4791 026062  001401            1$:     BEQ     3$              ;V177664
4792 026064  104432            2$:     PERR04                  ;V177666
4793 026066  077115            3$:     SOB     R1,1$           ;V177670
4794 026070  000207            1$:     RETURN                  ;V177672
```

```
 4797 026072                            MTP006: SUBTST  <<MTP006        INITIAL DATA TEST>>
                                        ;*****************************************************************************
                                        ;*SUBTEST        MTP006  INITIAL DATA TEST
                                        ;*****************************************************************************
 4798                                   ;       THIS TEST CHECKS THE DI/DO LINES BY
 4799                                   ;       SHIFTING A 1 THROUGH THE WORD.
 4800 026072  012737  000001  002234           MOV     #1,DATBUF       ;SET THE FIRST TEST BIT
 4801 026100  005037  002236                   CLR     DATBUF+2        ;CLEAR 2ND WORD
 4802 026104  013771  002234  000000  1$:      MOV     DATBUF,@(R1)    ;WRITE TEST WORD 1
 4803 026112  013771  002236  000002          MOV     DATBUF+2,@2(R1) ;AND TEST WORD 2
 4804 026120  017102  000000                   MOV     @(R1),R2
 4805 026124  023702  002234                   CMP     DATBUF,R2       ;NOW READ THEM
 4806 026130  001401                           BEQ     2$              ;BR IF FIRST 16 OK
 4807 026132  104433                           PERR07                  ;ERROR TRAP
 4808
 4809 026134  017102  000002          2$:     MOV     @2(R1),R2
 4810 026140  023702  002236                   CMP     DATBUF+2,R2     ;NOW READ SECOND WORD
 4811 026144  001401                           BEQ     3$              ;BR IF OK
 4812 026146  104434                           PERR10                  ;ERROR TRAP
 4813
 4814 026150  005737  002236          3$:     TST     DATBUF+2        ;HAS LAST BIT BEEN TESTED ?
 4815 026154  100405                           BMI     4$              ;MINUS MEANS BIT 31
 4816 026156                                   DLEFT   DATBUF          ;NO, SHIFT TEST BIT LEFT
 4817 026166  000746                           BR      1$              ;GO WRITE NEW TEST DATA
 4818                                                                  ;NOW GOING TO SHIFT A 0 IN DATA DIRECTION
 4819 026170  012737  177776  002234  4$:     MOV     #177776,DATBUF  ;PUT A 0 IN BIT 0
 4820 026176  012737  177777  002236          MOV     #-1, DATBUF+2   ;AND 1'S IN ALL OTHERS
 4821 026204  013771  002234  000000  5$:     MOV     DATBUF,@(R1)    ;WRITE THE DATA
 4822 026212  013771  002236  000002          MOV     DATBUF+2,@2(R1) ;2 WORDS WORTH
 4823 026220  017102  000000                   MOV     @(R1),R2
 4824 026224  023702  002234                   CMP     DATBUF,R2       ;NOW READ FIRST WORD
 4825 026230  001401                           BEQ     6$              ;BR IF OK
 4826 026232  104433                           PERR07
 4827
 4828 026234  017102  000002          6$:     MOV     @2(R1),R2
 4829 026240  023702  002236                   CMP     DATBUF+2,R2     ;NOW, READ SECOND WORD
 4830 026244  001401                           BEQ     7$              ;BR IF OK
 4831 026246  104434                           PERR10
 4832
 4833 026250  005737  002236          7$:     TST     DATBUF+2        ;TESTED BIT 31 YET?
 4834 026254  100005                           BPL     8$              ;BR IF YES, WE'RE DONE
 4835 026256                                   DLEFT   DATBUF
 4836 026266  000746                           BR      5$              ;KEEP GOING
 4837 026270  000207          8$:     RETURN
```

```
                                MTP007: SUBTST  <<MTP007        ADDRESS BIT TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MTP007  ADDRESS BIT TEST
                                ;*****************************************************************************
                                ;               THIS TEST CHECKS TO SEE THAT EACH ADDRESS
                                ;               BIT IN EACH 16K BANK CAN BE ASSERTED UNIQUELY.
                                ;               IT CHECKS FOR ADDRESS BITS THAT MAY BE STUCK
                                ;               HIGH, STUCK LOW OR STUCK TOGETHER.
026272  111100                          MOVB    (R1),R0
026274  105700                          TSTB    R0              ;READ AND COMPARE FOR ZEROS
026276  001401                          BEQ     1$              ;BR IF OK
026300  104435                          PERR11
026302  105111                  1$:     COMB    (R1)            ;COMPLEMENT THE BYTE
026304  111100                          MOVB    (R1),R0
026306  105700                          TSTB    R0              ;READ FOR NON ZEROS
026310  001001                          BNE     2$              ;BR IF OK
026312  104436                          PERR12
026314  040201                  2$:     BIC     R2,R1           ;MASK OFF THE ASSERTED BIT
026316  006302                          ASL     R2              ;SHIFT R2 FOR NEXT BIT
026320  050201                          BIS     R2,R1           ;SET THE NEW BIT INTO R1
026322  011100                          MOV     (R1),R0
026324  005700                          TST     R0              ;READ THE NEW ADDRESS
026326  001401                          BEQ     3$              ;READ FOR ZEROS
026330  104437                          PERR13
026332  005111                  3$:     COM     (R1)            ;COMPL THE WORD
026334  011100                          MOV     (R1),R0
026336  005700                          TST     R0              ;READ IT AGAIN
026340  001001                          BNE     4$
026342  104440                          PERR14
026344  022702  100000          4$:     CMP     #100000,R2
026350  001407                          BEQ     5$
026352  022702  010000                  CMP     #10000,R2       ;CHECK FOR MSB IN 4K BANK
026356  001356                          BNE     2$              ;NOT LAST BIT, BRANCH
026360  006302                          ASL     R2
026362  012701  160000                  MOV     #160000,R1
026366  000752                          BR      2$
026370  000207                  5$:     RETURN
```

```
                                MTP010: SUBTST  <<MTP010        BYTE ADDRESSING TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MTP010  BYTE ADDRESSING TEST
                                ;*****************************************************************************
                                        ;TEST 3 THIS TEST CHECKS FOR PROPER
                                        ;       BYTE ADDRESSING WITH ECC DISABLED
026372  010402                        MOV     R4,R2           ;R4 HAS LOWEST ADDRESS
026374  010403                        MOV     R4,R3           ;PUT IT IN R3 ALSO
026376  062702  000004                ADD     #4,R2           ;POINT R2 TO LAST BYTE +1
026402  012713  177777                MOV     #-1,(R3)        ;WRITE ALL ONES IN
026406  012763  177777  000002        MOV     #-1,2(R3)       ;THE 4 TEST BYTES
026414  105013                1$:     CLRB    (R3)            ;CLEAR A BYTE
026416  010401                        MOV     R4,R1           ;INITIALIZE R1 FOR EACH PASS
026420  020201                2$:     CMP     R2,R1           ;IF EQUAL, JUST READ LAST BYTE
026422  001420                        BEQ     6$              ;BR IF EQUAL
026424  020301                        CMP     R3,R1           ;IS THIS THE BYTE OF ZEROS
026426  001007                        BNE     4$              ;BR IF NOT
026430  111100                        MOVB    (R1),R0
                                ;WARNING IF YOU OPTOMIZE CHANGE THE PCBUMP FOR THIS ERROR INCASE OF TRAPS
026432  022700  000000                CMP     #0,R0           ;IT IS, COMPARE FOR ZEROS
026436  001401                        BEQ     3$
026440  104435                        PERR11
026442  005201                3$:     INC     R1              ;NEXT BYTE
026444  000765                        BR      2$              ;RETURN
026446  111100                4$:     MOVB    (R1),R0
026450  122700  177777                CMPB    #-1,R0          ;ITS NOT THE BYTE OF 0'S, READ 1'S
026454  001401                        BEQ     5$
026456  104436                        PERR12
026460  005201                5$:     INC     R1              ;MOVE TO NEXT BYTE
026462  000756                        BR      2$
026464  112713  177777        6$:     MOVB    #-1,(R3)        ;RESTORE 1'S TO BYTE JUST TESTED
026470  005203                        INC     R3              ;INC TO NEXT BYTE
026472  020302                        CMP     R3,R2           ;WAS THAT JUST THE LAST ONE?
026474  001347                        BNE     1$              ;BR IF NO
026476  000207                        RETURN
```

```
                                MTP011: SUBTST  <<MTP011        SINGLE BIT ERROR TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MTP011  SINGLE BIT ERROR TEST
                                ;*****************************************************************************
                                        ;(1)    CREATE A SINGLE BIT ERROR
                                        ;
                                        ;(2)    READ BACK SBE UNCORRECTED (WITH ECC DISABLE)
                                        ;
                                        ;(3)    ENABLE ECC & READ CORRECTED DATA
                                        ;
                                        ;(4)    CHECK THAT THE SBE FLAG WAS SET FROM THE LAST READ
                                        ;
                                        ;(5)    DO (1-4) FOR DATA CONSISTING OF 1 BIT SET IN EACH OF 32
                                        ;       POSITIONS OF A DOUBLE WORD
                                        ;       THEN DO IT AGAIN FOR 1 BIT CLEARED IN EACH OF 32 POSITIONS OF
                                        ;       A DOUBLE WORD
                                        ;       IE (64 TIMES)
                                        ;
                                        ;(6)    DO (1-5) FOR A SBE IN EACH OF 32 BIT POSITIONS
                                        ;       IE (RUN TEST 64 * 32 = 2048 TIMES)
026500  104503                          CLR1CSR                 ;CLEAR 1 SELECTED CSR
                                        ;BIG LOOP
026502  012737  000001  002234  MTLA11: MOV     #1,DATBUF       ;INITIAL DATA
026510  005037  002236                  CLR     DATBUF+2        ;32 BITS WORTH
                                        ;MEDIUM LOOP
026514  012737  000001  002244  MTLB11: MOV     #1,SBEMSK       ;INITIAL ERROR MASK
026522  005037  002246                  CLR     SBEMSK+2        ;32 BITS WORTH
                                        ;LITTLE LOOP
026526  013737  002234  002240  MTLC11: MOV     DATBUF,TSTDAT   ;
026534  013737  002236  002242          MOV     DATBUF+2,TSTDAT+2;TO SAVE ORIG DATA
026542  105737  002256                  TSTB    PASFLG  ;COMP DATA ON SECOND PASSONLY
026546  001404                          BEQ     4$              ;BR IF FIRST PASS
026550  005137  002240                  COM     TSTDAT          ;SECOND PASS, COMP BOTH WORDS
026554  005137  002242                  COM     TSTDAT+2
026560  013702  002240          4$:     MOV     TSTDAT,R2
026564  013703  002242                  MOV     TSTDAT+2,R3
026570  012737  002240  002272          MOV     #TSTDAT,SOURCE  ;SET UP ADDRESS FOR CHKGEN
026576  004737  040360                  CALL    CHKGEN          ;GEN CHECKBITS ON TSTDAT
                                        ;*********************************
                                        ;** CREATE A SINGLE BIT ERROR **
                                        ;*********************************
026602  013701  002244                  MOV     SBEMSK,R1
026606  074137  002240                  XOR     R1,TSTDAT
026612  013701  002246                  MOV     SBEMSK+2,R1
026616  074137  002242                  XOR     R1,TSTDAT+2
026622  013701  002362          MTLD11: MOV     TESTADD,R1      ;FIRST TEST ADDRESS
026626  013705  002364                  MOV     TESTADD+2,R5    ;SECOND TEST ADDRESS
026632  104471                          ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
026634  013711  002240                  MOV     TSTDAT,(R1)     ;WRITE FIRST 16 BITS
026640  104475                          CB1CSR                  ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
026642  013715  002242                  MOV     TSTDAT+2,(R5)   ;WRITE SECOND 16 BITS AND
                                                                ;CHECK BITS. WE NOW HAVE CHECKBITS
                                                                ;GENERATED ON DATBUF AND DATA WITH
                                                                ;ONE BIT IN ERROR (AS PER SBEMSK).
026646  104471                          ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
026650  011100                          MOV     (R1),R0
026652  020037  002240                  CMP     R0,TSTDAT       ;READ THE LOW WORD (UNCORRECTED)
```

```
4970 026656  001403                    BEQ     6$                  ;BR IF OK
4971 026660  010137  002032              MOV     R1,ADDRESS
4972 026664  104455                    PERR31
4973
4974 026666  011500              6$:   MOV     (R5),R0
4975 026670  020037  002242            CMP     R0,TSTDAT+2         ;READ THE HIGH WORD (UNCORRECTED)
4976 026674  001403                    BEQ     7$                  ;BR IF OK
4977 026676  010537  002032              MOV     R5,ADDRESS
4978 026702  104455                    PERR31
4979
4980 026704              7$:           IF KFLAG IS FALSE
4981 026712  104426                      READCSR
4982 026714                               IF #BIT4 OFF.IN CSR OR #BIT15 OFF.IN CSR
4983 026734  104045                        ERROR       +45
4984 026736                               END; OF IF #BIT4
4985 026736                           END; OF IF KFLAG
4986 026736  104512                    ERRGEN
4987 026740  104503                    CLR1CSR                     ;CLEAR 1 SELECTED CSR
4988 026742  011100                    MOV     (R1),R0
4989 026744  020002                    CMP     R0,R2               ;SEE IF ITS BEEN CORRECTED
4990 026746  001401                    BEQ     8$                  ;IT SHOULD HAVE BEEN
4991 026750  104456                    PERR32
4992
4993 026752  104510              8$:   TSTREAD                     ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
4994 026754  103411                    BCS     9$                  ;BR IF IT IS SET
4995 026756                            SET     HEADER              ;ENABLE PRINTING OF ERROR HEADER INFO
4996 026764  010137  002032              MOV     R1,ADDRESS
4997 026770  104460                    PERR34
4998 026772                            SET     HEADER              ;ENABLE PRINTING OF ERROR HEADER INFO
4999
5000 027000  104503              9$:   CLR1CSR                     ;CLEAR 1 SELECTED CSR
5001 027002  011500                    MOV     (R5),R0
5002 027004  020003                    CMP     R0,R3               ;SEE IF ITS BEEN CORRECTED
5003 027006  001401                    BEQ     10$                 ;BR IF OK
5004 027010  104456                    PERR32
5005
5006 027012  104510             10$:   TSTREAD                     ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
5007 027014  103411                    BCS     11$                 ;BR IF YES
5008 027016                            SET     HEADER              ;ENABLE PRINTING OF ERROR HEADER INFO
5009 027024  010137  002032              MOV     R1,ADDRESS
5010 027030  104460                    PERR34
5011 027032                            SET     HEADER              ;ENABLE PRINTING OF ERROR HEADER INFO
5012 027040  104512             11$:   ERRGEN                      ;TEST ERROR ADDRESS
5013 027042  105737  002256            TSTB    PASFLG
5014 027046  100452                    BMI     15$
5015 027050  005737  002246            TST     SBEMSK+2            ;TEST FOR LAST MASK BIT
5016 027054  100405                    BMI     12$                 ;MINUS MEANS BIT 31
5017 027056                            DLEFT   SBEMSK
5018 027066  000617                    BR      MTLC11
5019 027070             12$:           IF #SW11 SET.IN @SWR THEN GOTO 13$
5020 027100                            IF QVFLAG IS TRUE THEN GOTO 13$
5021 027106  005737  002236            TST     DATBUF+2            ;LAST DATA BIT ?
5022 027112  100406                    BMI     13$                 ;WHICH IS BIT 31
5023 027114                            DLEFT   DATBUF
5024 027124  000137  026514            JMP     MTLB11
5025 027130  105737  002256  13$:      TSTB    PASFLG  ;FIRST OR SECOND PASS ?
5026 027134  001004                    BNE     14$                 ;NON ZERO MEANS WE'RE DONE
```

```
5027 027136  105237  002256                    INCB    PASFLG  ;NOT DONE, GO DO SECOND PASS
5028 027142  000137  026502                    JMP     MTLA11
5029 027146  052737  000200  002256  14$:      BIS     #BIT7,PASFLG
5030 027154  005002                            CLR     R2
5031 027156  005003                            CLR     R3
5032 027160  005037  002240                    CLR     TSTDAT
5033 027164  005037  002242                    CLR     TSTDAT+2
5034 027170  012704  000040                    MOV     #40,R4
5035 027174  012737  003740  002274  15$:      MOV     #3740,CHECK
5036 027202  074437  002274                    XOR     R4,CHECK
5037 027206  006304                            ASL     R4
5038 027210  032704  020000                    BIT     #BIT13,R4
5039 027214  001002                            BNE     16$
5040 027216  000137  026622                    JMP     MTLD11
5041                                           ;CLEAR OUT ANY DBE'S OR SBE'S
5042 027222  104471                    16$:    ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
5043 027224  013701  002362                    MOV     TESTADD,R1
5044 027230  013705  002364                    MOV     TESTADD+2,R5
5045 027234                                    CLEAR   (R1),(R5)
5046 027240  104503                            CLR1CSR                 ;CLEAR 1 SELECTED CSR
5047 027242  000207                            RETURN
```

```
                        MTP012: SUBTST  <<MTP012        WRITE BYTE CLEARS SBE TEST>>
                        ;*****************************************************************************
                        ;*SUBTEST       MTP012  WRITE BYTE CLEARS SBE TEST
                        ;*****************************************************************************
                                        ;SINGLE BIT ERROR TEST TO INSURE THAT A WRITE
                                        ;BYTE CLEARS SINGLE BIT ERRORS.
027244  104503                  CLR1CSR                 ;CLEAR 1 SELECTED CSR
027246  012737  000001  002234  MOV     #1,DATBUF       ;INITIAL DATA
027254  005037  002236          CLR     DATBUF+2        ;32 BITS WORTH
027260  012737  000001  002244  1$:     MOV     #1,SBEMSK       ;INITIAL ERROR MASK
027266  005037  002246          CLR     SBEMSK+2        ;32 BITS WORTH
027272  013737  002234  002240  2$:     MOV     DATBUF,TSTDAT   ;SAVE ORIGINAL DATA
027300  013737  002236  002242          MOV     DATBUF+2,TSTDAT+2;BOTH WORDS
027306  012737  002240  002272          MOV     #TSTDAT,SOURCE  ;NEED ADDRESS FOR CHKGEN
027314  004737  040360                  CALL    CHKGEN          ;GENERATE CHECK BITS
027320  013701  002244                  MOV     SBEMSK,R1
027324  074137  002240                  XOR     R1,TSTDAT
027330  013701  002246                  MOV     SBEMSK+2,R1
027334  074137  002242                  XOR     R1,TSTDAT+2
027340  013704  002362                  MOV     TESTADD,R4      ;FIRST TEST ADDRESS
027344  010401                          MOV     R4,R1           ;PUT IT IN R1 ALSO
027346  104471                          ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
027350  013711  002240                  MOV     TSTDAT,(R1)     ;WRITE 16 BITS
027354  104475                          CB1CSR                  ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
027356  060501                          ADD     R5,R1           ;INDEX UP TO SECOND WORD
027360  013711  002242                  MOV     TSTDAT+2,(R1)   ;WRITE HIGH WORD+CHECKBITS
027364  104503                          CLR1CSR                 ;CLEAR 1 SELECTED CSR
                                                                ;IT'S DANGEROUS IF WE DON'T
027366  012702  002244                  MOV     #SBEMSK,R2      ;ADDRESS OF ERROR MASK
027372  160501                          SUB     R5,R1           ;RETURN TO FIRST WORD
027374  112711  177777          3$:     MOVB    #-1,(R1)        ;WRITE A BYTE OF 1'S
027400  005737  002500                  TST     KFLAG           ;IS THIS MF11S-K
027404  001403                          BEQ     4$              ;BRANCH IF NOT - IT'S MS11-M
027406  132712  177777                  BITB    #-1,(R2)        ;DID THIS BYTE HAVE THE BAD BIT IN IT?
027412  001420                          BEQ     6$              ;NO - BRANCH
027414  104510                  4$:     TSTREAD                 ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
027416  103011                          BCC     5$              ;NO - SKIP
027420                                  SET     HEADER          ;ENABLE PRINTING OF ERROR HEADER INFO
027426  010137  002032                  MOV     R1,ADDRESS
027432  104017                          ERROR   +17
027434                                  SET     HEADER          ;ENABLE PRINTING OF ERROR HEADER INFO

027442  111100                  5$:     MOVB    (R1),R0
027444  122700  177777                  CMPB    #-1,R0          ;CHECK DATA
027450  001414                          BEQ     7$              ;BR IF OK
027452  104457                          PERR33

027454  104510                  6$:     TSTREAD                 ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
                                                                ;READ THE BYTE
                                                                ;SBE ERROR BIT ONLY SET ?
027456  103771                          BCS     5$              ;SHOULD BE SET, BR IF OK
027460                                  SET     HEADER          ;ENABLE PRINTING OF ERROR HEADER INFO
027466  010137  002032                  MOV     R1,ADDRESS
027472  104460                          PERR34
027474                                  SET     HEADER          ;ENABLE PRINTING OF ERROR HEADER INFO

027502  132712  177777          7$:     BITB    #-1,(R2)        ;CHECK FOR LAST BYTE
```

```
5104 027506  001012                        BNE     8$              ;
5105 027510  005202                        INC     R2
5106 027512  005201                        INC     R1              ;MOVE TO NEXT BYTE
5107 027514  013704  002362                MOV     TESTADD,R4      ;FIRST TEST ADDRESS
5108 027520  032701  000002                BIT     #2,R1           ;TEST FOR LOWER WORD
5109 027524  001723                        BEQ     3$              ;BR IF IT'S LOW 16 BITS
5110 027526  062704  000002                ADD     #2,R4           ;ADJUST POINTER FOR ERROR REPT.
5111 027532  000720                        BR      3$
5112 027534  005737  002246          8$:   TST     SBEMSK+2        ;LAST ERROR BIT ?
5113 027540  100405                        BMI     9$              ;MINUS MEANS BIT 31
5114 027542                                DLEFT   SBEMSK
5115 027552  000647                        BR      2$
5116 027554                          9$:   IF #SW11 SET.IN @SWR THEN GOTO 10$
5117 027564                                IF QVFLAG IS TRUE THEN GOTO 10$
5118 027572  005737  002236                TST     DATBUF+2        ;LAST DATA BIT?
5119 027576  100405                        BMI     10$             ;MINUS = BIT 31
5120 027600                                DLEFT   DATBUF
5121 027610  000623                        BR      1$
5122                                       ;CLEAR OUT ANY DBE'S OR SBE'S
5123 027612  104471                 10$:   ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
5124 027614  013701  002362                MOV     TESTADD,R1
5125 027620  005011                        CLR     (R1)
5126 027622  060501                        ADD     R5,R1
5127 027624  005011                        CLR     (R1)
5128 027626  104503                        CLR1CSR                 ;CLEAR 1 SELECTED CSR
5129 027630  000207                        RETURN
```

.

.

```
5132 027632                        MTP013: SUBTST  <<MTP013        CREATE DOUBLE BIT ERROR TEST>>
                                   ;*****************************************************************************
                                   ;*SUBTEST       MTP013  CREATE DOUBLE BIT ERROR TEST
                                   ;*****************************************************************************
5133                                       ;DOUBLE BIT ERROR FORCE TO CHECK DOUBLE ERROR LOGIC
5134 027632  104503                  CLR1CSR                    ;CLEAR 1 SELECTED CSR
5135 027634  012701  002362          MOV     #TESTADD,R1
5136 027640  005037  002234    1$:   CLR     DATBUF             ;MAKE INITIAL DATA
5137 027644  005037  002236          CLR     DATBUF+2           ;ALL ZEROS
5138 027650  012737  000001  002244 2$:    MOV     #1,SBEMSK          ;INITIAL SINGLE ERROR MASK
5139 027656  005037  002246          CLR     SBEMSK+2           ;SECOND WORD
5140 027662  012737  000001  002250 3$:    MOV     #1,DBEMSK          ;INITIAL DOUBLE ERROR MASK
5141 027670  005037  002252          CLR     DBEMSK+2           ;32 BITS HERE ALSO
5142 027674  013737  002234  002240 4$:    MOV     DATBUF,TSTDAT      ;
5143 027702  013737  002236  002242       MOV     DATBUF+2,TSTDAT+2
5144 027710  105737  002256             TSTB    PASFLG  ;NO COMPLEMENTING FIRST PASS
5145 027714  001404                     BEQ     5$
5146 027716  005137  002240             COM     TSTDAT             ;COMP FIRST WORD
5147 027722  005137  002242             COM     TSTDAT+2           ;SECOND WORD
5148 027726  104503              5$:    CLR1CSR                    ;CLEAR 1 SELECTED CSR
5149 027730  023737  002244  002250     CMP     SBEMSK,DBEMSK      ;CAN'T HAVE THE SAME ERROR BIT SET
5150 027736  001004                     BNE     6$                 ;IN BOTH MASKS
5151 027740  023737  002246  002252     CMP     SBEMSK+2,DBEMSK+2  ;COULD BE EQUAL IN SECOND WORD
5152 027746  001460                     BEQ     13$                ;GO MAKE THEM NOT EQUAL
5153 027750  012737  002240  002272 6$:    MOV     #TSTDAT,SOURCE     ;SOURCE ADDRESS FOR CHKGEN
5154 027756  004737  040360             CALL    CHKGEN             ;GO GENERATE CHECK BITS
5155 027762  013702  002244             MOV     SBEMSK,R2
5156 027766  074237  002240             XOR     R2,TSTDAT
5157 027772  013702  002246             MOV     SBEMSK+2,R2
5158 027776  074237  002242             XOR     R2,TSTDAT+2
5159 030002  013702  002250             MOV     DBEMSK,R2
5160 030006  074237  002240             XOR     R2,TSTDAT
5161 030012  013702  002252             MOV     DBEMSK+2,R2
5162 030016  074237  002242             XOR     R2,TSTDAT+2
5163 030022  104471              16$:   ECC1DIS                    ;DISABLE ECC ON 1 SELECTED CSR
5164 030024  013731  002240             MOV     TSTDAT,@(R1)+      ;WRITE 16 BITS
5165 030030  104475                     CB1CSR                     ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
5166 030032  013771  002242  000000     MOV     TSTDAT+2,@(R1)     ;WRITE HIGH WORD
5167 030040  104503                     CLR1CSR                    ;CLEAR 1 SELECTED CSR
5168 030042  162701  000002             SUB     #2,R1              ;ADJUST TEST ADDRESS
5169 030046  005771  000000             TST     @(R1)              ;READ THE LOCATION
5170 030052  104501                     WAS1DBE                    ;WAS THERE ANY DOUBLE BIT ERRORS ON 1 SELECTED CSR
5171 030054  103411                     BCS     9$                 ;IT SHOULD BE SET
5172 030056                             SET     HEADER
5173 030064  011137  002032             MOV     (R1),ADDRESS
5174 030070  104030                     ERROR   +30
5175 030072                             SET     HEADER
```

```
5178 030100  104512                    9$:     ERRGEN
5179 030102  105737  002256                    TSTB    PASFLG
5180 030106  100452                    BMI     14$
5181 030110  005737  002252            13$:    TST     DBEMSK+2        ;CHECK MASK FOR LAST BIT
5182 030114  100405                    BMI     10$             ;MINUS = BIT31
5183 030116                            DLEFT   DBEMSK
5184 030126  000662                    BR      4$
5185 030130                            10$:    IF #SW11 SET.IN @SWR THEN GOTO 11$
5186 030140                            IF QVFLAG IS TRUE THEN GOTO 11$
5187 030146  005737  002246                    TST     SBEMSK+2        ;CHECK SINGLE ERROR MASK TOO
5188 030152  100405                    BMI     11$             ;BR IF DONE
5189 030154                            DLEFT   SBEMSK
5190 030164  000636                    BR      3$
5191 030166  105737  002256            11$:    TSTB    PASFLG  ;FIRST PASS
5192 030172  001003                    BNE     12$             ;NON ZERO MEANS WE'RE DONE
5193 030174  105237  002256                    INCB    PASFLG  ;FIRST PASS, NOT DONE
5194                                   ;CLEAR OUT ANY DBE'S OR SBE'S
5195 030200  000617                    BR      1$              ;KEEP GOING
5196 030202  052737  000200  002256  12$:     BIS     #BIT7,PASFLG    ;SET UP FOR CHECK BIT TEST
5197 030210  005037  002240                    CLR     TSTDAT
5198 030214  005037  002242                    CLR     TSTDAT+2
5199 030220  012737  000040  002244            MOV     #40,SBEMSK
5200 030226  012737  000100  002250            MOV     #100,DBEMSK
5201 030234  012737  003740  002274  14$:     MOV     #3740,CHECK
5202 030242  013702  002244                    MOV     SBEMSK,R2
5203 030246  074237  002274                    XOR     R2,CHECK
5204 030252  013702  002250                    MOV     DBEMSK,R2
5205 030256  074237  002274                    XOR     R2,CHECK
5206 030262  006337  002250                    ASL     DBEMSK
5207 030266  032737  020000  002250            BIT     #BIT13,DBEMSK
5208 030274  001652                    BEQ     16$
5209 030276  006337  002244                    ASL     SBEMSK
5210 030302  032737  004000  002244            BIT     #BIT11,SBEMSK
5211 030310  001006                    BNE     15$
5212 030312  013737  002244  002250            MOV     SBEMSK,DBEMSK
5213 030320  006337  002250                    ASL     DBEMSK
5214 030324  000743                    BR      14$
5215 030326  104471                    15$:    ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
5216 030330  012701  002362                    MOV     #TESTADD,R1
5217 030334                            CLEAR   @(R1)+,@(R1)
5218 030342  104503                    CLR1CSR                 ;CLEAR 1 SELECTED CSR
5219 030344  000207                    RETURN
```

```
5222 030346                                MTP014: SUBTST  <<MTP014        WRITE INHIBIT DURING DATIP WITH DBE TEST>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       MTP014  WRITE INHIBIT DURING DATIP WITH DBE TEST
                                           ;*****************************************************************************
5223                                                      ;THIS TEST CHECKS THE WRITE INHIBIT ON DOUBLE
5224                                                      ;BIT ERRORS DURING A DATIP OPERATION BY USE
5225                                                      ;OF AN 'ASRB' INSTRUCTION.
5226 030346                                       IF KFLAG IS TRUE THEN $RETURN
5227
5233                                                      ;NOTE- THIS TEST WILL ONLY BE RUN FOR MF11S-K MEMORY.
5234 030356  005037  002234            1$:      CLR     DATBUF          ;INITIAL DATA
5235 030362  005037  002236                     CLR     DATBUF+2        ;2 WORDS WORTH
5236 030366  012737  00000'  002244   2$:      MOV     #1,SBEMSK       ;INITIAL ERROR MASK
5237 030374  005037  002246                     CLR     SBEMSK+2        ;
5238 030400  012737  000001  002250   3$:      MOV     #1,DBEMSK       ;DOUBLE ERROR MASK
5239 030406  005037  002252                     CLR     DBEMSK+2        ;2 WORDS
5240 030412  013737  002234  002240   4$:      MOV     DATBUF,TSTDAT   ;PRESERVE ORIG DATA
5241 030420  013737  002236  002242            MOV     DATBUF+2,TSTDAT+2
5242 030426  105737  002256                     TSTB    PASFLG  ;SECOND PASS YET ?
5243 030432  001404                           BEQ     5$              ;BR IF NO
5244 030434  005137  002240                     COM     TSTDAT          ;COMPL DATA  ON SECOND PASS
5245 030440  005137  002242                     COM     TSTDAT+2
5246 030444  104503                     5$:     CLR1CSR                 ;CLEAR 1 SELECTED CSR
5247 030446  023737  002250  002244            CMP     DBEMSK,SBEMSK   ;CHECK FOR SAME MASKS
5248 030454  001004                           BNE     6$              ;BR IF OK
5249 030456  023737  002252  002246            CMP     DBEMSK+2,SBEMSK+2
5250 030464  001476                           BEQ     11$             ;BR IF THEY'RE EQUAL
5251 030466  012737  002240  002272   6$:      MOV     #TSTDAT,SOURCE  ;SET UP ADDRESS FOR CHKGEN
5252 030474  004737  040360                     CALL    CHKGEN          ;GENERATE CHECK BITS
5253 030500  013701  002244                     MOV     SBEMSK,R1
5254 030504  074137  002240                     XOR     R1,TSTDAT
5255 030510  013701  002246                     MOV     SBEMSK+2,R1
5256 030514  074137  002242                     XOR     R1,TSTDAT+2
5257 030520  013701  002250                     MOV     DBEMSK,R1
5258 030524  074137  002240                     XOR     R1,TSTDAT
5259 030530  013701  002252                     MOV     DBEMSK+2,R1
5260 030534  074137  002242                     XOR     R1,TSTDAT+2
5261 030540  012701  002362            7$:      MOV     #TESTADD,R1     ;TEST ADDRESS
5262 030544  104471                           ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
5263 030546  013731  002240                     MOV     TSTDAT,@(R1)+   ;WRITE FIRST 16 BITS
5264 030552  104475                           CB1CSR                  ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
5265 030554  013771  002242  000000            MOV     TSTDAT+2,@(R1)  ;SECOND 16 BITS+CHECKBITS
5266 030562  105037  002257                     CLRB    UPPFLG          ;INDICATE LOWER WORD
5267 030566  013703  002362                     MOV     TESTADD,R3      ;TEST ADDRESS
5268 030572  104503                     8$:     CLR1CSR                 ;CLEAR 1 SELECTED CSR
5269 030574  106223                           ASRB    (R3)+           ;SPECIAL DATIP INSTRUCTION
5270 030576  015100                           MOV     @-(R1),R0
5271 030600  023700  002240                     CMP     TSTDAT,R0       ;CHECK FOR UNCHANGED DATA
5272 030604  001404                           BEQ     9$              ;SHOULD BE UNCHANGED
5273 030606  017137  000000  002032            MOV     @(R1),ADDRESS
5274 030614  104455                           PERR31
5275
5276 030616  062701  000002            9$:      ADD     #2,R1           ;POINT TO UPPER WORD
5277 030622  017100  000000                     MOV     @(R1),R0
5278 030626  023700  002242                     CMP     TSTDAT+2,R0     ;READ IT
5279 030632  001404                           BEQ     10$             ;BR IF UNCHANGED
5280 030634  017137  000000  002032            MOV     @(R1),ADDRESS
```

```
                        PERR31

                10$:    CMPB    #3,UPPFLG               ;LOWER WORD
                        BEQ     11$             ;BR IF NO
                        INCB    UPPFLG
                        BR      8$
                11$:    TSTB    PASFLG
                        BMI     15$             ;BRANCH IF WE'RE TESTING CHECK BITS
                        TST     DBEMSK+2        ;LAST BIT IN MASK ?
                        BMI     12$             ;BR IF BIT 31
                        DLEFT   DBEMSK
                        BR      4$
                12$:    IF #SW11 SET.IN @SWR THEN GOTO 13$
                        IF QVFLAG IS TRUE THEN GOTO 13$
                        TST     SBEMSK+2        ;LAST BIT IN SINGLE ERROR MASK ?
                        BMI     13$             ;BR IF YES
                        DLEFT   SBEMSK
                        BR      3$
                13$:    TSTB    PASFLG  ;WHICH PASS
                        BNE     14$             ;BR IF WE'RE DONE
                        INCB    PASFLG  ;INDICATE SECOND PASS COMING
                        ;CLEAR OUT ANY DBE'S OR SBE'S
                        JMP     1$              ;GO DO IT!
                14$:    BIS     #BIT7,PASFLG
                        CLR     TSTDAT
                        CLR     TSTDAT+2
                        MOV     #40,SBEMSK
                        MOV     #100,DBEMSK
                15$:    MOV     #3740,CHECK
                        MOV     SBEMSK,R2
                        XOR     R2,CHECK
                        MOV     DBEMSK,R2
                        XOR     R2,CHECK
                        ASL     DBEMSK
                        BIT     #BIT13,DBEMSK
                        BEQ     7$
                        ASL     SBEMSK
                        BIT     #BIT11,SBEMSK
                        BNE     16$
                        MOV     SBEMSK,DBEMSK
                        ASL     DBEMSK
                        BR      15$
                16$:    ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
                        MOV     #TESTADD,R1
                        CLEAR   @(R1)+,@(R1)
                        CLR1CSR                 ;CLEAR 1 SELECTED CSR
                        RETURN
```

```
5330 031130                          MTP015: SUBTST  <<MTP015        WRITE INHIBIT OF BYTE WITH DBE>>
                                     ;*********************************************************************
                                     ;*SUBTEST       MTP015  WRITE INHIBIT OF BYTE WITH DBE
                                     ;*********************************************************************
5331                                 ;CHECK FOR WRITE INHIBIT DURING A WRITE BYTE.
5332                                 ;CHECKS FOR UNCORRECTED DATA.
5333 031130  005037  002234          1$:     CLR     DATBUF          ;INITIAL DATA
5334 031134  005037  002236                  CLR     DATBUF+2        ;32 BITS WORTH
5335 031140  012737  000001  002244  2$:     MOV     #1,SBEMSK       ;SINGLE ERROR MASK
5336 031146  005037  002246                  CLR     SBEMSK+2        ;
5337 031152  012737  000001  002250  3$:     MOV     #1,DBEMSK       ;DOUBLE ERROR MASK
5338 031160  005037  002252                  CLR     DBEMSK+2        ;
5339 031164  013737  002234  002240  4$:     MOV     DATBUF,TSTDAT   ;PRESERVE ORIG DATA
5340 031172  013737  002236  002242          MOV     DATBUF+2,TSTDAT+2
5341 031200  105737  002256                  TSTB    PASFLG  ;WHICH PASS ?
5342 031204  001404                          BEQ     5$              ;FIRST PASS, NO COMPLEMENTING
5343 031206  005137  002240                  COM     TSTDAT
5344 031212  005137  002242                  COM     TSTDAT+2        ;SECOND PASS, COMPLEMENT TSTDAT
5345 031216  104503                  5$:     CLR1CSR                 ;CLEAR 1 SELECTED CSR
5346 031220  023737  002244  002250          CMP     SBEMSK,DBEMSK   ;CHECK FOR SAME MASKS
5347 031226  001004                          BNE     6$              ;BR IF NOT EQUAL
5348 031230  023737  002246  002252          CMP     SBEMSK+2,DBEMSK+2       ;SECOND WORD ALSO
5349 031236  001474                          BEQ     11$             ;BR TO MAKE THEM NOT EQUAL
5350 031240  012737  002240  002272  6$:     MOV     #TSTDAT,SOURCE  ;ADDRESS FOR CHKGEN
5351 031246  004737  040360                  CALL    CHKGEN          ;GO GENERATE CHECK BITS
5352 031252  013701  002244                  MOV     SBEMSK,R1
5353 031256  074137  002240                  XOR     R1,TSTDAT
5354 031262  013701  002246                  MOV     SBEMSK+2,R1
5355 031266  074137  002242                  XOR     R1,TSTDAT+2
5356 031272  013701  002250                  MOV     DBEMSK,R1
5357 031276  074137  002240                  XOR     R1,TSTDAT
5358 031302  013701  002252                  MOV     DBEMSK+2,R1
5359 031306  074137  002242                  XOR     R1,TSTDAT+2
5360 031312  012701  002362          7$:     MOV     #TESTADD,R1     ;TEST LOCATION
5361 031316  104471                          ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
5362 031320  013731  002240                  MOV     TSTDAT,@(R1)+   ;WRITE FIRST 16 BITS
5363                                  ;LOAD CSR WITH IMAGE FROM R2
5364 031324  104475                          CB1CSR                  ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
5365 031326  013771  002242  000000          MOV     TSTDAT+2,@(R1)  ;WRITE SECOND 16 BITS + CHECKBITS
5366 031334  104503                          CLR1CSR                 ;CLEAR 1 SELECTED CSR
5367 031336  013702  002362                  MOV     TESTADD,R2      ;GET ADDRESS OF TEST LOC
5368 031342  010203                          MOV     R2,R3           ;R2 DESIGNATES FIRST BYTE
5369 031344  062703  000003                  ADD     #3,R3           ;R3 DESIGNATES LAST BYTE
5370 031350  112722  000360          8$:     MOVB    #360,(R2)+      ;TRY WRITING A BYTE
5371 031354  012701  002362                  MOV     #TESTADD,R1
5372 031360  017100  000000                  MOV     @(R1),R0
5373 031364  023700  002240                  CMP     TSTDAT,R0       ;CHECK FOR UNCHANGED DATA
5374 031370  001404                          BEQ     9$              ;BR IF OK
5375 031372  017137  000000  002032          MOV     @(R1),ADDRESS
5376 031400  104455                          PERR31
5377
5378 031402  017100  000002          9$.     MOV     @2(R1),R0
5379 031406  023700  002242                  CMP     TSTDAT+2,R0     ;READ SECOND WORD
5380 031412  001404                          BEQ     10$             ;BR IF UNCHANGED
5381 031414  017137  000002  002032          MOV     @2(R1),ADDRESS
5382 031422  104455                          PERR31
5383
```

```
5384 031424  020203                    10$:   CMP     R2,R3            ;TESTED LAST BYTE ?
5385 031426  001350                           BNE     8$               ;BR IF NO
5386 031430  105737  002256           11$:   TSTB    PASFLG
5387 031434  100452                           BMI     15$              ;BRANCH IF TESTING CHECK BITS
5388 031436  005737  002252                   TST     DBEMSK+2         ;CHECKING FOR LAST ERROR BIT
5389 031442  100405                           BMI     12$              ;BR IF DONE HERE
5390 031444                                    DLEFT   DBEMSK
5391 031454  000643                           BR      4$
5392 031456                           12$:   IF #SW11 SET.IN @SWR THEN GOTO 13$
5393 031466                                  IF QVFLAG IS TRUE THEN GOTO 13$
5394 031474  005737  002246                   TST     SBEMSK+2         ;LAST SBE MASK
5395 031500  100405                           BMI     13$              ;BR IF DONE WITH THIS PASS
5396 031502                                    DLEFT   SBEMSK
5397 031512  000617                           BR      3$
5398 031514  105737  002256           13$:   TSTB    PASFLG  ;TEST PASS FLAG
5399 031520  001003                           BNE     14$              ;NON ZERO MEANS WE'RE DONE
5400 031522  105237  002256                   INCB    PASFLG  ;NOT DONR
5401 031526  000600                           BR      1$
5402 031530  052737  000200  002256  14$:    BIS     #BIT7,PASFLG
5403 031536  005037  002240                   CLR     TSTDAT
5404 031542  005037  002242                   CLR     TSTDAT+2
5405 031546  012737  000040  002244           MOV     #40,SBEMSK
5406 031554  012737  000100  002250           MOV     #100,DBEMSK
5407 031562  012737  003740  002274  15$:    MOV     #3740,CHECK
5408 031570  013702  002244                   MOV     SBEMSK,R2
5409 031574  074237  002274                   XOR     R2,CHECK
5410 031600  013702  002250                   MOV     DBEMSK,R2
5411 031604  074237  002274                   XOR     R2,CHECK
5412 031610  006337  002250                   ASL     DBEMSK
5413 031614  032737  020000  002250           BIT     #BIT13,DBEMSK
5414 031622  001633                           BEQ     7$
5415 031624  006337  002244                   ASL     SBEMSK
5416 031630  032737  004000  002244           BIT     #BIT11,SBEMSK
5417 031636  001006                           BNE     16$
5418 031640  013737  002244  002250           MOV     SBEMSK,DBEMSK
5419 031646  006337  002250                   ASL     DBEMSK
5420 031652  000743                           BR      15$
5421 031654  104471                  16$:   ECC1DIS                  ;DISABLE ECC ON 1 SELECTED CSR
5422 031656  012701  002362                   MOV     #TESTADD,R1      ;TEST LOCATION
5423 031662                                    CLEAR   @(R1)+,@(R1)     ;TO ERASE ANY DBE'S FROM TESTING
5424                                           ;RESTORE CSR
5425 031670  104503                           CLR1CSR                  ;CLEAR 1 SELECTED CSR
5426 031672  000207                           RETURN
```

```
5429 031674                                MTP016: SUBTST  <<MTP016        WRITE INHIBIT OF WORD WITH DBE>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       MTP016  WRITE INHIBIT OF WORD WITH DBE
                                           ;*****************************************************************************
                                           ;
5430                                               ;DOUBLE BIT ERROR WRITE CANCEL WITH
5431                                               ;WORD WRITE.
5432                                               ;CHECKS WRITE INHIBIT WITH WORD WRITES TO
5433                                               ;WORD WITH DOUBLE ERROR.
5434 031674  005037  002234          T12A:   CLR     DATBUF          ;BACKGROUND FOR DOUBLE ERRORS
5435 031700  005037  002236                  CLR     DATBUF+2        ;2 WORDS WORTH
5436 031704  012737  000001  002244        MOV     #1,SBEMSK       ;SINGLE ERROR MASK
5437 031712  005037  002246                  CLR     SBEMSK+2        ;
5438 031716  012737  000001  002250  T12B:   MOV     #1,DBEMSK       ;DOUBLE ERROR MASK
5439 031724  005037  002252                  CLR     DBEMSK+2
5440 031730  013737  002234  002240  1$:     MOV     DATBUF,TSTDAT   ;DATA FOR TEST
5441 031736  013737  002236  002242        MOV     DATBUF+2,TSTDAT+2;BOTH WORDS
5442 031744  105737  002256                  TSTB    PASFLG  ;COMP DATA ON SECOND PASS ONLY
5443 031750  001404                        BEQ     2$              ;BR IF FIRST PASS
5444 031752  005137  002240                  COM     TSTDAT          ;COMP FIRST WORD
5445 031756  005137  002242                  COM     TSTDAT+2        ;NOW SECOND WORD
5446 031762  023737  002244  002250  2$:     CMP     SBEMSK,DBEMSK   ;CHECK FOR IDENTICAL MASKS
5447 031770  001004                        BNE     3$              ;BR IF DIFFERENT
5448 031772  023737  002246  002252        CMP     SBEMSK+2,DBEMSK+2;UPPER WORD TOO
5449 032000  001502                        BEQ     8$              ;BR TO MAKE THEM NOT EQUAL
5450 032002  012737  002240  002272  3$:     MOV     #TSTDAT,SOURCE  ;NEED ADDR OF DATA FOR CHKGEN
5451 032010  004737  040360                  CALL    CHKGEN          ;GO GENERATE CHECK BITS
5452 032014  013701  002244                  MOV     SBEMSK,R1
5453 032020  074137  002240                  XOR     R1,TSTDAT
5454 032024  013701  002246                  MOV     SBEMSK+2,R1
5455 032030  074137  002242                  XOR     R1,TSTDAT+2
5456 032034  013701  002250                  MOV     DBEMSK,R1
5457 032040  074137  002240                  XOR     R1,TSTDAT
5458 032044  013701  002252                  MOV     DBEMSK+2,R1
5459 032050  074137  002242                  XOR     R1,TSTDAT+2
5460 032054  012701  002362          4$:     MOV     #TESTADD,R1     ;FIRST TEST ADRRESS
5461 032060  104471                        ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
5462 032062  013731  002240                  MOV     TSTDAT,@(R1)+   ;WRITE FIRST 16 BITS
5463 032066  104475                        CB1CSR                  ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
5464 032070  013771  002242  000000        MOV     TSTDAT+2,@(R1)  ;WRITE SECOND 16 BITS + CHECKBITS
5465 032076  105037  002257                  CLRB    UPPFLG          ;SET FOR 2 LOOPS
5466 032102  162701  000002                  SUB     #2,R1           ;POINT TO LOW WORD
5467 032106  104503                  5$:     CLR1CSR                 ;CLEAR 1 SELECTED CSR
5468 032110  012771  177400  000000        MOV     #177400,@(R1)   ;TRY WRITING LOCATION
5469 032116  012701  002362                  MOV     #TESTADD,R1
5470 032122  017100  000000                  MOV     @(R1),R0
5471 032126  023700  002240                  CMP     TSTDAT,R0       ;CHECK FOR ORIGINAL DATA
5472 032132  001404                        BEQ     6$              ;SHOULD BE UNCHANGED
5473 032134  017137  000000  002032        MOV     @(R1),ADDRESS
5474 032142  104455                        PERR31
5475
5476 032144  062701  000002          6$:     ADD     #2,R1
5477 032150  017100  000000                  MOV     @(R1),R0
5478 032154  023700  002242                  CMP     TSTDAT+2,R0     ;THIS SHOULD BE UNCHANGED ALSO
5479 032160  001404                        BEQ     7$
5480 032162  017137  000000  002032        MOV     @(R1),ADDRESS
5481 032170  104455                        PERR31
```

```
5484 032172  105737  002257          7$:     TSTB    UPPFLG          ;WHICH LOOP ?
5485 032176  001003                          BNE     8$              ;SECOND, BR OUT
5486 032200  105237  002257                  INCB    UPPFLG          ;FIRST, KEEP GOING
5487 032204  000740                          BR      5$
5488 032206  105737  002256          8$:     TSTB    PASFLG
5489 032212  100454                          BMI     12$
5490 032214  005737  002252                  TST     DBEMSK+2        ;LAST BIT ?
5491 032220  100405                          BMI     9$              ;MINUS = BIT 31
5492 032222                                  DLEFT   DBEMSK
5493 032232  000636                          BR      1$
5494 032234                          9$:     IF #SW11 SET.IN @SWR THEN GOTO 10$
5495 032244                                  IF QVFLAG IS TRUE THEN GOTO 10$
5496 032252  005737  002246                  TST     SBEMSK+2        ;LAST BIT IN THIS MASK ?
5497 032256  100406                          BMI     10$             ;BR IF LAST BIT
5498 032260                                  DLEFT   SBEMSK
5499 032270  000137  031716                  JMP     T12B
5500 032274  105737  002256          10$:    TSTB    PASFLG  ;FIRST PASS ?
5501 032300  001004                          BNE     11$             ;BR IF SECOND
5502 032302  105237  002256                  INCB    PASFLG  ;INDICATE SECOND PASS COMING
5503 032306  000137  031674                  JMP     T12A
5504 032312  052737  000200  002256  11$:    BIS     #BIT7,PASFLG
5505 032320  005037  002240                  CLR     TSTDAT
5506 032324  005037  002242                  CLR     TSTDAT+2
5507 032330  012737  000040  002244          MOV     #40,SBEMSK
5508 032336  012737  000100  002250          MOV     #100,DBEMSK
5509 032344  012737  003740  002274  12$:    MOV     #3740,CHECK
5510 032352  013702  002244                  MOV     SBEMSK,R2
5511 032356  074237  002274                  XOR     R2,CHECK
5512 032362  013702  002250                  MOV     DBEMSK,R2
5513 032366  074237  002274                  XOR     R2,CHECK
5514 032372  006337  002250                  ASL     DBEMSK
5515 032376  032737  020000  002250          BIT     #BIT13,DBEMSK
5516 032404  001623                          BEQ     4$
5517 032406  006337  002244                  ASL     SBEMSK
5518 032412  032737  004000  002244          BIT     #BIT11,SBEMSK
5519 032420  001006                          BNE     13$
5520 032422  013737  002244  002250          MOV     SBEMSK,DBEMSK
5521 032430  006337  002250                  ASL     DBEMSK
5522 032434  000743                          BR      12$
5523 032436  104471                  13$:    ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
5524 032440  012701  002362                  MOV     #TESTADD,R1     ;RESTORE TEST ADDRESS
5525 032444  005031                          CLR     @(R1)+          ;CLEAR ANY DBE'S FROM TEST
5526 032446  005071  000000                  CLR     @(R1)
5527 032452  104503                          CLR1CSR                 ;CLEAR 1 SELECTED MK11 CSR
5528 032454  000207                          RETURN
```

```
5531 032456                          MTP017: SUBTST  <<MTP017        HOLDING 1'S & 0'S TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MTP017  HOLDING 1'S & 0'S TEST
                                     ;*****************************************************************************
5532                                         ;*(1)    THIS TEST CHECKS THE MEMORY FOR THE CAPABILITY
5533                                         ;*       OF HOLDING 1'S AND 0'S BY WRITING A BACKGROUND
5534                                         ;*       OF 000377 AND READING IT
5535                                         ;*(2)    MEMORY IS WRITTEN USING A BYTE AT A TIME
5536                                         ;*(3)    STEPS 1 & 2 ARE REPEATED WITH A SWAPPED BACKGROUND PATTERN
5537                                         ;NOTE:   THIS TEST WRITES BYTES & READS WORDS
5538 032456  012701  060000                 MOV      #FIRST,R1
5539 032462  010104                         MOV      R1,R4
5540 032464  012705  160000                 MOV      #LAST+2,R5
5541 032470  012700  000377                 MOV      #377,R0          ;GET THE PATTERN INTO R0
5542 032474  010003                         MOV      R0,R3
5543 032476  000303                         SWAB     R3
5544 032500  110021                  1$:    MOVB     R0,(R1)+         ;WRITE A BYTE
5545 032502  110321                         MOVB     R3,(R1)+         ;WRITE THE MEMORY WITH THE BYTE STORED IN BAKPAT+1
5546 032504  020105                         CMP      R1,R5            ;COMPARE TEST LOC TO TOP + 2
5547 032506  103774                         BLO      1$               ;BRANCH IF LOWER
5548
5549 032510  014102                  2$:    MOV      -(R1),R2
5550 032512  020002                         CMP      R0,R2            ;TEST THE MEMORY TO SEE IF IT CONTAINS
5551                                                                   ;THE WORD STORED IN BAKPAT
5552 032514  001401                         BEQ      3$
5553 032516  104446                         PERR22
5554
5555 032520  020104                  3$:    CMP      R1,R4            ;KEEP ON TESTING THE MEMORY UNTIL
5556 032522  101372                         BHI      2$               ;R1 EQUALS THE LOWEST ADDRESS
5557 032524  000303                         SWAB     R3               ;CHANGE THE DATA PATTERN
5558 032526  000300                         SWAB     R0
5559 032530  001763                         BEQ      1$               ;IF THE DATA PATTERN DOES NOT HAVE LOW
5560                                                                   ; BYTE =0 THEN FALL THRU
5561 032532  000207                  RETURN
```

```
5564 032534                    MTP020: SUBTST  <<MTP020        MARCHING 1'S & 0'S IN CHECK BITS TEST>>
                               ;*********************************************************************************
                               ;*SUBTEST       MTP020  MARCHING 1'S & 0'S IN CHECK BITS TEST
                               ;*********************************************************************************
5565                                   ;*THIS TEST IS CONCERNED ONLY WITH THE INTEGRITY
5566                                   ;*OF THE MOS RAMS THAT STORE THE CHECKBITS.
5567
5568                                   ;077 --> 100 DOWN
5569 032534  160201              MTPA20: SUB     R2,R1           ;V177640
5570 032536  160204                      SUB     R2,R4           ;V177642
5571 032540  005711                      TST     (R1)            ;V177644        ;1ST WORD OK?
5572 032542  001002                      BNE     1$              ;V177646        ;NO - SKIP
5573 032544  005714                      TST     (R4)            ;V177650        ;2ND WORD OK?
5574 032546  001401                      BEQ     2$              ;V177652        ;YES - SKIP
5575 032550  104453              1$:     PERR27                  ;V177654        ;GOOD=000000,,000000,,077
5576 032552  010314              2$:     MOV     R3,(R4)         ;V177656        ;2ND WORD <= 100000
5577 032554  005011                      CLR     (R1)            ;V177660        ;CLEAR 1ST WORD
5578 032556  020100                      CMP     R1,R0           ;V177662        ;ARE WE DONE?
5579 032560  002365                      BGE     MTPA20          ;V177664        ;BRANCH IF NOT
5580 032562  000207                      RETURN                  ;V177666
5581
5582                                   ;100 --> 077 UP
5583 032564  005711              MTPB20: TST     (R1)            ;V177640        ;1ST WORD OK?
5584 032566  001002                      BNE     3$              ;V177642        ;NO - SKIP
5585 032570  020314                      CMP     R3,(R4)         ;V177644        ;2ND WORD OK?
5586 032572  001401                      BEQ     4$              ;V177646        ;YES - SKIP
5587 032574  104452              3$:     PERR26                  ;V177650        ;GOOD=000000,,100000,,100
5588 032576  005014              4$:     CLR     (R4)            ;V177652        ;CLEAR 2ND WORD
5589 032600  005011                      CLR     (R1)            ;V177654        ;CLEAR 1ST WORD
5590 032602  060201                      ADD     R2,R1           ;V177656
5591 032604  060204                      ADD     R2,R4           ;V177660
5592 032606  020405                      CMP     R4,R5           ;V177662        ;TOP + 2 YET?
5593 032610  001365                      BNE     MTPB20          ;V177664        ;NO - LOOP
5594 032612  000207                      RETURN                  ;V177666
5595
5596                                   ;077 --> 100 UP
5597 032614  005711              MTPC20: TST     (R1)            ;V177640        ;1ST WORD OK?
5598 032616  001002                      BNE     5$              ;V177642        ;NO - SKIP
5599 032620  005714                      TST     (R4)            ;V177644        ;2ND WORD OK?
5600 032622  001401                      BEQ     6$              ;V177646        ;YES - SKIP
5601 032624  104453              5$:     PERR27                  ;V177650        ;GOOD=000000,,000000,,077
5602 032626  010314              6$:     MOV     R3,(R4)         ;V177652        ;WRITE 1ST WORD
5603 032630  005011                      CLR     (R1)            ;V177654        ;WRITE 2ND WORD
5604 032632  060204                      ADD     R2,R4           ;V177656
5605 032634  060201                      ADD     R2,R1           ;V177660
5606 032636  020405                      CMP     R4,R5           ;V177662        ;TOP + 2 YET?
5607 032640  001365                      BNE     MTPC20          ;V177664        ;NO - LOOP
5608 032642  000207                      RETURN                  ;V177666
```

```
5611                                    ;100 --> 077 DOWN
5612 032644  160201            MTPD20:  SUB     R2,R1          ;V177640
5613 032646  160204                     SUB     R2,R4          ;V177642
5614 032650  020314                     CMP     R3,(R4)        ;V177644        ;2ND WORD OK?
5615 032652  001002                     BNE     7$             ;V177646        ;NO - SKIP
5616 032654  005711                     TST     (R1)           ;V177650        ;1ST WORD OK?
5617 032656  001401                     BEQ     8$             ;V177652        ;YES - SKIP
5618 032660  104452            7$:      PERR26                 ;V177654        ;GOOD=000000,,100000,,100
5619 032662  005014            8$:      CLR     (R4)           ;V177656        ;WRITE 1ST WORD
5620 032664  005011                     CLR     (R1)           ;V177660        ;WRITE 2ND WORD
5621 032666  020100                     CMP     R1,R0          ;V177662
5622 032670  002365                     BGE     MTPD20         ;V177664
5623 032672  000207                     RETURN                 ;V177666
5624
5625                                    ;077 UP
5626 032674  005711            MTPE20:  TST     (R1)           ;V177640        ;1ST WORD OK?
5627 032676  001002                     BNE     9$             ;V177642        ;NO - SKIP
5628 032700  005714                     TST     (R4)           ;V177644        ;2ND WORD OK?
5629 032702  001401                     BEQ     10$            ;V177646        ;YES - SKIP
5630 032704  104453            9$:      PERR27                 ;V177650        ;GOOD=000000,,000000,,077
5631 032706  060201            10$:     ADD     R2,R1          ;V177652
5632 032710  060204                     ADD     R2,R4          ;V177654
5633 032712  020405                     CMP     R4,R5          ;V177656        ;TOP + 2 YET?
5634 032714  001367                     BNE     MTPE20         ;V177660        ;NO - LOOP
5635 032716  000207                     RETURN                 ;V177662
```

```
5638 032720                    MTPA21: SUBTST  <<MTPA21        MARCHING 1'S & 0'S PATTERN TEST>>
                               ;*****************************************************************************
                               ;*SUBTEST       MTPA21  MARCHING 1'S & 0'S PATTERN TEST
                               ;*****************************************************************************
5639                                   ;READ,BYTESWAP-MODIFY,READ,DOWN
5640 032720  014100           1$:      MOV     -(R1),R0;V177640
5641 032722  020200                    CMP     R2,R0   ;V177642
5642 032724  001401                    BEQ     2$      ;V177644
5643 032726  104443                    PERR17          ;V177646
5644
5645 032730  000311           2$:      SWAB    (R1)    ;V177650
5646 032732  011100                    MOV     (R1),R0 ;V177652
5647 032734  020300                    CMP     R3,R0   ;V177654
5648 032736  001401                    BEQ     3$      ;V177656
5649 032740  104444                    PERR20          ;V177660
5650
5651 032742  020401           3$:      CMP     R4,R1   ;V177662        ;DONE?
5652 032744  001365                    BNE     1$      ;V177664        ;NO - LOOP
5653 032746  000207                    RETURN          ;V177666        ;YES - RETURN
5654
5655 032750                    MTPB21: ;READ,BYTESWAP-MODIFY,READ,UP
5656 032750  011100           1$:      MOV     (R1),R0 ;V177640
5657 032752  020300                    CMP     R3,R0   ;V177642
5658 032754  001401                    BEQ     2$      ;V177644
5659 032756  104444                    PERR20          ;V177646
5660
5661 032760  000311           2$:      SWAB    (R1)    ;V177650
5662 032762  011100                    MOV     (R1),R0 ;V177652
5663 032764  020200                    CMP     R2,R0   ;V177654
5664 032766  001401                    BEQ     3$      ;V177656
5665 032770  104443                    PERR17          ;V177660
5666
5667 032772  062701  000002     3$:      ADD     #2,R1   ;V177662
5668 032776  020501                    CMP     R5,R1   ;V177666        ;DONE?
5669 033000  001363                    BNE     1$      ;V177670        ;NO - LOOP
5670 033002  000207                    RETURN          ;V177672        ;YES - RETURN
5671
5672 033004                    MTPC21: ;READ,BYTESWAP-MODIFY,READ,UP
5673 033004  011100           1$:      MOV     (R1),R0 ;V177640
5674 033006  020200                    CMP     R2,R0   ;V177642
5675 033010  001401                    BEQ     2$      ;V177644
5676 033012  104443                    PERR17          ;V177646
5677
5678 033014  000311           2$:      SWAB    (R1)    ;V177650
5679 033016  011100                    MOV     (R1),R0 ;V177652
5680 033020  020300                    CMP     R3,R0   ;V177654
5681 033022  001401                    BEQ     3$      ;V177656
5682 033024  104444                    PERR20          ;V177660
5683
5684 033026  062701  000002     3$:      ADD     #2,R1   ;V177662
5685 033032  020501                    CMP     R5,R1   ;V177666        ;DONE?
5686 033034  001363                    BNE     1$      ;V177670        ;NO - LOOP
5687 033036  000207                    RETURN          ;V177672        ;YES - RETURN
```

```
 5690 033040                        MTPD21: ;READ,BYTESWAP-MODIFY,READ,DOWN
 5691 033040  014100                1$:     MOV     -(R1),R0;V177640
 5692 033042  020300                        CMP     R3,R0   ;V177642
 5693 033044  001401                        BEQ     2$      ;V177644
 5694 033046  104444                        PERR20          ;V177646
 5695
 5696 033050  000311                2$:     SWAB    (R1)    ;V177650
 5697 033052  011100                        MOV     (R1),R0 ;V177652
 5698 033054  020200                        CMP     R2,R0   ;V177654
 5699 033056  001401                        BEQ     3$      ;V177656
 5700 033060  104443                        PERR17          ;V177660
 5701
 5702 033062  020401                3$:     CMP     R4,R1   ;V177662        ;DONE?
 5703 033064  001365                        BNE     1$      ;V177664        ;NO - LOOP
 5704 033066  000207                        RETURN          ;V177666        ;YES - RETURN
 5705
```

```
 5708 033070                            MTP022: SUBTST  <<MTP022        REFRESH & SHIFTING DIAGONAL TEST>>
                                        ;*********************************************************************************
                                        ;*SUBTEST       MTP022  REFRESH & SHIFTING DIAGONAL TEST
                                        ;*********************************************************************************
 5709                                           ;(1)    WE WRITE A DIAGONAL PATTERN IN MEMORY (WITH CACHE ON).
 5710                                           ;(2)    IF A REFRESH TEST WE DISTURB ALL ROWS FOR > 2 MS (WITH CACHE ON).
 5711                                           ;(3)    WE READ & CHECK FOR CORRECTNESS THE DIAGONAL PATTERN
 5712                                           ;       (WITH CACHE OFF).
 5713          000010                   KDIAG=8.                ;HOW OFTEN A DIAGONAL STRIPE OCCURS (MUST BE A POWER OF 2)
 5714 033070                                    FOR EVEN := #1 TO #2                    ;FOR DATA & COMPLEMENT DATA
 5715 033076                                      IF EVEN EQ #1
 5716 033106                                        LET R2 := ZEROS
 5717 033112                                        LET R3 := ONES
 5718 033116                                      ELSE
 5719 033120                                        LET R2 := ONES
 5720 033124                                        LET R3 := ZEROS
 5721 033130                                      END ;OF IF EVEN
 5722 033130                                      FOR STRIPES := #0 TO #KDIAG-1         ;FOR THE NUMBER OF STRIPES
 5723
 5724
 5725 033134  104423                              ;WRITE LOOP
 5726 033136                                      CACHON                          ;TURN CACHE ON
 5727 033144                                      LET COUNT := STRIPES
 5728 033150                                      LET R1 := #FIRST
 5729 033156                                      WHILE R1 LOS #LAST
 5730 033172                                        IF COUNT LT #0 THEN LET COUNT := #KDIAG-1
 5731 033204                                        IF #374 OFF.IN R1 THEN LET COUNT := COUNT - #1
 5732 033212                                        IF COUNT NE #0
 5733 033214                                          LET (R1) := R2
 5734 033220                                          LET 2(R1) := R2
 5735 033222                                        ELSE
 5736 033224                                          LET (R1) := R3
 5737 033230                                          LET 2(R1) := R3
 5738 033230                                        END ;OF IF COUNT
 5739 033234                                        LET COUNT := COUNT - #1
 5740 033240                                        LET R1 := R1 + #4
 5741                                             END ;OF WHILE
 5742                                             ;END OF WRITE LOOP
 5743 033242                                      IF DIAGFLAG IS FALSE THEN $CALL REFRESH
 5744                                             ;READ LOOP
 5745 033254                                      LET COUNT := STRIPES
 5746 033262                                      LET R1 := #FIRST
 5747 033266  104424                              CACHOFF                         ;TURN CACHE OFF
```

```
5749 033270                                     WHILE R1 LOS #LAST
5750 033276                                       IF COUNT LT #0 THEN LET COUNT := #KDIAG-1
5751 033312                                       IF #374 OFF.IN R1 THEN LET COUNT := COUNT - #1
5752 033324                                       IF COUNT NE #0
5753 033332                                         LET R0 := (R1)
5754 033334                                         IF R2 NE R0
5755 033340  104443                                   PERR17
5756 033342                                         END ;OF IF R2
5757 033342                                         LET R0 := 2(R1)
5758 033346                                         IF R2 NE R0
5759 033352  104443                                   PERR17
5760 033354                                         END ;OF IF R2
5761 033354                                       ELSE
5762 033356                                         LET R0 := (R1)
5763 033360                                         IF R3 NE R0
5764 033364  104444                                   PERR20
5765 033366                                         END ;OF IF R3
5766 033366                                         LET R0 := 2(R1)
5767 033372                                         IF R3 NE R0
5768 033376  104444                                   PERR20
5769 033400                                         END ;OF IF R3
5770 033400                                       END ;OF IF COUNT
5771 033400                                       LET COUNT := COUNT - #1
5772 033404                                       LET R1 := R1 + #4
5773 033410                                     END ;OF WHILE
5774                                             ;END OF READ LOOP
5775
5776 033412                                   END ;OF FOR STRIPES
5777 033426                                 END ;OF FOR EVEN
5778 033442  000207                         RETURN
5779
5780 033444                                 REFRESH:SUBTST  <<SUBR  REFRESH DELAY>>
                                            ;*****************************************************************************
                                            ;*SUBTEST        SUBR    REFRESH DELAY
                                            ;*****************************************************************************
5781                                                ;DISTURB EACH ROW FOR > 3.2 MS
5782 033444                                         FOR R0 := #FIRST TO #FIRST+374 BY #4
5783 033450  004737  033514                           CALL REFSUB
5784 033454                                         END ;OF FOR R0
5785 033466                                         LET R0 := #FIRST+BIT14
5786 033472                                         WHILE R0 LOS #LAST+BIT14+374
5787 033500  004737  033514                           CALL REFSUB
5788 033504                                           LET R0 := R0 + #4
5789 033510                                         END ;OF WHILE
5790 033512  000207                                 RETURN
5791 033514  012704  000640            REFSUB:  MOV     #640,R4                ;TIME FOR A > 3.2 MS LOOP
5792 033520  062700  000002                     ADD     #2,R0
5793 033524  005140                    1$:      COM     -(R0)
5794 033526  005120                             COM     (R0)+
5795 033530  005110                             COM     (R0)
5796 033532  005110                             COM     (R0)
5797 033534  077405                             SOB     R4,1$
5798 033536  162700  000002                     SUB     #2,R0
5799 033542  000207                             RETURN
```

```
 5803 033544                    MTPA24: SUBTST  <<MTPA24        FAST GALLOPING PATTERN TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST       MTPA24  FAST GALLOPING PATTERN TEST
                                ;*****************************************************************************
 5804                                   ;THE TOTAL TEST (INCLUDING SETUP) IS AS FOLLOWS
 5805                                   ;*(1)    THIS TEST WRITES THE MEMORY WITH A BACK GROUND PATTERN
 5806                                   ;*       STORED AT LOCATION BAKPAT
 5807                                   ;*(2)    TEST BEGINS AT LOWEST LOCATION BEING TESTED
 5808                                   ;*       (LETS NAME IT 'A')
 5809                                   ;*(3)    LETS NAME THE 1ST LOCATION IN THE ROW/COLUMN UNDER TEST AS 'B'.
 5810                                   ;*(4)    SWAPS BYTES FOR LOCATION 'A'.
 5811                                   ;*(5)    READS 'A', READS 'B'
 5812                                   ;*(6)    'B' = 'B'+400   (ADDS 64 DOUBLE WORDS TO 'B')
 5813                                   ;*(7)    REPEATS STEPS 5 AND 6 UNTIL 'B' IS GREATER THAN THE
 5814                                   ;*(8)    END OF THE BANK  A+2
 5815                                   ;*(9)    REPEATS  STEPS 3-8 UNTILL 'A' REACHES THE END OF THE BANK
 5816                                   ;*(10)   AFTER EXECUTING THE TEST  DATA IS COMPLEMENTED
 5817                                   ;*       AND STEPS 1-9 ARE REPEATED
 5818                                   ;REGISTERS ARE USED AS FOLLOWS
 5819                                   ;R0      TEST DATA
 5820                                   ;R1      'A'
 5821                                   ;R2      'B'
 5822                                   ;R3      BAKPAT
 5823                                   ;R4      SWAPAT
 5824                                   ;R5      LAST
 5825
 5826                                   ;NOTE THE PATTERN STARTS AT MTPB24!..!.!!!!!!!!!!
 5827
 5828                                   ;UIPAR'S
 5829 033544  011100            1$:     MOV     (R1),R0         ;V177640        ;READ 'A'
 5830 033546  020004                    CMP     R0,R4           ;V177642        ;CHECK 'A'
 5831 033550  001401                    BEQ     2$              ;V177644        ;BR IF OK
 5832 033552  104447                    PERR23                  ;V177646        ;REPORT ERROR
 5833
 5834 033554  011200            2$:     MOV     (R2),R0         ;V177650        ;READ 'B'
 5835 033556  020003                    CMP     R0,R3           ;V177652        ;CHECK 'B'
 5836 033560  001401                    BEQ     3$              ;V177654        ;BR IF OK
 5837 033562  104450                    PERR24                  ;V177656        ;REPORT ERROR
 5838
 5839 033564  062702  000400    3$:     ADD     #400,R2         ;V177660        ;BUMP 'B'
 5840 033570  020205                    CMP     R2,R5           ;V177664        ;AT END YET?
 5841 033572  101764                    BLOS    1$              ;V177666        ;BR IF NO
 5842
 5843 033574  062701  000002            ADD     #2,R1           ;V177670        ;BUMP 'A'
 5844 033600  000137  033604            JMP     @#MTPB24        ;V177674        ;GOTO V177260
```

```
5847 033604                          MTPB24: SUBTST  <<MTPB24        FAST GALLOP PART B>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MTPB24  FAST GALLOP PART B
                                     ;*****************************************************************************
5848                                         ;SDPAR'S
5849 033604  010411                          MOV     R4,(R1)         ;V172260        ;WRITE 'A'
5850 033606  020105                          CMP     R1,R5           ;V172262        ;DONE?
5851 033610  001001                          BNE     1$              ;V172264        ;BR IF NO
5852 033612  000207                          RETURN                  ;V172266        ;YES - RETURN
5853 033614  000137  033620        1$:       JMP     @#MTPC24        ;V172270        ;GOTO V172360
5854
5855 033620                          MTPC24: SUBTST  <<MTPC24        FAST GALLOP PART C>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MTPC24  FAST GALLOP PART C
                                     ;*****************************************************************************
5856                                         ;KDPAR'S
5857 033620  010102                          MOV     R1,R2           ;V172360        ;RESET 'B' <--- 'A'
5858 033622  011100                          MOV     (R1),R0         ;V172362        ;READ 'A'
5859 033624  020004                          CMP     R0,R4           ;V172364        ;CHECK 'A'
5860 033626  001401                          BEQ     1$              ;V172366        ;BR IF OK
5861 033630  104447                          PERR23                  ;V172370        ;REPORT ERROR
5862 033632  000137  033564        1$:       JMP     @#MTPA24+20     ;V172372        ;GOTO V177660
```

```
 5865 033636                            MTP025: SUBTST  <<MTP025        INTERRUPT ENABLE TEST>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       MTP025  INTERRUPT ENABLE TEST
                                        ;*****************************************************************************
 5866 033636  005037  002240                    CLR     TSTDAT          ;GENERATE CHECKBITS ON 0,,0
 5867 033642  005037  002242                    CLR     TSTDAT+2
 5868 033646  012737  002240  002272            MOV     #TSTDAT,SOURCE
 5869 033654  004737  040360                    CALL    CHKGEN
 5870 033660  012737  000003  002074            MOV     #3,NOPAR                ;SETUP PARITY ACTION
 5871 033666  012701  002362                    MOV     #TESTADD,R1     ;FIRST TEST ADDRESS
 5872 033672  012737  033732  002264            MOV     #1$,PARTHERE    ;SETUP TRAP DESTINATION
 5873 033700  004737  034154                    CALL    MTPA25          ;WRITE DATA & CHECKBITS
 5874 033704  104473                            ECC1INIT                ;INITIALIZE 1 SELECTED MK11 CSR
 5875 033706  005771  000000                    TST     @(R1)           ;ACCESS LOCATIONS FOR DBE TRAPS
 5876 033712  005771  000002                    TST     @2(R1)
 5877                                           ;NONE - GOOD - ACCESS FOR SBE TRAPS
 5878 033716  104507                            ENA1SBE                 ;DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
 5879 033720  005771  000000                    TST     @(R1)
 5880 033724  005771  000002                    TST     @2(R1)
 5881 033730  000404                            BR      2$              ;NONE - GOOD - SKIP
 5882 033732  104426                    1$:     READCSR
 5883 033734                                    FATAL   27
 5884 033742  005237  002240            2$:     INC     TSTDAT          ;CHECK FOR CORRECT ACTION ON SBE'S
 5885 033746  004737  034102                    CALL    MTPD25          ;IN ALL 4 BYTES
 5886 033752  012737  000400  002240            MOV     #400,TSTDAT
 5887 033760  004737  034102                    CALL    MTPD25
 5888 033764  005037  002240                    CLR     TSTDAT
 5889 033770  005237  002242                    INC     TSTDAT+2
 5890 033774  004737  034102                    CALL    MTPD25
 5891 034000  012737  000400  002242            MOV     #400,TSTDAT+2
 5892 034006  004737  034102                    CALL    MTPD25
 5893
 5894 034012  005037  002242                    CLR     TSTDAT+2        ;CHECK FOR CORRECT ACTION ON DBE'S
 5895 034016  012737  000003  002240            MOV     #3,TSTDAT       ;IN ALL 4 BYTES
 5896 034024  004737  034124                    CALL    MTPE25
 5897 034030  012737  001400  002240            MOV     #1400,TSTDAT
 5898 034036  004737  034124                    CALL    MTPE25
 5899 034042  005037  002240                    CLR     TSTDAT
 5900 034046  012737  000003  002242            MOV     #3,TSTDAT+2
 5901 034054  004737  034124                    CALL    MTPE25
 5902 034060  012737  001400  002242            MOV     #1400,TSTDAT+2
 5903 034066  004737  034124                    CALL    MTPE25
 5904 034072  104503                            C R1CSR                 ;CLEAR 1 SELECTED MK11 CSR
 5905 034074  005037  002074                    CLR     NOPAR           ;INDICATE PARITY ACTION
 5906 034100  000207                            RETURN
 5907
 5908 034102  004737  034154    MTPD25: CALL    MTPA25          ;WRITE DATA & CHECKBITS
 5909 034106  104471                            ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
 5910 034110  004737  034176                    CALL    MTPB25          ;CHECK FOR NO TRAPS
 5911 034114  104507                            ENA1SBE                 ;DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
 5912 034116  004737  034236                    CALL    MTPC25          ;CHECK FOR EXPECTED TRAP
 5913 034122  000207                            RETURN
```

```
5916 034124  004737  034154          MTPE25: CALL    MTPA25          ;WRITE DATA & CHECKBITS
5917 034130  104471                          ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
5918 034132  004737  034176                  CALL    MTPB25          ;CHECK FOR NO TRAPS
5919                                         ;ENABLE DBE TRAPS
5920 034136  104473                          ECC1INIT                ;INITIALIZE 1 SELECTED MK11 CSR
5921 034140  004737  034236                  CALL    MTPC25          ;CHECK FOR EXPECTED TRAP
5922 034144  104507                          ENA1SBE                 ;DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
5923 034146  004737  034236                  CALL    MTPC25          ;CHECK FOR EXPECTED TRAP
5924 034152  000207                          RETURN
5925
5926                                         ;WRITE TSTDAT & TSTDAT+2 & CHECKBITS
5927 034154  104471                  MTPA25: ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
5928 034156  013771  002240  000000          MOV     TSTDAT,@(R1)    ;WRITE FIRST 16 BITS
5929 034164  104475                          CB1CSR                  ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
5930 034166  013771  002242  000002          MOV     TSTDAT+2,@2(R1) ;WRITE 2ND 16 BITS & CHECKBITS
5931 034174  000207                          RETURN
5932
5933                                         ;CHECK FOR NO TRAP OCCURING CONDITION
5934 034176  012737  034216  002264  MTPB25: MOV     #1$,PARTHERE    ;SETUP TRAP DESTINATION
5935 034204  005771  000000                  TST     @(R1)           ;ACCESS LOCATIONS
5936 034210  005771  000002                  TST     @2(R1)
5937 034214  000207                          RETURN                  ;NO TRAP - GOOD - RETURN
5938
5939 034216  104426                  1$:     READCSR
5940 034220  011137  002032                  MOV     (R1),ADDRESS    ;SAVE VIRTUAL ADDRESS
5941 034224  104024                          ERROR   +24
5942 034226                                  SET     HEADER
5943 034234  000207                          RETURN
5944
5945                                         ;TRAP SHOULD OCCURE TEST
5946 034236  012737  034252  002264  MTPC25: MOV     #1$,PARTHERE    ;SETUP TRAP DESTINATION
5947 034244  005771  000000                  TST     @(R1)           ;ACCESS 1ST LOCATION
5948 034250  000405                          BR      2$              ;NO TRAP - BAD NEWS - SKIP
5949 034252  012737  034302  002264  1$:     MOV     #3$,PARTHERE    ;SETUP TRAP DESTINATION
5950 034260  005771  000002                  TST     @2(R1)          ;ACCESS 2ND LOCATION
5951 034264  104426                  2$:     READCSR                 ;NO TRAP - BAD NEWS
5952 034266  011137  002032                  MOV     (R1),ADDRESS    ;SAVE VIRTUAL ADDRESS
5953 034272  104025                          ERROR   +25
5954 034274                                  SET     HEADER
5955 034302  000207                  3$:     RETURN
5956
```

```
5959 034304                          MTPA26: SUBTST  <<MTPA26        RANDOM DATA (WRITE)>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        MTPA26  RANDOM DATA (WRITE)
                                     ;*****************************************************************************
5960 034304  000137  034354         1$:     JMP     @#MTPC26        ;V177640        GOTO V172360
5961 034310  010221                         MOV     R2,(R1)+        ;V177644
5962 034312  010321                         MOV     R3,(R1)+        ;V177646
5963 034314  077005                         SOB     R0,1$           ;V177650
5964 034316  000207                         RETURN                  ;V177652
5965
5966 034320                          MTPB26: SUBTST  <<MTPB26        RANDOM DATA (READ)>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        MTPB26  RANDOM DATA (READ)
                                     ;*****************************************************************************
5967                                         .DSABL  AMA
5968                                         .ENABL  LSB
5969 034320  000137  034354         1$:     JMP     @#MTPC26        ;V177640        GOTO V172360
5970 034324  020221                         CMP     R2,(R1)+        ;V177644
5971 034326  001401                         BEQ     2$              ;V177646
5972 034330  104451                         PERR25                  ;V177650
5973 034332  005127                 2$:     COM     (PC)+           ;V177652
5974 034334  000000                 RANODD: 0                       ;V177654        FOR ERROR REPORTING
5975 034336  020321                         CMP     R3,(R1)+        ;V177656
5976 034340  001401                         BEQ     3$              ;V177660
5977 034342  104451                         PERR25                  ;V177662
5978 034344  005167  177764         3$:     COM     RANODD          ;V177664
5979 034350  077015                         SOB     R0,1$           ;V177670
5980 034352  000207                         RETURN                  ;V177672
5981                                         .DSABL  LSB
5982                                         .ENABL  AMA
5983
5984 034354                          MTPC26: SUBTST  <<RANDOM NUMBER SUBPROGRAM>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        RANDOM NUMBER SUBPROGRAM
                                     ;*****************************************************************************
5985                                 ;CALLER MUST SETUP
5986                                 ;       MOV     SEEDLO,R3
5987                                 ;       MOV     SEEDHI,R2
5988                                 ;       MOV     R3,R5
5989                                 ;       MOV     R2,R4
5990 034354  073427  000007                 ASHC    #7,R4           ;V172360
5991 034360  060305                         ADD     R3,R5           ;V172364
5992 034362  005504                         ADC     R4              ;V172366
5993 034364  060204                         ADD     R2,R4           ;V172370
5994 034366  062705  001057                 ADD     #1057,R5        ;V172372
5995 034372  000240                         NOP                     ;V172376        GOTO V172260
5996
5997 034374                          MTPD26: SUBTST  <<RANDOM NUMBER SUBSUBPROGRAM>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        RANDOM NUMBER SUBSUBPROGRAM
                                     ;*****************************************************************************
5998 034374  005504                         ADC     R4              ;V172260
5999 034376  062704  047401                 ADD     #47401,R4       ;V172262
6000 034402  010503                         MOV     R5,R3           ;V172266
6001 034404  010402                         MOV     R4,R2           ;V172270
6002 034406  000137  034310                 JMP     @#MTPA26+4      ;V172272        GOTO V177644
```

```
6005 034412                          MTP030: SUBTST  <<MT0030        FLUSH OUT DBE'S>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MT0030  FLUSH OUT DBE'S
                                     ;*****************************************************************************
6006 034412  011002                  1$:     MOV     (R0),R2         ;V177640
6007 034414  010220                          MOV     R2,(R0)+        ;V177642
6008 034416  077103                          SOB     R1,1$           ;V177644
6009 034420  000207                          RETURN                  ;V177646
6010
6011 034422                          MTP031: SUBTST  <<MTP031        SOB-A-LONG TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MTP031  SOB-A-LONG TEST
                                     ;*****************************************************************************
6012                                         .DSABL  AMA
6013 034422  000000                          0                       ;MOVE TERMINATOR
6014 034424  077001                  1$:     SOB     R0,1$           ;SOB TILL R0 UNDERFLOWS
6015 034426  005167  177772                  COM     1$              ;WRITE COMPLEMENT OF SOB
6016 034432  020167  177766                  CMP     R1,1$           ;READ & CHECK FOR NOT 'SOB R0,DOT'
6017 034436  001403                          BEQ     2$              ;OK - SKIP
6018 034440  104454                          PERR30
6019 034442  010167  177756                  MOV     R1,1$
6020 034446  005167  177752          2$:     COM     1$              ;CORRECT SOB INSTRUCTION
6021 034452  010200                          MOV     R2,R0           ;REINITIALIZE SOB CONSTANT
6022                                         ;UPDATE MOVE REGISTERS
6023 034454  010503                          MOV     R5,R3
6024 034456  005725                          TST     (R5)+           ;BUMP (SAFELY) BY 2
6025 034460  010504                          MOV     R5,R4
6026 034462  020537  002472                  CMP     R5,@#LINK1      ;DONE?
6027 034466  001001                          BNE     3$              ;NO - SKIP
6028 034470  000207                          RETURN                  ;YES
6029
6030 034472  014344                  3$:     MOV     -(R3),-(R4)
6031 034474  001376                          BNE     3$
6032 034476  000752                          BR      1$
6033         000056                  SOBLENGTH=.-MTP031
6034                                         .ENABL  AMA
```

```
6062 034500                          MTP032: SUBTST  <<MTP032        WRITE RECOVERY TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MTP032  WRITE RECOVERY TEST
                                     ;*****************************************************************************
6063                                         ;THE TEST ACTUALLY EXECUTED ALREADY IN THE MEMORY UNDER TEST.
6064                                         ;THIS CODE INSURES THAT IT CHANGED MEMORY TO HAVE
6065                                         ;1/2 BANK OF #5141      WHICH IS A ''COM -(R1)'' INSTRUCTION AND
6066                                         ;1/2 BANK OF #110       WHICH IS A ''JMP (R0)''   INSTRUCTION.
6067
6068 034500  012401                  1$:    MOV     (R4)+,R1        ;V177640        ;GET DATA FROM LOWER 1/2 BANK
6069 034502  020102                        CMP     R1,R2           ;V177642        ;IS IT #5141?
6070 034504  001401                        BEQ     2$              ;V177644        ;YES - SKIP
6071 034506  104430                        PERR02                  ;V177646        ;NO - TAKE ERROR TRAP
6072 034510  077305                  2$:    SOB     R3,1$           ;V177650        ;LOOP FOR 1/2 BANK
6073 034512  013703  002472                MOV     @#LINK1,R3      ;V177652        ;RESTORE LOOP SIZE
6074 034516  012400                  3$:    MOV     (R4)+,R0        ;V177656        ;GET DATA FROM UPPER 1/2 BANK
6075 034520  020005                        CMP     R0,R5           ;V177660        ;IS IT #110?
6076 034522  001401                        BEQ     4$              ;V177662        ;YES - SKIP
6077 034524  104427                        PERR01                  ;V177664        ;NO- TAKE ERROR TRAP
6078 034526  077305                  4$:    SOB     R3,3$           ;V177666        ;LOOP FOR 1/2 BANK
6079 034530  000207                        RETURN
```

```
                                MTP033: SUBTST  <<MTP033        BRANCH GOBBLE TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MTP033  BRANCH GOBBLE TEST
                                ;*****************************************************************************
                                        .DSABL  AMA
                                        0                       ;MOVE TERMINATOR
                                BGTEST: 0                       ;TEST WORD (TWO BYTES)
                                BRGOBB: SEC                     ;SET CARRY (TO BE ADDED TO 'BGTEST')
                                        ADCB    (R1)            ;INCREMENT LOW BYTE OF 'BGTEST'
                                        BMI     1$              ;BRANCH WHEN BIT7 IS SET
                                        INCB    (R2)            ;INCREMENT HIGH BYTE OF 'BGTEST'
                                        BR      BRGOBB          ;LOOP 128 TIMES

                                        ;NOW CHECK FOR CORRECT CONDITION CODES
                                1$:     BVS     2$              ;BR IF V-BIT SET (SHOULD BE)
                                        PERR35                  ;NO - REPORT ERROR AND ABORT TEST
                                                                ;COND CODES NOT EQUAL TO 1010
                                2$:     CLV                     ;CLEAR V-BIT
                                        INCB    (R2)            ;INCREMENT HIGH BYTE OF 'BGTEST' ONCE MORE
                                        BCS     3$              ;BR IF C-BIT SET (SHOULD NOT BE)
                                        BVC     3$              ;BR IF V-BIT CLEAR (SHOULD NOT BE)
                                        BMI     4$              ;BR IF N-BIT SET (SHOULD BE)
                                3$:     PERR35                  ;NO - REPORT ERROR AND ABORT TEST
                                                                ;COND CODES NOT EQUAL TO 1010

                                        ;UPDATE TEST POINTERS
                                4$:     MOV     PC,R1
                                5$:     SUB     #5$-BGTEST,R1
                                        MOV     R1,R2
                                        INC     R2

                                        ;UPDATE MOVE REGISTERS
                                        MOV     R5,R3
                                        TST     (R5)+           ;BUMP (SAFELY) BY 2
                                        MOV     R5,R4

                                        ;DONE?
                                        CMP     R5,@#LINK1      ;DONE?
                                        BNE     6$              ;NO - SKIP
                                        RETURN                  ;YES - RETURN

                                        ;MOVE CODE 1 LOCATION
                                6$:     MOV     -(R3),-(R4)
                                        BNE     6$
                                        CLR     (R1)            ;CLEAR TEST WORD 'BGTEST'
                                        BR      BRGOBB          ;RUN MOVED CODE AGAIN
                                GBLENGTH=.-MTP033
                                        .ENABL  AMA
```

```
 6128 034630                    MTP034: SUBTST  <<MTP034        SOFT ERROR - BACKROUND PATTERN TEST>>
                                ;*********************************************************************************
                                ;*SUBTEST        MTP034  SOFT ERROR - BACKROUND PATTERN TEST
                                ;*********************************************************************************
 6129 034630  010220            1$:     MOV     R2,(R0)+        ;V177640
 6130 034632  077102                    SOB     R1,MTP034       ;V177642
 6131 034634  000240                    NOP                     ;V177644
 6132 034636  012401            2$:     MOV     (R4)+,R1        ;V177646
 6133 034640  020102                    CMP     R1,R2           ;V177650
 6134 034642  001402                    BEQ     3$              ;V177652
 6135 034644  104430                    PERR02                  ;V177654
 6136 034646  000240                    NOP                     ;V177656
 6137 034650  077306            3$:     SOB     R3,2$           ;V177660
 6138 034652  000207                    RETURN                  ;V177662
```

```
6140 034654                                MTP035:SUBTST   <<MTP035        WORST CASE NOISE PARITY TEST>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       MTP035  WORST CASE NOISE PARITY TEST
                                           ;*****************************************************************************
6141 034654  012737  000003  002074              MOV     #3,NOPAR        ;SET PARITY TRAPS TO RETURN TO 'PARTHERE'
6142
6143 034662                                      FOR R0 := #FIRST TO #LAST BY #4000
6144 034666  012737  000005  002144                MOV   #BIT2!BIT0,CSR  ;SET WRITE WRONG PARITY & PAR. TRAPS INTO CSR
6145 034674  104425                                LOADCSR
6146 034676  012737  034732  002264                MOV   #1$,PARTHERE
6147 034704  011010                                MOV   (R0),(R0)       ;WWP TEST LOCATION
6148 034706  005710                                TST   (R0)
6149 034710  010037  002032                        MOV   R0,ADDRESS
6150 034714  104050                                ERROR +50
6151 034716  004737  053464                        CALL  PERBNK
6152 034722  032763  002000  002626                BIT   #BIT10,CONFIG+2(R3)
6153 034730  001002                                BNE   2$
6154 034732  104426                           1$:  READCSR
6155 034734  104512                                ERRGEN
6156                                               .
6157 034736  104503                           2$:  CLR1CSR
6158 034740  011010                                MOV   (R0),(R0)       ;CLEAR WRONG PARITY IN MEMORY
6159 034742  012737  000001  002144                MOV   #BIT0,CSR
6160 034750  104425                                LOADCSR
6161 034752  012737  034764  002264                MOV   #3$,PARTHERE
6162 034760  005710                                TST   (R0)
6163 034762  000405                                BR    4$
6164 034764  010037  002032                     3$:  MOV   R0,ADDRESS
6165 034770  104050                                ERROR +50
6166 034772  004737  053464                        CALL  PERBNK
6167 034776                                    4$:  END; OF FOR
6168
6169 035010  005037  002074                   CLR     NOPAR           ;RESET PARITY TRAP ACTION
6170 035014  000207                           RETURN
```

```
                                        .SBTTL  MISC SUBROUTINES

                                REGCOPY:SUBTST  <<SUBR  COPY R0 TO R4,R1 TO R3, & R2 TO R5>>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    COPY R0 TO R4,R1 TO R3, & R2 TO R5
                                ;*****************************************************************************
035016  010004                          MOV     R0,R4
035020  010103                          MOV     R1,R3
035022  010205                          MOV     R2,R5
035024  000207                          RETURN

035026                          FLIPWARN:SUBTST <<FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS>>
                                ;*****************************************************************************
                                ;*SUBTEST        FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS
                                ;*****************************************************************************
035026                                  PUSH    R0
035030  005237  002556                  INC     FLIPLOC
035034  042737  177774  002556          BIC     #^C3,FLIPLOC
035042  022737  000001  002556          CMP     #1,FLIPLOC
035050  001414                          BEQ     1$
035052  022737  000002  002556          CMP     #2,FLIPLOC
035060  001413                          BEQ     2$
035062  022737  000003  002556          CMP     #3,FLIPLOC
035070  001414                          BEQ     3$
035072  005000                          CLR     R0
035074  013704  002554                  MOV     ONES,R4
035100  000414                          BR      4$
035102                          1$:     CLEAR   R0,R4
035106  000411                          BR      4$
035110  012700  000401          2$:     MOV     #401,R0
035114  013704  002554                  MOV     ONES,R4
035120  000404                          BR      4$
035122  012700  000401          3$:     MOV     #401,R0
035126  012704  000401                  MOV     #401,R4
035132  010037  025636          4$:     MOV     R0,WARN2
035136  010037  025652                  MOV     R0,WARN3
035142  010037  025676                  MOV     R0,WARN4
035146  010037  025712                  MOV     R0,WARN5
035152                                  POP     R0
035154  000207                          RETURN
```

```
6207 035156                                BACKGND:SUBTST  <<SUBR  WRITE BACKGROUND>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       SUBR    WRITE BACKGROUND
                                           ;*****************************************************************************
6208                                               ;WRITES DATA FROM R2
6209 035156  104415                                SAVREG
6210 035160  012700  060000                        MOV     #FIRST,R0
6211 035164  012701  040000                        MOV     #SIZE,R1
6212 035170  005737  002424                        TST     NO22BIT
6213 035174  001415                                BEQ     WARN6B
6214 035176  012737  000207  025520  WARN6A:       MOV     #207,MTP000+4          ;WARNING PUTTING 'RETURN' AFTER WRITE
6215 035204  012737  025514  002254                MOV     #MTP000,SUPDOADD
6216 035212  004737  025322                        CALL    SUPD03
6217 035216  012737  000240  025520                MOV     #240,MTP000+4          ;RESTORE 'NOP' AFTER WRITE
6218 035224  104416                                RESREG
6219 035226  000207                                RETURN
6220 035230                                WARN6B: BMOV    MTP000
6221 035236  012737  000207  177644  WARN6:       MOV     #207,UIPAR2            ;WARNING PUTTING 'RETURN' INSTRUCTION AFTER WRITE
6222 035244  004737  025144                        CALL    SUPD01
6223 035250  104416                                RESREG
6224 035252  000207                                RETURN
```

```
6227 035254                          PCONFIG:SUBTST  <<SUBR  PRINT CONFIGURATION MAP>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       SUBR    PRINT CONFIGURATION MAP
                                     ;*****************************************************************************
6228 035254                                  PUSH    TKVEC,TKVEC+2,R0
6229 035266  010637  035570                  MOV     SP,PCONFS               ;SAVE LAST GOOD SP
6230 035272  012737  035536  000060          MOV     #PCONF2,TKVEC
6231 035300  012737  000340  000062          MOV     #340,TKVEC+2
6232 035306  017700  145272                  MOV     @$TKB,R0                ;KILL ANY OLD INTERRUPT
6233 035312  042737  000200  177776          BIC     #BIT7,PSW               ;LOWER CPU PRIORITY TO 140
6234 035320  052777  000100  145254          BIS     #BIT6,@$TKS             ;ENABLE KEYBOARD INTERRUPTS
6235
6236 035326                                  TYPE    MSG001
6237 035332                                  TYPE    MSG002
6238 035336                                  IF LASTBANK EQ #7
6239 035346  012700  000010                    MOV   #8.,R0
6240 035352  010004                          MOV     R0,R4
6241 035354                                    CLEAR R1,R3
6242 035360                                    TYPE  MSG115
6243 035364  004737  035572                    CALL  TCONFIG
6244 035370  000462                            BR    PCONF2
6245 035372                                  END; OF IF LASTBANK
6246 035372                                  TYPE    MSG003
6247                                         ;IF FAT PAPER ON TERMINAL GOTO 1$
6248 035376                                  IF #SW4 SET.IN @SWR THEN JUMPTO PCONF1
6249 035412  012700  000074                  MOV     #60.,R0
6250 035416  010004                          MOV     R0,R4
6251 035420                                  CLEAR   R1,R3
6252 035424                                  TYPE    MSG004
6253 035430  004737  035572                  CALL    TCONFIG                 ;GO TYPE CONFIGURATION (1ST HALF)
6254 035434                                  TYPE    $CRLF
6255 035440                                  TYPE    MSG017                  ;PRINT SPACE(S)
6256 035444                                  TYPE    MSG011
6257 035450                                  TYPE    $CRLF
6258 035454                                  TYPE    MSG017                  ;PRINT SPACE(S)
6259 035460                                  TYPE    MSG012
6260 035464  012701  000360                  MOV     #60.*2*2,R1
6261 035470  010103                          MOV     R1,R3
6262 035472  004737  035572                  CALL    TCONFIG
6263 035476  000417                          BR      PCONF2
6264
6265 035500  012700  000170          PCONF1: MOV     #120.,R0
6266 035504  010004                          MOV     R0,R4
6267 035506                                  CLEAR   R1,R3
6268 035512                                  TYPE    MSG014                  ;SPACE
6269 035516                                  TYPE    MSG011
6270 035522                                  TYPE    MSG004
6271 035526                                  TYPE    MSG012
6272 035532  004737  035572                  CALL    TCONFIG
6273
6274 035536  013706  035570          PCONF2: MOV     PCONFS,SP               ;RESTORE STACK
6275 035542  042777  000100  145032          BIC     #BIT6,@$TKS
6276 035550  117700  145030                  MOVB    @$TKB,R0                ;READ CHAR TO KILL FLAG
6277 035554                                  POP     R0,TKVEC+2,TKVEC
6278 035566  000207                          RETURN
6279
6280 035570  000000          PCONFS: 0                                       ;STACK SAVED HERE!
```

```
6283 035572                                    SUBTST  <<SUBR  TYPE CONFIGURATION>>
                                       ;*****************************************************************************
                                       ;*SUBTEST       SUBR    TYPE CONFIGURATION
                                       ;*****************************************************************************
6284                                   ;*****************************************************************************
6285                                   ;CALL:  MOV     #N,R0           ;N=NUMBER OF CHARACTERS
6286                                   ;       MOV     R0,R4           ;BACKUP
6287                                   ;       MOV     #K.R1           ;INDEX CONSTANT
6288                                   ;       MOV     R1,R3           ;BACKUP
6289                                   ;       CALL    TCONFIG         ;ACTUAL CALL
6290                                   ;       RETURN                  ;ONLY RETURN
6291                                   ;*****************************************************************************
6292
6293                                           ;************
6294                                           ;** ERROR **
6295                                           ;************
6296 035572                           TCONFIG:TYPE    MSG005
6297 035576  032761  000001  002624    1$:     BIT     #BIT0,CONFIG(R1)        ;ERROR ON THIS BANK?
6298 035604  001403                            BEQ     2$                      ;NO - SKIP
6299 035606                                    TYPE    MSG013                  ;PRINT 'X'
6300 035612  000402                            BR      3$
6301 035614                           2$:      TYPE    MSG014                  ;PRINT SPACE
6302 035620  062701  000004           3$:      ADD     #4,R1                   ;BUMP POINTER
6303 035624  077014                            SOB     R0,1$                   ;LOOP TILL DONE
6304 035626  010400                            MOV     R4,R0
6305 035630  010301                            MOV     R3,R1
6306
6307                                           ;************
6308                                           ;** CPU'S **
6309                                           ;************
6310 035632                                    TYPE    MSG008
6311 035636  016105  002624           4$:      MOV     CONFIG(R1),R5
6312 035642  006205                            ASR     R5                      ;GET CPU BITS
6313 035644  042705  177760                    BIC     #^C17,R5                ;CLEAR NON INTERESTING BITS
6314 035650  005705                            TST     R5                      ;IS THERE ANYTHING THERE?
6315 035652  001003                            BNE     8$                      ;YES - BRANCH.
6316 035654  112705  000040                    MOVB    #' ,R5                  ;NO - MOVE A BLANK INTO R5
6317 035660  000402                            BR      9$                      ;BRANCH OVER NEXT INSTRUCTION
6323 035662  062705  000060           8$:      ADD     #60,R5                  ;MAKE ASCII
6324 035666  110537  073603           9$:      MOVB    R5,MSG015               ;PLUG INTO MEMORY
6325 035672                                    TYPE    MSG015
6326 035676  062701  000004                    ADD     #4,R1                   ;BUMP POINTER
6327 035702  077023                            SOB     R0,4$                   ;LOOP TILL DONE
6328 035704  010400                            MOV     R4,R0
6329 035706  010301                            MOV     R3,R1
```

```
6332                                            ;*****************
6333                                            ;** INTERLEAVE **
6334                                            ;*****************
6335 035710                                     TYPE    MSG007
6336                                            ;THIS IS AN ENTRY POINT FROM ERROR REPORTS
6337 035714  032761  010000  002626  TCFIG1:    BIT     #BIT12,CONFIG+2(R1)
6338 035722  001014                            BNE     1$
6339 035724  032761  000002  002624            BIT     #BIT1,CONFIG(R1)        ;IS THERE ANY MEMORY HERE?
6340 035732  001004                            BNE     18$                     ;BRANCH IF MEMORY PRESENT.
6341 035734  112737  000040  073603            MOVB    #' ,MSG015              ;MOVE A BLANK IN TO BE PRINTED
6342 035742  000424                            BR      16$                     ;BRANCH TO TYPE ROUTINE
6343 035744  112737  000055  073603  18$:      MOVB    #'-,MSG015
6344 035752  000420                            BR      16$
6345 035754  016105  002624          1$:       MOV     CONFIG(R1),R5
6346 035760  042705  007777                    BIC     #^C170000,R5            ;GET CSR INTERLEAVE
6347 035764  000305                            SWAB    R5
6348 035766  072527  177774                    ASH     #-4,R5
6349 035772  022705  000012                    CMP     #10.,R5
6350 035776  100002                            BPL     2$
6351 036000  062705  000007                    ADD     #7,R5
6352 036004  062705  000060          2$:       ADD     #60,R5                  ;MAKE ASCII
6353 036010  110537  073603                    MOVB    R5,MSG015               ;PLUG INTO MEMORY
6354 036014                          16$:       TYPE    MSG015
6355 036020                                     IF NOTAB NE #0 THEN $RETURN
6356 036030  062701  000004                    ADD     #4,R1                   ;BUMP POINTER
6357 036034  077051                            SOB     R0,TCFIG1               ;LOOP TILL DONE
6358 036036  010400                            MOV     R4,R0
6359 036040  010301                            MOV     R3,R1
6360
6361                                            ;******************
6362                                            ;** MEMORY TYPE **
6363                                            ;******************
6364                                            .ENABL  LSB
6365 036042                                     TYPE    MSG009
6366 036046  033761  002104  002624  TCFIG2:    BIT     CPUBIT,CONFIG(R1)
6367 036054  001447                            BEQ     17$
6368 036056  016105  002626                    MOV     CONFIG+2(R1),R5
6369 036062  000305                            SWAB    R5                      ;GET MEMORY TYPE
6370 036064  042705  177770                    BIC     #^C7,R5                 ;CLEAR NON INTERESTING BITS
6371 036070  005705                            TST     R5
6372 036072  001440                            BEQ     17$
6373 036074  032705  000004                    BIT     #BIT2,R5
6374 036100  001004                            BNE     4$
6375 036102  112737  000102  073603            MOVB    #'B,MSG015
6376 036110  000434                            BR      8$
6377 036112  032705  000002          4$:       BIT     #BIT1,R5
6378 036116  001013                            BNE     6$
6379 036120  032705  000001                    BIT     #BIT0,R5
6380 036124  001004                            BNE     5$
6381 036126  112737  000115  073603            MOVB    #'M,MSG015
6382 036134  000422                            BR      8$
6383 036136  112737  000113  073603  5$:       MOVB    #'K,MSG015
6384 036144  000416                            BR      8$
6385 036146  032705  000001          6$:       BIT     #BIT0,R5
6386 036152  001004                            BNE     7$
6387 036154  112737  000114  073603            MOVB    #'L,MSG015
6388 036162  000407                            BR      8$
```

```
 6389 036164  112737  000120  073603  7$:     MOVB    #'P,MSG015
 6390 036172  000403                  BR      8$
 6391 036174  112737  000040  073603  17$:    MOVB    #' ,MSG015
 6392 036202                          8$:     TYPE    MSG015
 6393 036206                                  IF NOTAB NE #0 THEN $RETURN
 6394 036216  062701  000004                  ADD     #4,R1                   ;BUMP POINTER
 6395 036222  077067                          SOB     R0,TCFIG2               ;LOOP TILL DONE
 6396 036224  010400                          MOV     R4,R0
 6397 036226  010301                          MOV     R3,R1
 6398                                         .DSABL  LSB
 6399
 6400                                         ;*********
 6401                                         ;** CSR **
 6402                                         ;*********
 6403 036230                                  TYPE    MSG016
 6404 036234  112737  000040  073603  TCFIG3: MOVB    #' ,MSG015
 6405 036242  016105  002624                  MOV     CONFIG(R1),R5
 6406 036246  032705  000002                  BIT     #BIT1,R5
 6407 036252  001414                          BEQ     16$
 6408 036254  042705  170377                  BIC     #^C7400,R5
 6409 036260  000305                          SWAB    R5
 6410 036262  022705  000012                  CMP     #10.,R5
 6411 036266  100002                          BPL     10$
 6412 036270  062705  000007                  ADD     #7,R5
 6413 036274  062705  000060          10$:    ADD     #60,R5                  ;MAKE ASCII
 6414 036300  110537  073603                  MOVB    R5,MSG015               ;PLUG INTO MEMORY
 6415 036304                          16$:    TYPE    MSG015
 6416 036310                                  IF NOTAB NE #0 THEN $RETURN
 6417 036320  062701  000004                  ADD     #4,R1                   ;BUMP POINTER
 6418 036324  077035                          SOB     R0,TCFIG3               ;LOOP TILL DONE
 6419 036326  010400                          MOV     R4,R0
 6420 036330  010301                          MOV     R3,R1
 6421
 6422                                         ;****************
 6423                                         ;** PROTECTED **
 6424                                         ;****************
 6425 036332                                  TYPE    MSG010
 6426 036336  105761  002624          11$:    TSTB    CONFIG(R1)              ;BANK PROTECTED?
 6427 036342  100004                          BPL     12$                     ;NO - SKIP
 6428 036344  112737  000120  073603          MOVB    #'P,MSG015
 6429 036352  000407                          BR      13$
 6430 036354  032761  000100  002624  12$:    BIT     #BIT6,CONFIG(R1)        ;PROTECTED REGION OF ECC?
 6431 036362  001406                          BEQ     14$                     ;NO - SKIP
 6432 036364  112737  000111  073603          MOVB    #'I,MSG015
 6433 036372                          13$:    TYPE    MSG015
 6434 036376  000402                          BR      15$
 6435 036400                          14$:    TYPE    MSG014                  ;PRINT SPACE
 6436 036404  062701  000004          15$:    ADD     #4,R1                   ;BUMP POINTER
 6437 036410  077026                          SOB     R0,11$                  ;LOOP TILL DONE
 6438 036412  010400                          MOV     R4,R0
 6439 036414  010301                          MOV     R3,R1
 6440 036416  000207                          RETURN
```

```
                                        .SBTTL  TRAP    PARITY ERROR HANDLER
                                ;*******************************************************************************
                                ;VECTOR TO HERE FROM TRAPS TO 114
                                ;IGNORE ERRORS BUT COUNT IF NOPAR FLAG = 1.
                                ;*******************************************************************************
                                ;
                                ;       CODE    ACTION
                                ;
                                ;       ---     ------
                                ;         0     PRINT UNEXPECTED PARITY TRAP
                                ;         1     COUNT ERROR
                                ;         2     SET ''ABORT'' / SETUP 'BADPC'' / RETURN VIA PCBUMP
                                ;         3     RETURN VIA 'PARTHERE''
                                ;
036420  022737  000001  002074  PARITY: CMP     #1,NOPAR                ;COUNTING PARITY ERRORS?
036426  001003                        BNE     1$                      ;NO - SKIP
036430  005237  002070                INC     PARCNT                  ;PARITY ERROR COUNTER + 1
036434  000002                        RTI
036436  022737  000002  002074  1$:   CMP     #2,NOPAR                ;ACTION CODE = 2 ?
036444  001013                        BNE     2$                      ;NO - SKIP
036446                                SET     ABORTFLAG               ;YES
036454  004737  036626                CALL    BADSTACK                ;FIND BAD SP,PC,PSW OFF STACK
036460  063716  002276                ADD     PCBUMP,(SP)             ;UPDATE RETURN PC
036464  042766  000004  000002        BIC     #BIT2,2(SP)             ;SHOW FAILURE BY .NE.
036472  000002                        RTI
036474  022737  000003  002074  2$:   CMP     #3,NOPAR                ;ACTION CODE = 3 ?
036502  001003                        BNE     3$                      ;NO - SKIP
036504  013716  002264                MOV     PARTHERE,(SP)
036510  000002                        RTI
036512  004737  036626          3$:   CALL    BADSTACK                ;FIND BAD SP,PC,PSW OFF STACK
036516                                FATAL   32
```

```
                                        .SBTTL  TRAP    NON-EXISTANT MEMORY (HOLES) HANDLER
                                ;*******************************************************************************
                                ;VECTOR TO HERE (SOMETIMES) FROM TRAPS TO 4
                                ;       CODE IN NONEM DETERMINES ACTION AS FOLLOWS:
                                ; 1)    IGNORE ERRORS BUT COUNT IF NONEM (NO NON-EXISTANT MEMORY) FLAG = 1.
                                ; 2)    TO EXIT PATTERN 0 DURING SIZING IF NON-EXIST MEM ERROR
                                ;*******************************************************************************
036524  022737  000001  002076  NONEXIST:CMP     #1,NONEM                ;COUNTING NON-EXISTANT MEMORY ERRORS?
036532  001011                          BNE     2$                      ;NO - SKIP
036534  005237  002066                  INC     NEMCNT                  ;BUMP NON-EXISTANT MEMORY COUNTER
036540  022737  000001  002066          CMP     #1,NEMCNT               ;FIRST ERROR?
036546  001002                          BNE     1$                      ;NO - SKIP
036550  010037  002032                  MOV     R0,ADDRESS              ;ASSUME R0 CONTAINS THE ADDRESS ACCESSED
036554  000002                  1$:     RTI
036556  005237  002066          2$:     INC     NEMCNT                  ;BUMP NON-EXISTANT MEMORY COUNTER
036562  012701  000001                  MOV     #1,R1                   ;DUMMY UP R1 FOR A FORCED SOB EXIT
036566  000002                          RTI

                                ;*******************************************************************************
                                        .SBTTL  TRAP    TIMEOUT (TRAP TO 4) HANDLER
036570  004737  036626          TIMEOUT:CALL    BADSTACK                ;FIND BAD SP,PC,PSW OFF STACK
036574                                  FATAL   6
                                ;*******************************************************************************
                                        .SBTTL  TRAP    MEMORY MANAGEMENT (TRAP TO 250) HANDLER
036602  004737  036626          MMTRAP: CALL    BADSTACK                ;FIND BAD SP,PC,PSW OFF STACK
036606                                  FATAL   7
                                        .SBTTL  TRAP    RESERVED INSTRUCTION HANDLER
036614  004737  036626          PDP1105:CALL    BADSTACK                ;FIND BAD SP,PC,PSW OFF STACK
036620                                  FATAL   5


036626                          BADSTACK:SUBTST <<FIND BAD SP, PC, & PSW FROM STACK>>
                                ;*******************************************************************************
                                ;*SUBTEST       FIND BAD SP, PC, & PSW FROM STACK
                                ;*******************************************************************************
036626  010637  002024                  MOV     SP,BADSP
036632  062737  000002  002024          ADD     #2,BADSP
036640  016637  000002  002020          MOV     2(SP),BADPC
036646  016637  000004  002030          MOV     4(SP),BADPSW
036654  000207                          RETURN
```

```
                                        .SBTTL  TRAP    KERNEL TRAP HANDLER
                                ;*******************************************************************************
                                ;KERNEL IS A TRAP THAT COMES HERE
                                ;*******************************************************************************
036656  042766  140000  000002  $KERNEL:      BIC     #140000,2(SP)
036664  000002                  RTI
                                ;*******************************************************************************
                                        .SBTTL  TRAP    ENERGIZE TRAP HANDLER
036666  052737  000001  177572  $ENERGIZE:BIS   #BIT0,MMR0
036674  000002                  RTI
                                ;*******************************************************************************
                                        .SBTTL  TRAP    DEENERGIZE TRAP HANDLER
036676  042737  000001  177572  $DEENERGIZE:BIC #BIT0,MMR0
036704  000002                  RTI
                                ;*******************************************************************************
                                        .SBTTL  TRAP    CACHON TRAP HANDLER
036706  005737  002514          $CACHN: TST     CACHKN          ;IS THERE A CACHE
036712  001406                  BEQ     1$              ;NO - RETURN
036714  013737  002514  177746  MOV     CACHKN,CONTRL   ;SETUP CACHE AS PER CONSTANT (USUALLY 1 = FULLY ON)
036722  052737  000001  177746  BIS     #BIT0,CONTRL    ;DISABLE TRAPS (BUT NOT ABORTS)
036730  000002                  1$:     RTI
                                ;*******************************************************************************
                                        .SBTTL  TRAP    CACHOFF TRAP HANDLER
036732  005737  002514          $CACHF: TST     CACHKN          ;IS THERE A CACHE?
036736  001403                  BEQ     1$              ;NO - RETURN
                                ;DISABLE TRAPS (NOT ABORTS), FORCE MISSES, FLUSH, BYPASS
036740  053737  002520  177746  BIS     CACHKF,CONTRL
036746  000002                  1$:     RTI
```

```
                                                .SBTTL  TRAP    LOAD CSR TRAP HANDLER
                                                ;LOAD CORRECT CSR WITH DATA IN CSR
                                                ;PROGRAM CSR'S ASSERT INHIBIT MODE POINTER WHEN LOADED
036750                                  $LOADC: PUSH    R0,R1                   ;SAVE REGISTERS
036754  013700  002146                        MOV     CSRNO,R0                ;CREATE CSR ADDRESS
036760                                        IF INHECC IS TRUE THEN GOTO 3$  ;DON'T WANT INH. MODE POINTER ON
036766  005737  002502                        TST     PGMCSR                  ;PROGRAM IN INTERLEAVED SPACE?
036772  100007                                BPL     1$                      ;BRANCH IF NOT
036774  113701  002503                        MOVB    PGMCSR+1,R1             ;CHECK SECOND CSR
037000  042701  177740                        BIC     #^C37,R1                ;CLEAR UNNECESSARY BITS
037004  020137  002146                        CMP     R1,CSRNO                ;IS THIS THE CURRENT CSR?
037010  001404                                BEQ     2$                      ;BRANCH IF IT IS
037012  123737  002502  002146    1$:         CMPB    PGMCSR,CSRNO            ;IS THIS THE CURRENT CSR?
037020  001003                                BNE     3$                      ;BRANCH IF NOT
037022  052737  020000  002144    2$:         BIS     #BIT13,CSR              ;SET THE INHIBIT MODE POINTER TO  1ST 16K
037030  013760  002144  172100    3$:         MOV     CSR,CSRADD(R0)          ;LOAD THE CSR
037036                                        POP     R1,R0                   ;RESTORE REGISTERS
037042  000002                                RTI
                                                .SBTTL  TRAP    READ CSR TRAP HANDLER
                                                ;READ THE CORRECT CSR INTO LOCATIONS CSR
037044                                  $READC: PUSH    R0
037046  013700  002146                        MOV     CSRNO,R0
037052  016037  172100  002144                MOV     CSRADD(R0),CSR          ;READ IT
037060                                        POP     R0
037062  000002                                RTI
```

```
                                                .SBTTL  TRAP    TEST (R1) & READ CSR CAREFULLY
037064                                  $TSTRD: PUSH    R0,R2,R3
037072  012700  172100                        MOV     #CSRADD,R0              ;CREATE CSR ADDRESS
037076  063700  002146                        ADD     CSRNO,R0
037102  005002                                CLR     R2
037104  005737  002502                        TST     PGMCSR
037110  100007                                BPL     1$
037112  113703  002503                        MOVB    PGMCSR+1,R3
037116  042703  000200                        BIC     #BIT7,R3
037122  020337  002146                        CMP     R3,CSRNO
037126  001404                                BEQ     2$
037130  123737  002502  002146          1$:   CMPB    PGMCSR,CSRNO
037136  001002                                BNE     3$
037140  012702  020000                  2$:   MOV     #BIT13,R2
037144  005737  002424                  3$:   TST     NO22BIT         ;IS THIS AN 11/44?
037150  001403                                BEQ     4$              ;BRANCH IF IT IS
037152  004737  037240                        CALL    TSTRD1
037156  000405                                BR      5$
037160                                  4$:   BMOV    TSTRD1
037166  004737  177640                        CALL    FASTCITY                ;CALL TO THE USER INSTRUCTION PAR'S
                                              ;IF SINGLE BIT ERROR ONLY - SET CARRY BIT
037172                                  5$:   POP     R3,R2,R0
037200                                        IF #BIT4 SET.IN CSR AND #BIT15 OFF.IN CSR
037220  052766  000001  000002                  BIS   #BIT0,2(SP)
037226                                        ELSE
037230  042766  000001  000002                  BIC   #BIT0,2(SP)
037236                                        END ;OF IF #BIT4
037236  000002                                RTI

037240  010210                  TSTRD1: MOV     R2,(R0)                 :V177640
037242                                  TESTAREA                        :V177642 ;ENTER SUPERVISOR MODE
037250  105711                          TSTB    (R1)                    :V177646
037252  042737  140000  177776          BIC     #BIT15!BIT14,PSW        :V177650
037260  011037  002144                  MOV     (R0),CSR                :V177656
037264  000207                          RETURN                          :V177662
```

```
                                                .SBTTL  TRAP    ECC DISABLE ALL CSR'S TRAP HANDLER
037266  012737  000002  002144  $ECCDIS:MOV     #BIT1,CSR
037274  004737  040012                  CALL    CSROUT
037300  000002                          RTI
                                                .SBTTL  TRAP    ECC DISABLE OF 1 SELECTED CSR TRAP HANDLER
037302  012737  000002  002144  $ECC1DIS:MOV    #BIT1,CSR
037310  104425                          LOADCSR
037312  000002                          RTI
                                                .SBTTL  TRAP    INITIALIZE ALL CSR'S TRAP HANDLER
037314  012737  000001  002144  $ECCINIT:MOV    #BIT0,CSR
037322  004737  040012                  CALL    CSROUT
037326  000002                          RTI
                                                .SBTTL  TRAP    INITIALIZE 1 SELECTED CSR TRAP HANDLFR
037330  012737  000001  002144  $ECC1INIT:MOV   #BIT0,CSR
037336  104425                          LOADCSR
037340  000002                          RTI
                                                .SBTTL  TRAP    ENABLE SBE PARITY TRAPS ON ALL CSR'S
037342  012737  000003  002144  $ENASBE:MOV     #BIT0!BIT1,CSR
037350  004737  040012                  CALL    CSROUT
037354  000002                          RTI
                                                .SBTTL  TRAP    ENABLE SBE PARITY TRAPS ON 1 SELECTED CSR
037356  012737  000003  002144  $ENA1SBE:MOV    #BIT0!BIT1,CSR
037364  104425                          LOADCSR
037366  000002                          RTI
                                                .SBTTL  TRAP    WRITE CHECKBITS THRU ALL CSR'S TRAP HANDLER
037370  013737  002274  002144  $CBCSR: MOV     CHECK,CSR               ;BITS 11-5
037376  052737  000006  002144          BIS     #BIT1!BIT2,CSR          ;CHECK MODE
037404  004737  040012                  CALL CSROUT
037410  000002                          RTI
                                                .SBTTL  TRAP    WRITE CHECKBITS THRU 1 SELECTED CSR TRAP HANDLER
037412  013737  002274  002144  $CB1CSR:MOV     CHECK,CSR               ;BITS 11-5
037420  052737  000006  002144          BIS     #BIT1!BIT2,CSR          ;CHECK MODE
037426  104425                          LOADCSR
037430  000002                          RTI
```

```
                                                .SBTTL  TRAP    WAS THERE A SBE ON ANY CSR TRAP HANDLER
037432                                  $WASSBE:PUSH    R1,R4
037436  013701  002216                        MOV     TOTCSRS,R1      ;GET CSR'S BYTE
037442  005004                                CLR     R4
037444                                        BEGIN LWSBE
037444                                          FOR CSRNO := #0 TO #36 BY #2
037450  006301                                    ASL         R1
037452                                            ON.ERROR
037454  104426                                      READCSR
037456                                              IF #BIT4 SET.IN CSR
037466                                                SET     R4
037472                                                LEAVE LWSBE
037474                                              END ;OF IF #BIT4
037474                                            END ;OF ON.ERROR
037474                                            IF R1 EQ #0 THEN LEAVE LWSBE
037500                                          END ;OF FOR CSRNO
037516                                        END LWSBE
037516  006004                                ROR     R4              ;SET C BIT FOR ERROR
037520                                        POP     R4,R1
037524                                        ON.ERROR
037526  052766  000001  000002                  BIS   #BIT0,2(SP)
037534                                        ELSE
037536  042766  000001  000002                  BIC   #BIT0,2(SP)
037544                                        END ;OF ON.ERROR
037544  000002                                RTI
                                                .SBTTL  TRAP    WAS THERE A SBE IN 1 SELECTED CSR TRAP HANDLER
                                                ;ON RETURN IF CARRY IS SET THERE WAS A SBE
037546  104426                          $WAS1SBE:READCSR
037550  042766  000001  000002                BIC     #BIT0,2(SP)     ;CLR C BIT ON STACK
037556  032737  000020  002144                BIT     #BIT4,CSR
037564  001403                                BEQ     1$
037566  052766  000001  000002                BIS     #BIT0,2(SP)     ;SET C BIT ON STACK
037574  000002                          1$:   RTI
```

```
                                                .SBTTL  TRAP    WAS THERE A DBE ON ANY CSR TRAP HANDLER
                                    $WASDBE:PUSH    R1,R4
                                            MOV     TOTCSRS,R1      ;GET CSR'S BYTE
                                            CLR     R4
                                            BEGIN LWDBE
                                              FOR CSRNO := #0 TO #36 BY #2
                                                ASL         R1
                                                ON.ERROR
                                                  READCSR
                                                  IF #BIT15 SET.IN CSR
                                                    SET     R4
                                                    LEAVE LWDBE
                                                  END ;OF IF #BIT4
                                                END ;OF ON.ERROR
                                              IF R1 EQ #0 THEN LEAVE LWDBE
                                              END ;OF FOR CSRNO
                                            END LWDBE
                                            ROR     R4              ;SET C BIT FOR ERROR
                                            POP     R4,R1
                                            ON.ERROR
                                              BIS   #BIT0,2(SP)
                                            ELSE
                                              BIC   #BIT0,2(SP)
                                            END ;OF ON.ERROR
                                            RTI
                                                .SBTTL  TRAP    WAS THERE A DBE ON 1 SELECTED CSR TRAP HANDLER
                                                ;ON RETURN IF CARRY IS SET THERE WAS A DBE
                                    $WAS1DBE:READCSR
                                            TST     CSR             ;DBE?
                                            BPL     3$              ;NO - SKIP
                                            BIS     #BIT0,2(SP)     ;SET C BIT ON STACK
                                            RTI
                                    3$:     BIC     #BIT0,2(SP)     ;CLR C BIT ON STACK
                                            RTI
```

```
6722                                                    .SBTTL  TRAP    CLEAR ALL ECC CSR'S TRAP HANDLER
6723 037742                                    $CLRCSR:CLEAR   CSR
6724 037746  004737  040012                            CALL    CSROUT
6725 037752  000002                                    RTI
6726                                                    .SBTTL  TRAP    CLEAR 1 SELECTED CSR TRAP HANDLER
6727 037754                                    $CLR1CSR:CLEAR  CSR
6728 037760  104425                                    LOADCSR
6729 037762  000002                                    RTI
6730                                                    .SBTTL  TRAP    ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN ALL CSR'S TRAP HANDLER
6731                                                    ;CHECKBITS ALREADY IN LOC ''CSR''
6732 037764  052737  000006  002144  $CHKDIS:BIS     #BIT1!BIT2,CSR          ;ECC DISABLE & DIAG CHECK MODE
6733 037772  004737  040012                            CALL    CSROUT
6734 037776  000002                                    RTI
6735                                                    .SBTTL  TRAP    ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN 1 SELECTED CSR
6736                                                    ;CHECKBITS ALREADY IN LOC ''CSR''
6737 040000  052737  000006  002144  $CHK1DIS:BIS    #BIT1.BIT2,CSR          ;ECC DISABLE & DIAG CHECK MODE
6738 040006  104425                                    LOADCSR
6739 040010  000002                                    RTI
```

```
                                CSROUT: SUBTST  <<SUBR  WRITE IN ALL CSR'S>>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    WRITE IN ALL CSR'S
                                ;*****************************************************************************
040012                                  PUSH    R1
040014  013701  002216                  MOV     TOTCSRS,R1      ;GET CSR'S BYTE
040020                                  BEGIN LCSROUT
040020                                    FOR CSRNO := #0 TO #36 BY #2
040024  006301                              ASL         R1
040026                                      ON.ERROR
040030  104425                                LOADCSR
040032                                      END ;OF ON.ERROR
040032                                    IF R1 EQ #0 THEN LEAVE LCSROUT
040036                                    END ;OF FOR CSRNO
040054                                  END LCSROUT
040054                                  POP     R1
040056  000207                          RETURN

040060                          $INVALID:       SUBTST  <<TRAP  INVALIDATE BACKGROUND PATTERN>>
                                ;*****************************************************************************
                                ;*SUBTEST        TRAP    INVALIDATE BACKGROUND PATTERN
                                ;*****************************************************************************
040060                                  PUSH    R0,R1
040064  013701  002100                  MOV     BANK,R1
040070  006301                          ASL     R1
040072  006301                          ASL     R1
040074  042761  020000  002626          BIC     #BIT13,CONFIG+2(R1)
040102                                  POP     R1,R0
040106  000002                          RTI
```

```
6766 040110                                $ERRGEN:    SUBTST<<TRAP    GENERATE AND TEST ERROR ADDRESS>>
                                           ;*****************************************************************************
                                           ;*SUBTEST    TRAP    GENERATE AND TEST ERROR ADDRESS
                                           ;*****************************************************************************
6767 040110                                        PUSH    R0,R1,R2,R3
6768 040120  013703  002102                       MOV     BANKINDEX,R3
6769 040124  005737  002426                       TST     NOSUPER
6770 040130  001003                               BNE     6$
6771 040132  013700  172246                       MOV     SIPAR3,R0               ;GENERATE WHAT ERROR ADDR SHOULD BE
6772 040136  000402                               BR      7$
6773 040140  013700  177646               6$:     MOV     UIPAR3,R0
6774 040144  072027  177773               7$:     ASH     #-5,R0
6775 040150  005737  002130                       TST     EUFLAG
6776 040154  001002                               BNE     1$
6777 040156  042700  177600                       BIC     #^C177,R0
6778 040162  000301                       1$:     SWAB    R1                      ;GET CURRENT ADDRESS BITS 11 AND 12
6779 040164  006201                               ASR     R1
6780 040166  006201                               ASR     R1
6781 040170  006201                               ASR     R1
6782 040172  042701  177775                       BIC     #^C2,R1
6783 040176  060100                               ADD     R1,R0                   ;ADD THEM TO THE ADJUSTED PAR VALUE
6784                                              ;GET ERROR ADDRESS FROM CSR UNDER TEST
6785 040200  013701  002144                       MOV     CSR,R1
6786 040204  072127  177773                       ASH     #-5,R1
6787 040210  042701  177600                       BIC     #^C177,R1
6788 040214  005737  002424                       TST     NO22BIT                 ;DO WE LOOK FOR AN EUB BIT?
6789 040220  001024                               BNE     2$                      ;BRANCH IF NOT NECESSARY
6790 040222  005737  002130                       TST     EUFLAG                  ;IS IT EUB?
6791 040226  001421                               BEQ     2$                      ;BRANCH IF NOT
6792 040230                                       PUSH    R0                      ;SAVE GENERATED ERROR ADDRESS
6793 040232  013702  002146                       MOV     CSRNO,R2                ;GET CSR NUMBER
6794 040236  052762  040000  172100               BIS     #BIT14,CSRADD(R2)       ;TURN ON EUB BIT CAREFULLY
6795 040244  016200  172100                       MOV     CSRADD(R2),R0           ;GET CSR CONTENTS
6796 040250  042762  040000  172100               BIC     #BIT14,CSRADD(R2)       ;TURN OFF EUB BIT CAREFULLY
6797 040256  042700  177037                       BIC     #^C740,R0               ;CLEAR EVERYTHING BUT ERROR ADDR
6798 040262  006300                               ASL     R0
6799 040264  006300                               ASL     R0                      ;SHIFT ADDR BITS 18-21 INTO POSITION
6800 040266  060001                               ADD     R0,R1                   ;ADD TO CURRENT ERROR ADDRESS
6801 040270                                       POP     R0
6802 040272  020001                       2$:     CMP     R0,R1                   ;COMPARE REAL AND GENERATED ERR. ADDR.
6803 040274  001420                               BEQ     5$                      ;BRANCH IF THEY ARE THE SAME
6804 040276  005737  002134                       TST     INTFLAG                 ;INTERLEAVED?
6805 040302  001411                               BEQ     3$                      ;NO - WE HAVE AN ERROR
6806 040304  062700  000100                       ADD     #100,R0
6807 040310  005737  002136                       TST     INT64K                  ;64K INTERLEAVED MEMORY?
6808 040314  001002                               BNE     4$
6809 040316  062700  000100                       ADD     #100,R0
6810 040322  020001                       4$:     CMP     R0,R1
6811 040324  001404                               BEQ     5$
6812 040326  005737  002064               3$:     TST     SKPERR                  ;ARE WE SUPPOSED TO SKIP ERROR P.O.?
6813 040332  001001                               BNE     5$                      ;YES - SKIP ERROR PRINTOUT
6814 040334  104462                               PERR36                          ;ELSE PRINT ERROR ADDRESS ERROR
6815 040336  010137  002430               5$:     MOV     R1,ERRADD               ;SAVE CSR'S ERROR ADDRESS
6816 040342  005037  002064                       CLR     SKPERR                  ;ENABLE THE ERROR PRINTOUT AGAIN
6817 040346                                       POP     R3,R2,R1,R0             ;RESTORE REGISTERS
6818 040356  000002                               RTI
```

```
 6821 040360                          CHKGEN: SUBTST<<SUBR    GENERATE CHECK BITS>>
                                      ;*********************************************************************************
                                      ;*SUBTEST       SUBR    GENERATE CHECK BITS
                                      ;*********************************************************************************
 6822                                         ;CHECK BIT GENERATOR ROUTINE
 6823                                         ;CALLING SEQUENCE IS:
 6824                                         ;       MOV     #WORD1,SOURCE   ;SOURCE = ADDRESS OF DATA
 6825                                         ;       CALL    CHKGEN
 6826                                         ;
 6827                                         ;CHECK BITS RETURNED IN BITS 11-5 OF LOCATION CHECK
 6828                                         ;
 6829 040360                                 PUSH    R0,R1,R2,R3,R4,R5
 6830 040374  012702  000077                 MOV     #77,R2          ;DEFAULT CHECKBITS FOR DOUBLE WORD OF ZEROS
 6831 040400  012703  040466                 MOV     #CHKTAB,R3      ;ADDRESS OF CHECKBIT TABLE
 6832 040404  013705  002272                 MOV     SOURCE,R5       ;GET SOURCE ADDRESS
 6833 040410  012501                         MOV     (R5)+,R1        ;GET LSB'S
 6834 040412  011500                         MOV     (R5),R0         ;GET MSB'S
 6835
 6836 040414  006704                  1$:    SXT     R4              ;EXTEND SIGN OF DOUBLE WORD TO R4
 6837 040416  142304                         BICB    (R3)+,R4        ;ELIMINATE BITS THAT DON'T COUNT
 6838 040420  074402                         XOR     R4,R2           ;COMPLEMENT MASKED BITS IN CHECKBITS
 6839 040422  073027  000001                 ASHC    #1,R0           ;DOUBLE PRECISION LEFT SHIFT R0,,R1
 6840 040426  001372                         BNE     1$              ;LOOP TILL ALL BITS ARE CHECKED
 6841
 6842 040430  042702  177600                 BIC     #^C177,R2               ;KILL ALL JUNK BITS
 6843 040434  000302                         SWAB    R2              ;POSITION CHECKBITS IN BITS 11-5
 6844 040436  006202                         ASR     R2
 6845 040440  006202                         ASR     R2
 6846 040442  006202                         ASR     R2
 6847 040444  010237  002274                 MOV     R2,CHECK
 6848 040450                                 POP     R5,R4,R3,R2,R1,R0
 6849 040464  000207                         RETURN
```

```
6852 040466                    CHKTAB: ;BYTE #3
6853 040466  200                       .BYTE   ^C177           ;BIT 31
6854 040467  301                       .BYTE   ^C076           ;BIT 30
6855 040470  302                       .BYTE   ^C075           ;BIT 29
6856 040471  203                       .BYTE   ^C174           ;BIT 28
6857 040472  304                       .BYTE   ^C073           ;BIT 27
6858 040473  205                       .BYTE   ^C172           ;BIT 26
6859 040474  206                       .BYTE   ^C171           ;BIT 25
6860 040475  307                       .BYTE   ^C070           ;BIT 24
6861                                   ;BYTE #2
6862 040476  310                       .BYTE   ^C067           ;BIT 23
6863 040477  211                       .BYTE   ^C166           ;BIT 22
6864 040500  212                       .BYTE   ^C165           ;BIT 21
6865 040501  313                       .BYTE   ^C064           ;BIT 20
6866 040502  214                       .BYTE   ^C163           ;BIT 19
6867 040503  315                       .BYTE   ^C062           ;BIT 18
6868 040504  316                       .BYTE   ^C061           ;BIT 17
6869 040505  217                       .BYTE   ^C160           ;BIT 16
6870                                   ;BYTE #1
6871 040506  320                       .BYTE   ^C057           ;BIT 15
6872 040507  221                       .BYTE   ^C156           ;BIT 14
6873 040510  222                       .BYTE   ^C155           ;BIT 13
6874 040511  323                       .BYTE   ^C054           ;BIT 12
6875 040512  224                       .BYTE   ^C153           ;BIT 11
6876 040513  325                       .BYTE   ^C052           ;BIT 10
6877 040514  326                       .BYTE   ^C051           ;BIT  9
6878 040515  227                       .BYTE   ^C150           ;BIT  8
6879                                   ;BYTE #0
6880 040516  340                       .BYTE   ^C037           ;BIT  7
6881 040517  241                       .BYTE   ^C136           ;BIT  6
6882 040520  242                       .BYTE   ^C135           ;BIT  5
6883 040521  343                       .BYTE   ^C034           ;BIT  4
6884 040522  244                       .BYTE   ^C133           ;BIT  3
6885 040523  345                       .BYTE   ^C032           ;BIT  2
6886 040524  346                       .BYTE   ^C031           ;BIT  1
6887 040525  247                       .BYTE   ^C130           ;BIT  0
```

```
                                        SUBTST<<SUBR    MAPPER>>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    MAPPER
                                ;*****************************************************************************
                                ;
                                ;THIS SUBROUTINE MAPS THE MEMORY BANK (16K WORDS = 1 BANK)
                                ;IN R3 TO THE TEST PATTERN AREA (SUPERVISOR VIRTUAL (60000 - 157777) FOR
                                ;THE 11/44 AND 11/45-55; USER VIRTUAL (60000 - 157777) FOR ALL OTHER
                                ;PDP-11'S).
                                ;
                                ;CALL    MOV     BANKNO,R3               ;SET UP BANK ARGUEMENT
                                ;       CALL    MAPPER                  ;ACTUAL CALL
                                ;       RETURN                          ;ONLY RETURN

                                        ;SET SUPERVISOR/USER UP FOR 1 TO 1 MAP
040526                          MAPPER: PUSH    R0,R1,R2,R4,R5
040540  012700  172340                  MOV     #KIPAR0,R0              ;FIRST AREA TO MAP TO
040544  012701  172240                  MOV     #SIPAR0,R1              ;FIRST ADDRESS REGISTER
040550  012704  172200                  MOV     #SIPDR0,R4              ;FIRST DESCRIPTOR REGISTER
040554  005737  002426                  TST     NOSUPER                 ;CAN WE USE SUPERVISOR MODE?
040560  001404                          BEQ     4$                      ;YES, BRANCH
040562  012701  177640                  MOV     #UIPAR0,R1              ;FIRST ADDRESS REGISTER
040566  012704  177600                  MOV     #UIPDR0,R4              ;FIRST DESCRIPTOR REGISTER
040572  012702  077406          4$:     MOV     #77406,R2               ;CONSTANT FOR 4K PAGE, UP, R/W
040576  012705  000010                  MOV     #8.,R5                  ;COUNTER
040602  012021                  1$:     MOV     (R0)+,(R1)+             ;PUT IN SUPERVISOR ADDRESS
040604  010224                          MOV     R2,(R4)+                ;PUT IN SUPERVISOR DESCRIPTOR
040606  077503                          SOB     R5,1$                   ;LOOP TILL DONE
040610  012741  177600                  MOV     #177600,-(R1)           ;CORRECT LAST FIELD FOR PERIPHERALS PAGE

                                        ;SET UP SUPERVISOR/USER FOR TEST AREA
040614  022703  000170                  CMP     #120.,R3                ;MAP NOTHING (1 TO 1)?
040620  001512                          BEQ     3$                      ;YES - SKIP
040622  072327  000011                  ASH     #9.,R3                  ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
                                                                        ;FOR MEMORY MANAGEMENT = 1000
040626  012701  172246                  MOV     #SIPAR3,R1              ;SETUP FOR AUTO INCREMENTING
040632  005737  002426                  TST     NOSUPER                 ;DO WE HAVE SUPERVISOR MODE?
040636  001402                          BEQ     5$                      ;YES - BRANCH
040640  012701  177646                  MOV     #UIPAR3,R1              ;SETUP FOR AUTO INCREMENTING
040644  012702  000004          5$:     MOV     #4,R2                   ;COUNTER
040650  010321                  2$:     MOV     R3,(R1)+                ;PLUG IN PAR INFO
040652  062703  000200                  ADD     #200,R3                 ;BUMP ADDRESS 4K
040656  077204                          SOB     R2,2$                   ;LOOP TILL DONE
040660  005737  002232                  TST     SPLTCSR
040664  001442                          BEQ     9$
040666  162701  000010                  SUB     #10,R1
040672  010102                          MOV     R1,R2
040674  062702  000004                  ADD     #4,R2
040700  022737  000001  002232          CMP     #1,SPLTCSR
040706  001403                          BEQ     10$
040710  010200                          MOV     R2,R0
040712  010102                          MOV     R1,R2
040714  010001                          MOV     R0,R1
040716  012122                  10$:    MOV     (R1)+,(R2)+
040720  011112                          MOV     (R1),(R2)
040722  013700  002102                  MOV     BANKINDEX,R0
040726  005737  002136                  TST     INT64K
040732  001403                          BEQ     11$
```

```
 6944 040734  012700  004000                    MOV     #4000,R0
 6945 040740  000402                            BR      12$
 6946 040742  012700  010000          11$:      MOV     #10000,R0
 6947 040746  005737  002426          12$:      TST     NOSUPER
 6948 040752  001403                            BEQ     13$
 6949 040754  012701  177652                    MOV     #UIPAR5,R1
 6950 040760  000402                            BR      14$
 6951 040762  012701  172252          13$:      MOV     #SIPAR5,R1
 6952 040766  060021                  14$:      ADD     R0,(R1)+
 6953 040770  060011                            ADD     R0,(R1)
 6954                                           ;IF WE ONLY HAVE AN 124K SYSTEM, WE DON'T WANT TO TEST THE
 6955                                           ;LAST 4K, WHERE THE UNIBUS DEVICE PAGE IS. INSTEAD, THE
 6956                                           ;PROGRAM WILL REMAP THE LAST 4K TO 8-12K. ALSO, IF THERE
 6957                                           ;IS A BANK 177 ON AN 11/44, THE PROGRAM WILL REMAP THE LAST
 6958                                           ;4K TO 8-12K FOR THE SAME REASON.
 6959 040772  022737  000007  002526  9$:       CMP     #7,LASTBANK
 6960 041000  001004                            BNE     7$
 6961 041002  022703  010000                    CMP     #10000,R3
 6962 041006  001017                            BNE     3$
 6963 041010  000404                            BR      8$
 6964 041012  022737  000177  002526  7$:       CMP     #177,LASTBANK
 6965 041020  001012                            BNE     3$
 6966 041022  005737  002426          8$:       TST     NOSUPER
 6967 041026  001404                            BEQ     6$
 6968 041030  013737  177652  177654            MOV     UIPAR5,UIPAR6
 6969 041036  000403                            BR      3$
 6970 041040  013737  172252  172254  6$:       MOV     SIPAR5,SIPAR6
 6971 041046                          3$:       POP     R5,R4,R2,R1,R0
 6972 041060  000207                            RETURN
 6973                                           .SBTTL  TRAP    MAP KERNEL (ALMOST 1 TO 1) TRAP HANDLER
 6974 041062                          $KMAP:    PUSH    R0,R1,R2,R3,R4
 6975 041074  005000                            CLR     R0                      ;1ST AREA TO MAP TO
 6976 041076  012701  172340                    MOV     #KIPAR0,R1              ;FIRST ADDRESS
 6977 041102  012702  077406                    MOV     #77406,R2               ;CONSTANT FOR 4K PAGE,UP,R/W
 6978 041106  012703  172300                    MOV     #KIPDR0,R3              ;1ST PAGE DESCRIPTOR REGISTER
 6979 041112  012704  000010                    MOV     #8.,R4                  ;COUNTER
 6980 041116  010021                  1$:       MOV     R0,(R1)+                ;PUT IN KERNEL ADDRESS
 6981 041120  010223                            MOV     R2,(R3)+                ;PUT IN KERNEL DISCRIPTOR
 6982 041122  062700  000200                    ADD     #200,R0                 ;ADD ADDRESS CONSTANT FOR 4K CHANGE
 6983 041126  077405                            SOB     R4,1$                   ;LOOP TILL DONE
 6984 041130  012741  177600                    MOV     #177600,-(R1)           ;THE PERIPHERALS PAGE TO KIPAR7
 6985 041134  012741  177400                    MOV     #177400,-(R1)           ;AND NEXT LOWER PAGE TO KIPAR6
 6992 041140                                    POP     R4,R3,R2,R1,R0
 6993 041152  000002                            RTI
```

```
 6996 041154                                    RELOCATE:SUBTST <<RELOCATE PROGRAM>>
                                                ;*****************************************************************************
                                                ;*SUBTEST       RELOCATE PROGRAM
                                                ;*****************************************************************************
 6997 041154                                            IF #SW12 SET.IN @SWR THEN $RETURN ERROR
 6998 041170                                            IF APTFLAG IS TRUE OR ACTFLAG IS TRUE
 6999 041204                                              IF $PASS NE #0 THEN $RETURN ERROR
 7000 041216                                            END; OF IF APTFLAG
 7001 041216                                            BEGIN LOADERBANK
 7002 041216                                              FOR BANK := #1 TO LASTBANK
 7003 041224  004737  042700                                CALL EXBANK
 7004 041230                                                IF ACFLAG IS TRUE AND PFLAG IS FALSE AND BMFLAG IS FALSE
 7005 041252  013700  002100                                  MOV       BANK,R0
 7006 041256  010037  002402                                  MOV       R0,LOADBANK
 7007 041262  013701  002536                                  MOV       LOADHOME,R1
 7008 041266  004737  042350                                  CALL      BANKMOV
 7009 041272  004737  042646                                  CALL      NEWLOAD          ;MAP NEW LOADER BANK IN KERNEL
 7010 041276  013701  002102                                  MOV       BANKINDEX,R1
 7011 041302  052761  100000  002626                          BIS       #BIT15,CONFIG+2(R1)      ;MARK LOADER
 7012 041310  042761  020000  002626                          BIC       #BIT13,CONFIG+2(R1)      ;INVALIDATE BACKGROUND PATTERN
 7013 041316                                                  LEAVE LOADERBANK
 7014 041320                                                END ;OF IF ACFLAG
 7015 041320                                              END ;OF FOR BANK
 7016 041334                                              IF #SW13 OFF.IN @SWR
 7017 041344                                                TYPE        MSG075           ;RELOCATION NOT POSSIBLE
 7018 041350                                              END ;OF IF #SW13
 7019 041350                                              $RETURN ERROR
 7020 041354                                            END LOADERBANK
 7021 041354                                            BEGIN FINDBANK
 7022 041354  013702  002526                            MOV     LASTBANK,R2
 7023 041360  006302                                    ASL     R2
 7024 041362  006302                                    ASL     R2               ;R2 <- R2 * 4
 7025 041364                                              FOR R1 := #2*2 TO R2 BY #4
 7026 041370                                                IF #BIT7!BIT0 OFF.IN CONFIG(R1) ;IF NO ERRORS & NOT PROGRAM SPACE
 7027 041400                                                  IF #BIT15 OFF.IN CONFIG+2(R1) ;IF NOT LOADER BANK
 7028 041410                                                    IF CPUBIT SET.IN CONFIG(R1) ;IF ACCESSABLE
 7029 041420                                                      IF #BIT9 SET.IN CONFIG+2(R1) THEN LEAVE FINDBANK ;IF PARITY
 7030 041430                                                        IF #BIT6 SET.IN CONFIG(R1) AND #BIT7 OFF.IN CONFIG(R1)
 7031                                                                ;IF 1ST PROTECTABLE ECC BANK
 7032 041450                                                        LEAVE FINDBANK
 7033 041452                                                      END ;OF IF #BIT6
 7034 041452                                                      IF INHECC IS FALSE
 7035 041460                                                        SET   INHECC
 7036 041466  010137  002510                                      MOV   R1,INHBANK
 7037 041472                                                      END; OF IF INHECC
 7038 041472                                                    END ;OF IF CPUBIT
 7039 041472                                                  END ;OF IF #BIT15
 7040 041472                                                END ;OF IF #BIT7
 7041 041472                                              END ;OF FOR
 7042 041502                                              IF FULLREL IS FALSE
 7043 041510                                                IF INHECC IS TRUE
 7044 041516  013701  002510                                  MOV     INHBANK,R1
 7045 041522  023727  002260  000030                          CMP     REALPAT,#30      ;IS THIS PATTERN 30?
 7046 041530  001423                                          BEQ     RELENT1          ;YES - SKIP MESSAGE
 7047 C-1532                                                  TYPE    MSG123
 7048 041536  000420                                          BR      RELENT1
 7049 041540                                                END; OF IF INHECC
```

```
7050 041540                                    END; OF IF FULLREL
7051 041540  005037  002506                    CLR   INHECC                    ;MAKE SURE FLAG IS TURNED OFF!
7052 041544                                    IF #SW13 OFF.IN @SWR
7053 041554  023727  002260  000030              CMP         REALPAT,#30       ;IS THIS PATTERN 30?
7054 041562  001402                              BEQ         SKUB              ;YES - SKIP MESSAGE
7055 041564                                      TYPE        MSG075            ;RELOCATION NOT POSSIBLE
7056 041570                                    END ;OF IF #SW13
7057 041570                            SKUB:   $RETURN ERROR
7058 041574                                  END FINDBANK
7059 041574                                  CLEAR   INHECC                    ;IF WE RELOCATED PROPERLY, THIS SHOULD BE OFF!
7060 041600  042761  020000  002626  RELENT1: BIC    #BIT13,CONFIG+2(R1)       ;INVALIDATE BACKGROUND PATTERN
7061 041606  005000                          CLR     R0
7062 041610  071027  000004                  DIV     #4,R0
7063 041614                          RELOC1: LET NEWBANK := R0
7064 041620  013737  002502  002504          MOV     PGMCSR,PGMCSR+2           ;SAVE CURRENT PGM. CSR
7065 041626  004737  042516                  CALL    USERMAP                   ;MAP NEWBANK TO USER PAR
7066 041632                                  USER                              ;ENTER USER MODE
7067 041640                                  BMOV    0,100000,SIZE             ;MOVE PROGRAM
7068 041652  104417                          KERNEL                            ;ENTER KERNEL MODE
7069 041654  005737  002424                  TST     NO22BIT
7070 041660  001021                          BNE     JMPRL1
7071 041662  042737  000040  172516          BIC     #BIT5,MMR3                ;TURN OFF UNIBUS MAP
7072 041670  013700  002270                  MOV     NEWBANK,R0
7073 041674  006200                          ASR     R0
7074 041676                                  ON.ERROR
7075 041700  012737  100000  170200            MOV   #BIT15,MAPL0
7076 041706                                  END ;OF ON.ERROR
7077 041706  010037  170202                  MOV     R0,MAPH0
7078 041712  004737  042304                  CALL    LOWMAP                    ;SETUP LOWER 16K IN UNIBUS MAP
7079 041716  052737  000040  172516          BIS     #BIT5,MMR3                ;ENERGIZE UNIBUS MAP
7080 041724  042737  000001  177572  JMPRL1: BIC     #BIT0,MMR0                ;DEENERGIZE MEMORY MANAGEMENT
7081 041732  004737  042600                  CALL    NEWKERNEL
7082 041736  013700  002270                  MOV     NEWBANK,R0
7083 041742  006300                          ASL     R0
7084 041744  006300                          ASL     R0                        ;R0 <- R0 * 4
7085 041746  016002  002624                  MOV     CONFIG(R0),R2
7086 041752  000302                          SWAB    R2
7087 041754  042702  177760                  BIC     #^C17 R2
7088 041760  006302                          ASL     R2
7089 041762  052737  000001  177572          BIS     #BIT0,MMR0                ;ENERGIZE MEMORY MANAGEMENT
7090 041770  010237  002502                  MOV     R2,PGMCSR                 ;PUT NEW PGM. CSR INTO PGMCSR
7091 041774  032760  010000  002626          BIT     #BI  2,CONFIG+2(R0)       ;IS THE NEW BANK INTERLEAVED?
7092 042002  001412                          BEQ     1$                        ;BRANCH IF NOT INTERLEAVED
7093 042004  016002  002624                  MOV     CONFIG(R0),R2
7094 042010  042702  007777                  BIC     #^C170000,R2
7095 042014  072227  177775                  ASH     #-3,R2
7096 042020  052702  100000                  BIS     #BIT15,R2
7097 042024  050237  002502                  BIS     R2,PGMCSR
7098 042030                            1$:   SET     RLFLAG
7099 042036                                  $RETURN NOERROR
```

```
7102 042042                                 UNRELOCATE:SUBTST       <<UNRELOCATE PROGRAM>>
                                            ;*****************************************************************************
                                            ;*SUBTEST       UNRELOCATE PROGRAM
                                            ;*****************************************************************************
7103                                                ;RESTORE LOADERS
7104 042042                                         PUSH    R0
7105 042044  013701  002402                         MOV     LOADBANK,R1
7106 042050  013700  002536                         MOV     LOADHOME,R0
7107 042054  004737  042350                         CALL    BANKMOV
7108 042060  004737  042646                         CALL    NEWLOAD                 ;MAP NEW LOADER BANK IN KERNEL SPACE
7109 042064                                         PUSH    BANK
7110 042070  013737  002402  002100                 MOV     LOADBANK,BANK
7111 042076  004737  042700                         CALL    EXBANK
7112 042102  013701  002102                         MOV     BANKINDEX,R1
7113 042106  042761  100000  002626                 BIC     #BIT15,CONFIG+2(R1)     ;CLEAR LOADER FLAG
7114 042114  013737  002536  002100                 MOV     LOADHOME,BANK
7115 042122  004737  042700                         CALL    EXBANK
7116 042126  013701  002102                         MOV     BANKINDEX,R1
7117 042132  042761  020000  002626                 BIC     #BIT13,CONFIG+2(R1)     ;INVALIDATE BACKGROUND PATTERN
7118 042140                                         POP     BANK
7119 042144                                         CLEAR   INHECC                  ;MAKE SURE ECC TESTS ARE NOT INHIBITED!
7120
7121                                                ;RESTORE BANK 0
7122 042150  042737  020000  002626                 BIC     #BIT13,CONFIG+2         ;INVALIDATE BACKGROUND PATTERN
7123 042156                                         LET NEWBANK := #0
7124 042162  004737  042516                         CALL    USERMAP                 ;MAP NEWBANK TO USER PAR
7125 042166                                         USER                            ;ENTER USER MODE
7126 042174                                         BMOV    0,100000,SIZE           ;MOVE PROGRAM
7127 042206  104417                                 KERNEL                          ;ENTER KERNEL MODE
7128 042210  042737  000001  177572                 BIC     #BIT0,MMR0              ;DEENERGIZE MEMORY MANAGEMENT
7129 042216  004737  042600                         CALL    NEWKERNEL
7130 042222  013737  002504  002502                 MOV     PGMCSR+2,PGMCSR         ;RESTORE PREVIOUS PGM. CSR
7131 042230  052737  000001  177572                 BIS     #BIT0,MMR0              ;ENERGIZE MEMORY MANAGEMENT
7132 042236  005037  002124                         CLR     RLFLAG
7133 042242  005737  002424                         TST     NO22BIT
7134 042246  001014                                 BNE     1$
7135 042250  042737  000040  172516                 BIC     #BIT5,MMR3              ;TURN OFF UNIBUS MAP
7136 042256                                         CLEAR   MAPL0,MAPH0
7137 042266  004737  042304                         CALL    LOWMAP                  ;SETUP LOWER 16K OF UNIBUS MAP
7138 042272  052737  000040  172516                 BIS     #BIT5,MMR3              ;ENERGIZE UNIBUS MAP
7139 042300                                 1$:     POP     R0
7140 042302  000207                                 RETURN
7141
7142 042304                                 LOWMAP: SUBTST  <<SETUP LOWER 16K OF UNIBUS MAP>>
                                            ;*****************************************************************************
                                            ;*SUBTEST       SETUP LOWER 16K OF UNIBUS MAP
                                            ;*****************************************************************************
7143 042304                                         PUSH    R0,R1,R2
7144 042312  012700  170200                         MOV     #MAPL0,R0
7145 042316  012701  170204                         MOV     #MAPL1,R1
7146 042322  012702  000003                         MOV     #3,R2
7147 042326  012011                         1$:     MOV     (R0)+,(R1)
7148 042330  062721  020000                         ADD     #BIT13,(R1)+
7149 042334  012021                                 MOV     (R0)+,(R1)+
7150 042336  077205                                 SOB     R2,1$
7151 042340                                         POP     R2,R1,R0
7152 042346  000207                                 RETURN
```

```
7155 042350                          BANKMOV:SUBTST  <<MOVE BANKS>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        MOVE BANKS
                                     ;*****************************************************************************
7156                                         ;MOVE 3/4 OF A BANK
7157                                         ;CALLING SEQUENCE
7158                                         ;R0 = DESTINATION BANK
7159                                         ;R1 = SOURCE BANK
7160 042350  104415                         SAVREG
7161 042352  004737  042516                 CALL    USERMAP
7162 042356  104416                         RESREG
7163 042360  104415                         SAVREG
7164 042362  072027  000011                 ASH     #9.,R0
7165 042366  072127  000011                 ASH     #9.,R1
7166 042372  012702  177650                 MOV     #UIPAR4,R2
7167 042376  012703  000200                 MOV     #200,R3
7168
7169 042402  010122                        MOV     R1,(R2)+                ;MAP 1ST HALF BANK
7170 042404  060301                        ADD     R3,R1                   ;BUMP BY 4K
7171 042406  010122                        MOV     R1,(R2)+
7172 042410  060301                        ADD     R3,R1
7173
7174 042412  010022                        MOV     R0,(R2)+
7175 042414  060300                        ADD     R3,R0
7176 042416  010022                        MOV     R0,(R2)+
7177 042420  060300                        ADD     R3,R0
7178
7179 042422                                 USER
7180 042430                                 BMOV    100000,140000,SIZE/2    ;MOV 1ST HALF BANK
7181 042442  104417                         KERNEL                          ;ENTER KERNEL MODE
7182
7183 042444  012702  177650                 MOV     #UIPAR4,R2
7184
7185 042450  010122                        MOV     R1,(R2)+                ;MAP 2ND HALF BANK
7186 042452  060301                        ADD     R3,R1                   ;BUMP BY 4K
7187 042454  010122                        MOV     R1,(R2)+
7188 042456  060301                        ADD     R3,R1
7189
7190 042460  010022                        MOV     R0,(R2)+
7191 042462  060300                        ADD     R3,R0
7192 042464  010022                        MOV     R0,(R2)+
7193 042466  060300                        ADD     R3,R0
7194
7195 042470                                 USER
7196 042476                                 BMOV    100000,140000,SIZE/4    ;MOV 3ND FOURTH OF BANK
7197 042510  104417                         KERNEL                          ;ENTER KERNEL MODE
7198
7199 042512  104416                         RESREG
7200 042514  000207                         RETURN
```

```
7203 042516                          USERMAP:SUBTST  <<SUBR  MAP USER TO NEW BANK>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        SUBR    MAP USER TO NEW BANK
                                     ;*****************************************************************************
7204 042516  012701  177640                 MOV     #UIPAR0,R1              ;COPY KERNEL PAR'S & PDR'S (0-3)
7205 042522  012702  172340                 MOV     #KIPAR0,R2
7206 042526  012703  177600                 MOV     #UIPDR0,R3
7207 042532  012704  172300                 MOV     #KIPDR0,R4
7208 042536  012705  000004                 MOV     #4,R5
7209 042542  012221                 1$:     MOV     (R2)+,(R1)+
7210 042544  011423                         MOV     (R4),(R3)+
7211 042546  077503                         SOB     R5,1$
7212
7213 042550  013700  002270                 MOV     NEWBANK,R0
7214 042554  072027  000011                 ASH     #9.,R0                  ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
7215                                                                         ;FOR MEMORY MANAGEMENT = 1000
7216 042560  012705  000004                 MOV     #4,R5
7217 042564  010021                 2$:     MOV     R0,(R1)+                ;SETUP UIPAR(4-7)
7218 042566  062700  000200                 ADD     #200,R0                 ;BUMP ADDRESS 4K
7219 042572  011423                         MOV     (R4),(R3)+              ;SETUP UIPDR(4-7)
7220 042574  077505                         SOB     R5,2$
7221 042576  000207                         RETURN
7222
7223 042600                          NEWKERNEL:SUBTST        <<SUBR  SETUP KERNEL PAR'S FOR NEW BANK>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        SUBR    SETUP KERNEL PAR'S FOR NEW BANK
                                     ;*****************************************************************************
7224 042600                                 PUSH    R0,R1,R5
7225 042606  012700  172340                 MOV     #KIPAR0,R0
7226 042612  013701  002270                 MOV     NEWBANK,R1
7227 042616  072127  000011                 ASH     #9.,R1                  ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
7228                                                                         ;FOR MEMORY MANAGEMENT = 1000
7229 042622  012705  000004                 MOV     #4,R5
7230 042626  010120                 1$:     MOV     R1,(R0)+                ;SETUP KIPAR(0-3)
7231 042630  062701  000200                 ADD     #200,R1
7232 042634  077504                         SOB     R5,1$
7233 042636                                 POP     R5,R1,R0
7234 042644  000207                         RETURN
7235
7236 042646                          NEWLOAD:SUBTST  <<SUBR  SETUP KERNEL PAR'S FOR NEW LOADER BANK>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        SUBR    SETUP KERNEL PAR'S FOR NEW LOADER BANK
                                     ;*****************************************************************************
7237                                        ;R0 CONTAINS THE DESTINATION BANK
7238 042646                                 PUSH    R0,R1
7239 042652  012701  172350                 MOV     #KIPAR4,R1
7240 042656  072027  000011                 ASH     #9.,R0                  ;BANK 1 STARTS AT 100000 LESS 6 LSB'S (1000)
7241 042662  010021                         MOV     R0,(R1)+                ;SETUP KIPAR4
7242 042664  062700  000200                 ADD     #200,R0
7243 042670  010021                         MOV     R0,(R1)+                ;SETUP KIPAR5
7244 042672                                 POP     R1,R0
7245 042676  000207                         RETURN
```

```
                                EXBANK: SUBTST  <<SUBR  EXAMINE BANK>>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    EXAMINE BANK
                                ;*****************************************************************************
                                ;DOES THE FOLLOWING:
                                ;(1)  SETS UP 'BANKINDEX'' AND R1 BASED ON VALUE OF 'BANK''.
                                ;(2)  SETS THE 'MKFLAG'' IF THE BANK IS ECC.
                                ;(3)  SETS THE 'KFLAG'' IF THE BANK IS MF11S-K.
                                ;(4)  SETS THE 'KPFLAG'' IF THE BANK IS THE PROTECTED REGION OF ECC MEMORY.
                                ;(5)  SETS THE 'ACFLAG'' IF THE BANK CAN BE ACCESSED BY THIS CPU.
                                ;(6)  SETS THE 'PFLAG'' IF THE BANK IS IN PROGRAM SPACE.
                                ;(7)  SETS THE  'RRFLAG'' IF RELOCATION IS REQUIRED TO TEST THIS BANK; HOWEVER,
                                ;     IT COMPLEMENTS THIS FLAG IF THE RELOCATION FLAG 'RLFLAG'' IS SET (THIS IS
                                ;     NECESSARY FOR THE USE OF THE RECURSIVE 'MODE'' SUBROUTINES). THE 'RRFLAG''
                                ;     IS ALWAYS SET TO DISABLE TESTING IF FIELD SERVICE MODE  'SELECTED BANKS''
                                ;     ARE BEING TESTED AND THIS BANK IS NOT SELECTED.
                                ;(8)  SETS THE 'BMFLAG'' IF THE BANK IS A BAD MEMORY;  HOWEVER, IT  COMPLEMENTS
                                ;     THIS FLAG IF THE 'WORST'' FLAG IS NOT SET  (THIS IS NECESSARY FOR THE USE
                                ;     OF THE RECURSIVE 'MODE'' SUBROUTINES).
                                ;(9)  SETS THE 'EUFLAG'' IF THE BANK HAS EXTENDED UNIBUS MEMORY.
                                ;(10) SETS THE ''INTFLAG'' IF THE BANK IS INTERLEAVED.
                                ;(11) SETS THE ''INT64K'' FLAG IF THE BANK IS INTERLEAVED ON 64K WORD BOUNDS.
                                ;(12) SETS THE 'SKIPMK'' FLAG IF THIS BANK IS INTERLEAVED, AND HAS ALREADY
                                ;     BEEN TESTED.
                                ;
 042700                           PUSH    R0,R1,R2
 042706                           CLEAR   MKFLAG,KPFLAG,KFLAG,EUFLAG
 042726                           SET     ACFLAG
 042734                           CLEAR   PFLAG,RRFLAG,BMFLAG
 042750                           CLEAR   INTFLAG,INT64K,SKIPMK
 042764  013701  002100           MOV     BANK,R1
 042770  006301                   ASL     R1
 042772  006301                   ASL     R1                  ;R1 <- R1 * 4
 042774  010137  002102           MOV     R1,BANKINDEX
 043000  032761  000100  002624   BIT     #BIT6,CONFIG(R1)    ;PROTECTED REGION OF ECC MEMORY?
 043006  001403                   BEQ     1$                  ;NO - SKIP
 043010                           SET     KPFLAG
 043016  012700  000002    1$:    MOV     #BIT1,R0
                                IF R0 SET.IN CPUBIT AND R0 OFF.IN CONFIG(R1)
 043022                           CLR     ACFLAG
                                END ;OF IF R0
 043042  005737  002114           TST     ACFLAG              ;ACTIVE MEMORY?
 043046  001415                   BEQ     12$                 ;BRANCH IF NOT
 043050  016102  002626           MOV     CONFIG+2(R1),R2
 043054  000302                   SWAB    R2
 043056  042702  177770           BIC     #^C7,R2             ;ISOLATE MEM TYPE BITS
 043062  020227  000002           CMP     R2,#2               ;IS THIS AN ILLEGAL MEM TYPE?
 043066  003005                   BGT     12$                 ;BRANCH IF NOT
 043070                           SET     BMFLAG              ;SET BAD BANK FLAG
 043076  000137  043340           JMP     ENEXBK              ;JUMP OVER REST OF FLAG TESTS
 043102  032761  000400  002626   BIT     #BIT8,CONFIG+2(R1)  ;IS THIS EUB?
 043110  001003                   BNE     2$                  ;BRANCH IF NOT
 043112                           SET     EUFLAG              ;YES - SET EUB FLAG
 043120  032761  001000  002626   BIT     #BIT9,CONFIG+2(R1)  ;IS THERE ECC THERE?
 043126  001012                   BNE     3$                  ;NO - SKIP
 043130                           SET     MKFLAG              ;YES - SET MKFLAG
 043136  032761  000400  002626   BIT     #BIT8,CONFIG+2(R1)  ;IS THIS MF11S-K MEMORY
```

```
 7309 043144  001403                          BEQ     3$                      ;NO - IT'S MS11-M
 7310 043146                                   SET     KFLAG                   ;YES - SET KFLAG
 7311 043154  032761  000200  002624  3$:     BIT     #BIT7,CONFIG(R1)        ;BANK = PROGRAM SPACE?
 7312 043162  001406                          BEQ     5$                      ;NO - SKIP
 7313 043164                                   SET     PFLAG,RRFLAG
 7314 043200  005737  002124          5$:     TST     RLFLAG                  ;IS PROGRAM RELOCATED?
 7315 043204  001402                          BEQ     6$                      ;NO - SKIP
 7316 043206  005137  002122                  COM     RRFLAG                  ;YES - COMPLEMENT RELOCATION REQUIRED FLAG
 7317 043212  032761  000001  002624  6$:     BIT     #BIT0,CONFIG(R1)        ;ERRORS PRESENT IN THIS BANK?
 7318 043220  001403                          BEQ     8$                      ;NO - SKIP
 7319 043222                                   SET     BMFLAG
 7320 043230  005737  002540          8$:     TST     WORST                   ;IS THIS A WORST FIRST PASS?
 7321 043234  001002                          BNE     9$                      ;YES - SKIP
 7322 043236  005137  002126                  COM     BMFLAG                  ;NO - COMPLEMENT BAD MEMORY FLAG
 7323 043242                          9$:     IF SELONLY IS TRUE AND #BIT14 OFF.IN CONFIG+2(R1)
 7324 043260                                    SET   RRFLAG
 7325 043266                                   END ;OF  IF SELONLY
 7326 043266  032761  010000  002626          BIT     #BIT12,CONFIG+2(R1)     ;IS THIS BANK INTERLEAVED?
 7327 043274  001421                          BEQ     ENEXBK                  ;BRANCH IF IT IS NOT
 7328 043276                                   SET     INTFLAG
 7329 043304  032761  004000  002626          BIT     #BIT11,CONFIG+2(R1)     ;IS THIS BANK INTERLEAVED WITH 64K BOARDS?
 7330 043312  001403                          BEQ     10$                     ;BRANCH IF IT IS NOT
 7331 043314                                   SET     INT64K
 7332 043322  032761  000040  002624  10$:    BIT     #BIT5,CONFIG(R1)        ;SHOULD THIS BANK BE TESTED?
 7333 043330  001403                          BEQ     ENEXBK                  ;BRANCH IF IT SHOULD
 7334 043332                                   SET     SKIPMK
 7335 043340                          ENEXBK: POP     R2,R1,R0                ;RESTORE REGISTERS
 7336 043346  000207                          RETURN
```

```
                                BANKOK: SUBTST  <<SUBR  BANK OK?>>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    BANK OK?
                                ;*****************************************************************************
                                        ;TEST TO INSURE THAT THE TYPE OF MEMORY IN THE PRESENT BANK
                                        ;IS OF THE TYPE WE ARE TESTING 'TMFLAG'.
                                        ;RESULT IS RETURNED IN THE CONDITION CODES (OK = (=0)).
043350  013700  002132                  MOV     TMFLAG,R0
043354  005100                          COM     R0
043356  013701  002116                  MOV     MKFLAG,R1
043362  074001                          XOR     R0,R1
043364  000207                          RETURN                          ;OK = (=OK)

043366                          INCRPT:
043366                          INCPAT: SUBTST  <<SUBR  INCREMENT PATTERN TESTING >>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    INCREMENT PATTERN TESTING
                                ;*****************************************************************************
                                        ;INCREMENT THE PATTERN & SET UP THE CONDITION CODES
                                        ;RESULT - Z BIT SET INDICATES OVERFLOW
043366  005237  002110                  INC     PATTERN
043372  022737  000030  002110          CMP     #30,PATTERN             ;SET UP CONDITION CODES
043400  000207                          RETURN                          ;NOT EQUAL TO ZERO IS GOOD (NO OVERFLOW)

043402                          SETPAT:
043402                          HIPAT:  SUBTST  <<SUBR  SET HIGHEST PATTERN TESTING TYPE>>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    SET HIGHEST PATTERN TESTING TYPE
                                ;*****************************************************************************
043402  012737  000027  002110          MOV     #27,PATTERN             ;SET HIGHEST PATTERN
043410  000207                          RETURN

043412                          INCBNK: SUBTST  <<SUBR  INCREMENT BANK & TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    INCREMENT BANK & TEST
                                ;*****************************************************************************
                                        ;RESULTS RETURNED IN CONDITION CODES
043412  005237  002100                  INC     BANK
043416  023737  002526  002100          CMP     LASTBANK,BANK           ;TOO FAR?
043424  000207                          RETURN
```

```
7369 043426                          BOOT:   SUBTST  <<BOOTSTRAP ROUTINE>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       BOOTSTRAP ROUTINE
                                     ;*****************************************************************************
7370                                         ;INITIALIZE ALL CSR'S
7371                                         ;UNRELOCATE IF NECESSARY
7372                                         ;FLUSH OUT ANY DBE'S
7373                                         ;TURN OFF MEMORY MANAGEMENT
7374                                         ;TURN OFF THE UNIBUS MAP
7375                                         ;BOOT RK0 OR RK1
7376 043426  104472                         ECCINIT          ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
7377 043430                                  SET4    #BOOT1           ;TRAPS TO 4 GOTO BOOT1
7378 043436                                  IF RLFLAG IS TRUE THEN $CALL UNRELOCATE
7379 043450  004737  023412                  CALL    MT0030           ;FLUSH OUT DBE'S
7380 043454  104421                         DEENERGIZE               ;TURN OFF MEMORY MANAGEMENT
7381 043456  005737  002424                  TST     NO22BIT          ;IS THIS AN 11/44?
7382 043462  001003                         BNE     BOOT1
7383 043464  042737  000040  172516         BIC     #BIT5,MMR3                    ;TURN OFF THE UNIBUS MAP
7384 043472  005001                 BOOT1:  CLR     R1
7385 043474  000005                 1$:     RESET
7386 043476  012700  177406                 MOV     #177406,R0
7387 043502  010160  000004                 MOV     R1,4(R0)
7388 043506  012710  177400                 MOV     #177400,(R0)
7389 043512  012740  000005                 MOV     #5,-(R0)
7390 043516  105710                 2$:     TSTB    (R0)
7391 043520  100376                         BPL     2$
7392 043522  062701  020000                 ADD     #BIT13,R1
7393 043526  005710                         TST     (R0)
7394 043530  100761                         BMI     1$
7395 043532  005007                         CLR     PC
```

```
7398 043534                          EXIT:   SUBTST  <<HALT PROGRAM>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       HALT PROGRAM
                                     ;*****************************************************************************
7399 043534  004737  043566                 CALL    SHUTUP
7400 043540                          EXIT2:  IF APTFLAG IS TRUE OR ACTFLAG IS TRUE
7401 043554  000777                           BR      .
7402 043556                                  ELSE
7403 043560  000000                  $EXHALT:  HALT
7404 043562  000137  003630                   JMP     START
7405 043566                                  END ;OF IF APTFLAG
7406
7407 043566                          SHUTUP: SUBTST  <<SHUTDOWN DIAGNOSTIC>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       SHUTDOWN DIAGNOSTIC
                                     ;*****************************************************************************
7408                                         ;INITIALIZE ALL CSR'S
7409                                         ;UNRELOCATE
7410                                         ;FLUSH OUT DBE'S
7411                                         ;RESTORE LOADERS
7412                                         ;TURN OFF MEMORY MANAGEMENT
7413                                         ;UNMAP THE UNIBUS MAP
7417 043566  104472                         ECCINIT         ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
7418 043570                                  IF RLFLAG IS TRUE THEN $CALL UNRELOCATE
7419 043602                                  IF QUICK IS FALSE
7420 043610  004737  023412                   CALL  MT0030          ;FLUSH OUT DBE'S
7421 043614                                  END ;OF IF QUICK
7422 043614  012700  000001                 MOV     #1,R0           ;DESTINATION BANK
7423 043620  013701  002536                 MOV     LOADHOME,R1     ;SOURCE BANK
7424 043624  004737  042350                 CALL    BANKMOV
7425 043630  104421                         DEENERGIZE              ;TURN OFF MEMORY MANAGEMENT
7426 043632  005737  002424                 TST     NO22BIT         ;DOES THIS PDP-11 HAVE 22-BIT ADDR?
7427 043636  001003                         BNE     1$              ;BRANCH IF NOT
7428 043640  042737  000040  172516         BIC     #BIT5,MMR3              ;TURN OFF UNIBUS MAP
7432 043646  000207                 1$:     RETURN
7433
7434 043650                          APTDOWN:SUBTST  <<APT SHUTDOWN SEQUENCE>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       APT SHUTDOWN SEQUENCE
                                     ;*****************************************************************************
7435 043650                                  MAP     #0                      ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
7436 043664                                  TESTAREA                ;ENTER TEST MODE
7437 043672  012737  043650  060024         MOV     #APTDOWN,FIRST+24
7438 043700  012737  000340  060026         MOV     #340,FIRST+26
7439 043706  012737  000000  123650         MOV     #0,FIRST+APTDOWN
7440 043714  104417                         KERNEL                  ;ENTER KERNEL MODE
7441 043716  000000                 APTHLT: HALT
```

```
                                        SUBTST  <<BLOCK MOVE SUBROUTINE>>
                                ;*******************************************************************************
                                ;*SUBTEST        BLOCK MOVE SUBROUTINE
                                ;*******************************************************************************
                                        ;BLOCK3 HAS 3 ARGUEMENTS
                                        ;BLOCK2 HAS 2 ARGUEMENTS
                                        ;BLOCK1 HAS 1 ARGUEMENTS
                                        ;
                                        ;ALL ARE CALLED BY THE BMOV MACRO
                                        .ENABL  LSB
043720                          BLOCK1: PUSH    R0,R1,R2
043726  012702  177640                  MOV     #FASTCITY,R2
043732  012701  000020                  MOV     #16.,R1
043736  000413                          BR      3$
                                
043740                          BLOCK2: PUSH    R0,R1,R2
043746  012701  000020                  MOV     #16.,R1
043752  000404                          BR      2$
                                
043754                          BLOCK3: PUSH    R0,R1,R2
043762  012501                          MOV     (R5)+,R1
043764  012502                  2$:     MOV     (R5)+,R2
043766  012500                  3$:     MOV     (R5)+,R0
                                
043770  012022                  1$:     MOV     (R0)+,(R2)+
043772  077102                          SOB     R1,1$
043774                                  POP     R2,R1,R0
044002  000205                          RTS     R5
                                        .DSABL  LSB
```

```
                                        .SBTTL  FIELD SERVICE MODE

044004                          FIELDSERVICE:SUBTST     <<SUBR  FIELD SERVICE COMMAND MODE>>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    FIELD SERVICE COMMAND MODE
                                ;*****************************************************************************
044004  104415                          SAVREG
044006                                  TYPE    MSG020          ;FIELD SERVICE COMMAND MODE

044012                                  IF RLFLAG IS TRUE OR NOFSMODE IS TRUE
044026                                    TYPE  MSG048          ;NOT AVAILABLE NOW - TRY LATER!
044032  104416                            RESREG
044034  000207                            RETURN
044036                                  END ;OF IF RLFLAG
044036  005737  002514                  TST     CACHKN
044042  001402                          BEQ     1$
044044                                  PUSH    CONTRL          ;SAVE CACHE STATUS
044050                          1$:     PUSH    CSRNO,KAMIKAZE  ;SAVE CSR & KAMIKAZE STATUS
044060  104424                          CACHOFF                 ;TURN CACHE OFF
044062                                  SET     KAMIKAZE
044070                          FS1:    TYPE    MSG026          ;COMMAND:
044074  104414                          RDDEC                   ;READ A DECIMAL NUMBER
044076                                  POP     R0              ;COMMAND --> R0
044100  020027  000024                  CMP     R0,#20.
044104  101403                          BLOS    1$
044106                                  TYPE    MSG021
044112  000766                          BR      FS1
044114                          1$:     CASE R0
044124  044200                            FSCMD0                ;EXIT FIELD SERVICE COMMANDS
044126  044302                            FSCMD1                ;READ CSR
044130  044406                            FSCMD2                ;LOAD CSR
044132  044550                            FSCMD3                ;EXAMINE MEMORY
044134  045022                            FSCMD4                ;MODIFY MEMORY
044136  045340                            FSCMD5                ;SELECT BANK & PATTERN
044140  046172                            FSCMD6                ;TYPE CONFIGURATION MAP
044142  046200                            FSCMD7                ;BATTERY BACKUP - P/F TEST
044144  046520                            FSCMD8                ;SOB-A-LONG TEST
044146  047012                            FSCMD9                ;ERROR SUMMARY
044150  047216                            FCMD10                ;REFRESH TEST
044152  047510                            FCMD11                ;SET FILL COUNT
044154  047536                            FCMD12                ;ENTER KAMIKAZE MODE
044156  047560                            FCMD13                ;EXIT KAMIKAZE MODE
044160  047600                            FCMD14                ;TURN CACHE OFF
044162  047622                            FCMD15                ;TURN CACHE ON
044164  047640                            FCMD16                ;TEST ONLY SELECTED BANKS
044166  047724                            FCMD17                ;RESUME TESTING ALL BANKS
044170                                  END ;OF CASE
044176  000734                          BR      FS1
```

```
7522 044200                          FSCMD0: SUBTST  <<COMMAND 0     EXIT>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        COMMAND 0       EXIT
                                     ;*****************************************************************************
7523 044200                                  TYPE    MSG103          ;LEAVING FIELD SERVICE MODE
7524 044204  062706  000002                 ADD     #2,SP
7525 044210                                  IF SKIPKAMI IS TRUE
7526 044216  062706  000002                   ADD   #2,SP                   ;THROW AWAY OLD KAMIKAZE FLAG
7527 044222  005037  002006                   CLR   SKIPKAMI
7528 044226                                  ELSE
7529 044230                                    POP   KAMIKAZE              ;RESTORE OLD KAMIKAZE FLAG
7530 044234                                  END ;OF IF SKIPKAMI
7531 044234                                  POP     CSRNO
7532 044240  005737  002514                 TST     CACHKN
7533 044244  001414                         BEQ     RES0
7534 044246                                  IF CACHKN EQ CACHKF             ;IF CACHE IS OFF
7535 044256  062706  000002                   ADD   #2,SP                   ;THROW AWAY CACHE STATUS
7536 044262                                  ELSE
7537 044264  005737  002514                    TST   CACHKN
7538 044270  001402                            BEQ   RES0
7539 044272                                    POP   CONTRL                ;RESTORE CACHE STATUS
7540 044276                                  END ;OF IF CACHKN
7541 044276  104416                  RES0:   RESREG
7542 044300  000207                          RETURN
7543
7544 044302                          FSCMD1: SUBTST  <<FS    COMMAND 1       READ CSR>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        FS      COMMAND 1       READ CSR
                                     ;*****************************************************************************
7545 044302  004737  047766                 CALL    WHICHCSR
7546 044306  010637  002266                 MOV     SP,FSSTACK
7547 044312                                  SET4    #RES1               ;TRAPS TO 4 GOTO RES1
7548 044320  104426                         READCSR
7549 044322                                  SET     NOERROR
7550 044330  104026                         ERROR   +26                 ;USE ERROR ROUTINE FOR PRINTOUT
7551 044332                                  RES4                        ;RESET TRAPS TO 4 TO DEFAULT
7552 044352  000207                         RETURN
7553 044354                          RES1:   TYPE    MSG025              ;THIS CSR DOES NOT EXIST
7554 044360  013706  002266                 MOV     FSSTACK,SP
7555 044364                                  RES4                        ;RESET TRAPS TO 4 TO DEFAULT
7556 044404  000207                         RETURN
```

```
 7559 044406                        FSCMD2: SUBTST  <<FS    COMMAND 2       LOAD CSR>>
                                    ;*****************************************************************************
                                    ;*SUBTEST        FS      COMMAND 2       LOAD CSR
                                    ;*****************************************************************************
                                    ;
 7560 044406  004737  047766                CALL    WHICHCSR
 7561 044412  010637  002266                MOV     SP,FSSTACK
 7562 044416                                SET4    #RES2           ;TRAPS TO 4 GOTO RES2
 7563 044424  104426                        READCSR
 7564 044426                                TYPE    MSG027
 7565 044432                                SET     NOERROR
 7566 044440  104026                        ERROR   +26             ;USE ERROR ROUTINE FOR PRINTOUT
 7567 044442                                RES4                    ;RESET TRAPS TO 4 TO DEFAULT
 7568 044462                                TYPE    MSG023          ;FIRST CSR WORD
 7569 044466  104413                        RDOCT                   ;READ AN OCTAL NUMBER
 7570 044470                                POP     CSR             ;PUT IN IN LOC ''CSR''
  571 044474  104425                        LOADCSR
 7572 044476  104426                        READCSR
 7573 044500                                TYPE    MSG028
 7574 044504                                SET     NOERROR
 7575 044512  104026                        ERROR   +26             ;USE FOR PRINTOUT - NOT AN ERROR
 7576 044514  000207                        RETURN
 7577 044516                        RES2:   TYPE    MSG025          ;THIS CSR DOES NOT EXIST
 7578 044522  013706  002266                MOV     FSSTACK,SP
 7579 044526                                RES4                    ;RESET TRAPS TO 4 TO DEFAULT
 7580 044546  000207                        RETURN
```

```
 7583 044550                                FSCMD3: SUBTST  <<FS     COMMAND 3       EXAMINE MEMORY>>
                                            ;*****************************************************************************
                                            ;*SUBTEST        FS      COMMAND 3       EXAMINE MEMORY
                                            ;*****************************************************************************
 7584 044550                                        PUSH    BANK,NOPAR,PARTHERE,4
 7585 044570  012737  000002  002074                MOV     #2,NOPAR            ;INDICATE PARITY ACTION
 7586 044576                                        TYPE    MSG029              ;EXAMINE MEMORY
 7587 044602                                1$:     TYPE    MSG031              ;PHYSICAL ADDRESS (0-17775776)??
 7588 044606  104413                                RDOCT                       ;READ OCTAL NUMBER ONTO STACK & $HIOCT
 7589 044610  013737  060556  002100                MOV     $HIOCT,BANK         ;PUT MSB'S IN BANK
 7590 044616                                        POP     R0                  ;PUT LSB'S IN R0
 7591 044620  000241                                CLC
 7592 044622  006100                                ROL     R0
 7593 044624  006137  002100                        ROL     BANK
 7594 044630  000241                                CLC
 7595 044632  006000                                ROR     R0
 7596 044634  023737  002100  002526                CMP     BANK,LASTBANK       ;CHECK FOR BANK TOO HIGH
 7597 044642  003357                                BGT     1$                  ;BRANCH IF TRUE
 7598 044644  062700  060000                        ADD     #FIRST,R0
 7599 044650  032700  000001                        BIT     #BIT0,R0            ;CHECK FOR ODD ADDRESS
 7600 044654  001352                                BNE     1$                  ;BRANCH IF ODD ADDRESS
 7601 044656  020027  157776                        CMP     R0,#LAST            ;CHECK FOR ADDRESS OVER 16K
 7602 044662  101347                                BHI     1$                  ;BRANCH IF OVER 16K
 7603 044664  012737  044736  002264                MOV     #3$,PARTHERE        ;INCASE OF ABORTS
 7604 044672                                        SET4    #4$                 ;TRAPS TO 4 GOTO 4$
 7605 044700                                        MAP     BANK                        ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
 7606 044714                                        TESTAREA                    ;ENTER TEST MODE
 7607 044722  011001                                MOV     (R0),R1
 7608 044724  104417                                KERNEL                      ;ENTER KERNEL MODE
 7609 044726                                        TYPOCS  R1
 7610 044734  000410                                BR      EXCMD3
 7611
 7612 044736                                3$:     TYPE    MSG032                      ;PARITY ABORT
 7613 044742  000405                                BR      EXCMD3
 7614
 7615 044744  062706  000004                4$:     ADD     #4,SP                       ;FIX STACK
 7616 044750                                        TYPE    MSG033                      ;TIMEOUT TRAP
 7617 044754  000400                                BR      EXCMD3
 7618
 7619 044756  104417                        EXCMD3: KERNEL                      ;ENTER KERNEL MODE
 7620 044760                                        POP     4,PARTHERE,NOPAR,BANK
 7621 045000                                        RES4                        ;RESET TRAPS TO 4 TO DEFAULT
 7622 045020  000207                                RETURN
```

```
                                FSCMD4: SUBTST  <<FS    COMMAND 4       MODIFY MEMORY>>
                                ;*****************************************************************************
                                ;*SUBTEST        FS      COMMAND 4       MODIFY MEMORY
                                ;*****************************************************************************
045022                                  PUSH    BANK,NOPAR,PARTHERE,4
045042  012737  000003  002074          MOV     #3,NOPAR                ;INDICATE PARITY ACTION
045050                                  TYPE    MSG036                  ;MODIFY MEMORY
045054                          1$:     TYPE    MSG031          ;PHYSICAL ADDRESS (0-17775776)??
045060  104413                          RDOCT                   ;READ OCTAL NUMBER ONTO STACK & $HIOCT
045062  013737  060556  002100          MOV     $HIOCT,BANK     ;PUT MSB'S IN BANK
045070                                  POP     R0              ;PUT LSB'S IN R0
045072  000241                          CLC
045074  006100                          ROL     R0
045076  006137  002100                  ROL     BANK
045102  000241                          CLC
045104  006000                          ROR     R0
045106                                  IF BANK GT LASTBANK THEN GOTO 1$  ;CHECK FOR BANK TOO HIGH
045116  062700  060000                  ADD     #FIRST,R0
045122                                  IF #BIT0 SET.IN R0 THEN GOTO 1$ ;CHECK FOR ODD ADDRESS
045130                                  IF R0 HI #LAST THEN GOTO 1$     ;CHECK FOR ADDRESS OVER 16K
045136  012737  045204  002264          MOV     #3$,PARTHERE            ;INCASE OF ABORTS
045144                                  SET4    #4$             ;TRAPS TO 4 GOTO 4$
045152                                  MAP     BANK                    ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
045166  104511                          INVALIDATE
045170                                  TESTAREA                ;ENTER TEST MODE
045176  011001                          MOV     (R0),R1
                                        ;GETTING HERE MEANS WE GOT LUCKY - NO TRAPS
045200  104417                          KERNEL                  ;ENTER KERNEL MODE
045202  000410                          BR      5$

045204                          3$:     TYPE    MSG032                  ;PARITY ABORT
045210  000431                          BR      EXCMD4                  ;EXIT

045212  062706  000004          4$:     ADD     #4,SP           ;FIX STACK
045216                                  TYPE    MSG033          ;TIMEOUT TRAP
045222  000424                          BR      EXCMD4                  ;EXIT

045224                          5$:     TYPE    MSG037                  ;OLD DATA WAS
04523C                                  TYPOCS  R1                      ;PRINT IT
045236                                  TYPE    MSG039                  ;INPUT NEW DATA
045242  104413                          RDOCT                           ;READ ON OCTAL NUMBER ONTO THE STACK
045244                                  POP     R1                      ;GET NEW NUMBER
045246                                  TESTAREA                ;ENTER TEST MODE
045254  010110                          MOV     R1,(R0)                 ;PUT IT IN MEMORY
045256  011001                          MOV     (R0),R1                 ;READ IT AGAIN
045260  104417                          KERNEL                  ;ENTER KERNEL MODE
045262                                  TYPE    MSG038                  ;DATA IS NOW
045266                                  TYPOCS  R1                      ;PRINT IT

045274  104417                  EXCMD4: KERNEL                  ;ENTER KERNEL MODE
045276                                  POP     4,PARTHERE,NOPAR,BANK
045316                                  RES4                    ;RESET TRAPS TO 4 TO DEFAULT
045336  000207                          RETURN
```

```
7677 045340                                FSCMD5: SUBTST  <<FS    COMMAND 5       SELECT BANK & PATTERN>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       FS      COMMAND 5       SELECT BANK & PATTERN
                                           ;*****************************************************************************
7678 045340                                        PUSH    BANK,PATTERN,TESTADD,PCBUMP,TKVEC,TKVEC+2
7679 045370 010637  002266                         MOV     SP,FSSTACK              ;SAVE LAST GOOD STACK POINTER
7680 045374                                        TYPE    MSG040                  ;SELECT BANK & PATTERN TEST
7681 045400                                1$:     TYPE    MSG030                  ;BANK(0-177)?
7682 045404 104413                                 RDOCT                           ;READ AN OCTAL NUMBER ONTO THE STACK
7683 045406                                        POP     BANK                    ;PUT IT IN BANK
7684 045412                                        IF BANK GT LASTBANK THEN GOTO 1$  ;CHECK FOR BANK TOO HIGH
7685
7686 045422 013701  002100                         MOV     BANK,R1
7687 045426 006301                                 ASL     R1
7688 045430 006301                                 ASL     R1
7689 045432                                        IF CPUBIT OFF.IN CONFIG(R1)
7690 045442                                          TYPE  MSG041                  ;BANK NOT ACCESSABLE
7691 045446                                          GOTO 1$
7692 045450                                        END ;OF IF
7693
7694 045450                                2$:     TYPE    MSG042                  ;PATTERN(0-35)?
7695 045454 104413                                 RDOCT                           ;READ AN OCTAL NUMBER ONTO THE STACK
7696 045456                                        POP     PATTERN                 ;PUT IT IN PATTERN
7697 045462                                        IF PATTERN GT #35 THEN GOTO 2$  ;CHECK FOR PATTERN TO HIGH
7698 045472                                        IF PATTERN EQ #0
7699 045500                                          TYPE  MSG043                  ;PATTERN 0 DATA IS?
7700 045504 104413                                   RDOCT                         ;READ AN OCTAL NUMBER ONTO THE STACK
7701 045506                                          POP   R2                      ;PUT IT IN R2
7702 045510                                        END ;OF IF
7703
7704
7705 045510                                        MAP     BANK                    ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
7706 045524 104511                                 INVALIDATE
7707 045526 004737  042700                         CALL    EXBANK                  ;SET NEW MARGINS
7708 045532                                        IF RRFLAG IS TRUE
7709 045540                                          TYPE  MSG049                  ;BANK REQUIRES RELOCATION
7710 045544                                          GOTO CMD5C
7711 045546                                        END ;OF IF RRFLAG
7712 045546                                        TYPE    MSG046          ;TO ESCAPE TYPE ANY KEY!
7713 045552 012737  046102  000060                 MOV     #CMD5C,TKVEC
7714 045560 012737  000340  000062                 MOV     #340,TKVEC+2
7715 045566 017700  135012                         MOV     @$TKB,R0                ;KILL ANY OLD INTERRUPT
7716 045572 042737  000200  177776                 BIC     #BIT7,PSW               ;LOWER CPU PRIORITY TO 140
7717 045600 052777  000100  134774                 BIS     #BIT6,@$TKS             ;ENABLE KEYBOARD INTERRUPTS
7718
7719
7720 045606                                        SET     HEADER,MUT
7721 045622 013701  002100          CMD5B:         MOV     BANK,R1
7722 045626 006301                                 ASL     R1
7723 045630 006301                                 ASL     R1
7724 045632 005037  002232                         CLR     SPLTCSR
7725 045636 005037  002256                         CLR     PASFLG
7726 045642 012737  060000  002362                 MOV     #FIRST,TESTADD
7727 045650 012737  060002  002364                 MOV     #FIRST+2,TESTADD+2
7728 045656                                        IF #BIT12 SET.IN CONFIG+2(R1)
7729 045666 005237  002232                           INC   SPLTCSR
7730 045672                                          MAP   BANK
```

```
 7731 045706  012737  120000  002364          MOV   #120000,TESTADD+2
 7732 045714  005037  002232                  CLR   SPLTCSR
 7733 045720                              END; OF IF #BIT12
 7734 045720                              IF #SW0 SET.IN @SWR
 7735 045730  104470                        ECCDIS                        ;DISABLE ERROR CORRECTION
 7736 045732                              ELSE
 7737 045734                                PUSH  CSRNO
 7738 045740  104502                        CLRCSR                        ;CLEAR CSRS
 7739 045742                                POP   CSRNO
 7740 045746                              END ;OF IF
 7741 045746  005237  002232              INC     SPLTCSR
 7742 045752  012737  000002  002074      MOV     #2,NOPAR                ;PARITY ACTION
 7743 045760  012737  000002  002276      MOV     #2,PCBUMP               ;TRAPS ADD 2 TO PC
 7744 045766  013700  002110              MOV     PATTERN,R0
 7745 045772  006300                      ASL     R0
 7746 045774  004770  046006              CALL    @FSPAT(R0)
 7747 046000  005037  002074              CLR     NOPAR
 7748 046004  000706                      BR      CMD5B                   ;LOOP TILL KEYBOARD INTERRUPT
 7749
 7750 046006  016760                FSPAT:  MT0000  ;<1 SEC         DATA PATTERN TEST
 7751 046010  017036                        MT0001  ;<1 SEC         ADDRESS TEST
 7752 046012  017154                        MT0002  ;<1 SEC         COMPLEMENT ADDRESS TEST
 7753 046014  017312                        MT0003  ; 1 SEC         3 XOR 9 WORST CASE NOISE TEST
 7754 046016  017542                        MT0004  ; 1 SEC         ROTATING ZEROS TEST
 7755 046020  017662                        MT0005  ; 1 SEC         ROTATING ONES TEST
 7756 046022  020014                        MT0006  ;<1 SEC         INITIAL DATA TEST
 7757 046024  020050                        MT0007  ;<1 SEC         ADDRESS BIT TEST
 7758 046026  020112                        MT0010  ;<1 SEC         BYTE ADDRESSING TEST
 7759 046030  020146                        MT0011  ;<2 SEC         CREATE SINGLE BIT ERROR TEST
 7760 046032  020214                        MT0012  ;<1 SEC         WRITE BYTE CLEARS SBE TEST
 7761 046034  020310                        MT0013  ; 1 SEC         CREATE DOUBLE BIT ERROR TEST
 7762 046036  020364                        MT0014  ; 1 SEC         WRITE INHIBIT DURING DATIP WITH DBE
 7763 046040  020442                        MT0015  ; 1 SEC         WRITE INHIBIT OF BYTE WITH DBE
 7764 046042  020510                        MT0016  ;<1 SEC         WRITE INHIBIT OF WORD WITH DBE
 7765 046044  020556                        MT0017  ;<1 SEC         HOLDING 1'S & 0'S TEST
 7766 046046  020600                        MT0020  ;<1 SEC         MARCHING 1'S & 0'S IN CHECK BITS
 7767 046050  021630                        MT0021  ; 1 SEC         MARCHING 0'S & 1'S TEST
 7768 046052  022052                        MT0022  ;10 SEC         REFRESH & SHIFTING DIAGONAL TEST
 7769 046054  022104                        MT0023  ;10 SEC         SHIFTING DIAGONAL TEST
 7770 046056  022150                        MT0024  ;20 SEC         FAST GALLOPING PATTERN TEST
 7771 046060  022402                        MT0025  ;<1 SEC         INTERRUPT ENABLE TEST
 7772 046062  022450                        MT0026  ;<1 SEC         RANDOM DATA TEST
 7773 046064  022740                        MT0027  ; 1 SEC         UNIQUE BANK TEST
 7774 046066  023412                        MT0030  ; 1 SEC         FLUSH OUT DBE'S TEST
 7775 046070  023712                        MT0031  ; 3 SEC         SOB-A-LONG TEST
 7776 046072  024102                        MT0032  ;<1 SEC         WRITE RECOVERY TEST
 7777 046074  024430                        MT0033  ;35 SEC         BRANCH GOBBLE TEST
 7778 046076  024616                        MT0034  ; 1 SEC         SOFT ERROR TEST
 7779 046100  024762                        MT0035  ;<1 SEC         WORST CASE NOISE PARITY TEST
 7780
 7781 046102  013706  002266        CMD5C:  MOV     FSSTACK,SP              ;RECOVER OLD STACK POINTER
 7782 046106  042777  000100  134466        BIC     #BIT6,@$TKS
 7783 046114                              POP     TKVEC+2,TKVEC
 7784 046124  117700  134454              MOVB    @$TKB,R0                ;GET CHARACTER TO GET RID OF FLAG
 7785 046130                              POP     PCBUMP,TESTADD
 7786 046140                              POP     PATTERN,BANK
 7787 046150                              MAP     BANK                    ;REMAP OLD BANK
```

```
 7788 046164  004737  042700                  CALL    EXBANK
 7789 046170  000207                          RETURN
```

```
 7791 046172                        FSCMD6: SUBTST  <<FS    COMMAND 6       TYPE CONFIGURATION MAP>>
                                    ;*****************************************************************************
                                    ;*SUBTEST       FS      COMMAND 6       TYPE CONFIGURATION MAP
                                    ;*****************************************************************************
                                    ;
 7792 046172  004737  035254               CALL    PCONFIG
 7793 046176  000207                        RETURN
 7794
```

```
7797 046200                          FSCMD7: SUBTST  <<FS    COMMAND 7       BATTERY BACKUP TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       FS      COMMAND 7       BATTERY BACKUP TEST
                                     ;*****************************************************************************
7798 046200                                  PUSH    BANK,PATTERN,CSRNO,TKVEC,TKVEC+2
7799 046224                                  SET     FS7FLAG
7800 046232                                  IF #SW0 SET.IN @SWR
7801 046242  104470                            ECCDIS                        ;DISABLE ERROR CORRECTION
7802 046244                                  ELSE
7803 046246  104502                            CLRCSR                        ;CLEAR CSRS
7804 046250                                  END ;OF IF
7805 046250                                  TYPE    MSG050          ;BATTERY BACKUP TEST
7806                                                                  ;WRITING & CHECKING ADDRESS PATTERN AS BACKGROUND
7807 046254  004737  022740                  CALL    MT0027                  ;CREATE & CHECK BACKGROUND
7808 046260  012737  000012  002310          MOV     #10.,TIME
7809 046266                                  TYPE    MSG052                  ;REMOVE SYSTEM POWER FOR
7810 046272                                  TYPDEC  TIME
7811 046300                                  TYPE    MSG053                  ;SECONDS MAX!
7812 046304                                  TYPE    MSG046                  ;TO ESCAPE TYPE ANY KEY
7813 046310  012737  046352  000060          MOV     #CMD7A,TKVEC
7814 046316  012737  000340  000062          MOV     #340,TKVEC+2
7815 046324  017700  134254                  MOV     @$TKB,R0                ;KILL ANY OLD INTERRUPT
7816 046330  042737  000200  177776          BIC     #BIT7,PSW               ;LOWER CPU PRIORITY TO 140
7817 046336  052777  000100  134236          BIS     #BIT6,@$TKS             ;ENABLE KEYBOARD INTERRUPTS
7818 046344  010637  002266                  MOV     SP,FSSTACK
7819
7820 046350  000001                          WAIT
7821
7822 046352  013706  002266          CMD7A:  MOV     FSSTACK,SP              ;RECOVER OLD STACK POINTER
7823 046356  042777  000100  134216          BIC     #BIT6,@$TKS
7824 046364                                  POP     TKVEC+2,TKVEC
7825 046374  117700  134204                  MOVB    @$TKB,R0                ;GET CHARACTER TO GET RID OF FLAG
7826 046400                                  TYPE    MSG054                  ;NOW STARTING READ TEST OF MEMORY BANKS
7827 046404                                  FOR BANK := #0 TO LASTBANK
7828 046410  004737  042700                    CALL  EXBANK
7829 046414                                    IF ACFLAG IS TRUE AND RRFLAG IS FALSE
7830 046430  104511                              INVALIDATE
7831 046432                                      LET R2 := BANK
7832 046436  012700  060000                      MOV #FIRST,R0
7833 046442  010004                              MOV R0,R4
7834 046444  012701  040000                      MOV #SIZE,R1
7835 046450  010103                              MOV R1,R3
7836 046452  004737  025322                      CALL        SUPD03
7837 046456                                    END ;OF IF ACFLAG
7838 046456                                  END ;OF FOR BANK
7839 046472                                  TYPE    MSG047                  ;TEST COMPLETE
7840 046476  005037  002416                  CLR     FS7FLAG
7841 046502                                  POP     CSRNO,PATTERN,BANK
7842 046516  000207                          RETURN
7843
```

```
 7846 046520                                FSCMD8: SUBTST  <<FS    COMMAND 8       SOB-A-LONG TEST>>
                                                    ;*****************************************************************************
                                                    ;*SUBTEST       FS      COMMAND 8       SOB-A-LONG TEST
                                                    ;*****************************************************************************
 7847 046520                                        PUSH    BANK,PATTERN,TKVEC,TKVEC+2,NOPAR
 7848 046544  010637  002266                        MOV     SP,FSSTACK              ;SAVE LAST GOOD STACK POINTER
 7849 046550                                        TYPE    MSG055                  ;SOB-A-LONG TEST
 7850
 7851 046554                                        IF #SW0 SET.IN @SWR
 7852 046564  104470                                  ECCDIS                        ;DISABLE ERROR CORRECTION
 7853 046566                                        ELSE
 7854 046570  104502                                  CLRCSR                        ;CLEAR CSRS
 7855 046572                                        END ;OF IF
 7856 046572                                        TYPE    MSG056                  ;BELL = EACH PASS COMPLETE
 7857
 7858 046576                                        TYPE    MSG046                  ;TO ESCAPE TYPE ANY KEY!
 7859 046602  012737  046726  000060                MOV     #CMD8C,TKVEC
 7860 046610  012737  000340  000062                MOV     #340,TKVEC+2
 7861 046616  017700  133762                        MOV     @$TKB,R0                ;KILL ANY OLD INTERRUPT
 7862 046622  042737  000200  177776                BIC     #BIT7,PSW               ;LOWER CPU PRIORITY TO 140
 7863 046630  052777  000100  133744                BIS     #BIT6,@$TKS             ;ENABLE KEYBOARD INTERRUPTS
 7864
 7865
 7866 046636                                        SET     HEADER,MUT
 7867
 7868 046652                                CMD8B:  FOR BANK := #0 TO LASTBANK
 7869 046656  004737  042700                          CALL EXBANK
 7870 046662                                          IF ACFLAG IS TRUE AND RRFLAG IS FALSE
 7871 046676  104511                                    INVALIDATE
 7872 046700  004737  023712                            CALL MT0031
 7873 046704                                          END ;OF IF ACFLAG
 7874 046704                                        END ;OF FOR BANK
 7875 046720                                        TYPE    $BELL                   ;RING BELL
 7876 046724                                        GOTO    CMD8B
 7877
 7878 046726  013706  002266            CMD8C:      MOV     FSSTACK,SP                      ;RECOVER OLD STACK POINTER
 7879 046732  042777  000100  133642                BIC     #BIT6,@$TKS
 7880 046740  117700  133640                        MOVB    @$TKB,R0                ;READ CHAR TO KILL FLAG
 7881 046744                                        POP     NOPAR,TKVEC+2,TKVEC,PATTERN,BANK
 7882 046770                                        MAP     BANK                    ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
 7883 047004  004737  042700                        CALL    EXBANK
 7884 047010  000207                                RETURN
```

```
7887 047012                                FSCMD9: SUBTST  <<FS     COMMAND 9       ERROR SUMMARY>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       FS       COMMAND 9       ERROR SUMMARY
                                           ;*****************************************************************************
7888 047012                                        PUSH    R0,R2,R3,BANK
7889 047024  013737  061264  002404               MOV     $PASS,TEMP
7890 047032  005337  002404                        DEC     TEMP
7891 047036                                        TYPDEC  TEMP
7892 047044                                        TYPE    MSG125                  ;PASSES COMPLETED
7893 047050                                        TYPDEC  $ERTTL
7894 047056                                        TYPE    MSG079                  ;ERROR(S) DETECTED
7895 047062                                        IF $ERTTL NE #0
7896 047070  005037  002304                          CLR   SUCCESS
7897 047074                                          FOR BANK := #0 TO LASTBANK
7898 047100  013703  002100                            MOV BANK,R3
7899 047104  070327  000004                            MUL #4,R3
7900 047110                                            IFB CONFIG+2(R3) NE #0
7901 047116                                              IF SUCCESS IS FALSE
7902 047124                                                TYPE    MSG076          ;BANK  ERRORS
7903 047130                                                SET SUCCESS
7904 047136                                              END ;OF IF SUCCESS
7905 047136                                              TYPOCS    BANK,3
7906 047146  116300  002626                              MOVB      CONFIG+2(R3),R0
7907 047152  042700  177400                              BIC       #^C377,R0
7908 047156                                              TYPDEC    R0
7909 047162                                              TYPE      $CRLF
7910 047166                                            END ;OF IFB CONFIG(R3)
7911 047166                                          END ;OF FOR BANK
7912 047202                                        END ;OF IF $ERTTL
7913 047202                                        POP     BANK,R3,R2,R0
7914 047214  000207                                RETURN
```

```
                                FCMD10: SUBTST  <<FS    COMMAND 10      REFRESH TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        FS      COMMAND 10      REFRESH TEST
                                ;*****************************************************************************
                                        PUSH    BANK,PATTERN,TKVEC,TKVEC+2,NOPAR
                                        MOV     SP,FSSTACK              ;SAVE LAST GOOD STACK POINTER
                                        TYPE    MSG073                  ;REFRESH TEST

                                        IF #SW0 SET.IN @SWR
                                          ECCDIS                        ;DISABLE ERROR CORRECTION
                                        ELSE
                                          CLRCSR                        ;CLEAR CSRS
                                        END ;OF IF
                                        TYPE    MSG056                  ;BELL = EACH PASS COMPLETE

                                        TYPE    MSG046                  ;TO ESCAPE TYPE ANY KEY!
                                        MOV     #CMD10C,TKVEC
                                        MOV     #340,TKVEC+2
                                        MOV     @$TKB,R0                ;KILL ANY OLD INTERRUPT
                                        BIC     #BIT7,PSW               ;LOWER CPU PRIORITY TO 140
                                        BIS     #BIT6,@$TKS             ;ENABLE KEYBOARD INTERRUPTS

                                        SET     HEADER,MUT

                                CMD10B: FOR BANK := #0 TO LASTBANK
                                          CALL EXBANK
                                          IF ACFLAG IS TRUE AND RRFLAG IS FALSE
                                            INVALIDATE
                                            CALL MT0022
                                          END ;OF IF ACFLAG
                                        END ;OF FOR BANK
                                        TYPE    $BELL                   ;RING BELL
                                        GOTO    CMD10B

                                CMD10C: MOV     FSSTACK,SP              ;RECOVER OLD STACK POINTER
                                        BIC     #BIT6,@$TKS
                                        MOVB    @$TKB,R0                ;READ CHAR TO KILL FLAG
                                        POP     NOPAR,TKVEC+2,TKVEC,PATTERN,BANK
                                        MAP     BANK                    ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
                                        CALL    EXBANK
                                        RETURN
```

```
 7958 047510                            FCMD11: SUBTST  <<FS     COMMAND 11       SET FILL COUNT>>
                                        ;*****************************************************************************
                                        ;*SUBTEST        FS      COMMAND 11       SET FILL COUNT
                                        ;*****************************************************************************
 7959 047510                                    PUSH    R0
 7960 047512                                    TYPE    MSG085                   ;FILL COUNT(OCTAL)?
 7961 047516  104413                            RDOCT
 7962 047520                                    POP     R0
 7963 047522  042700  177760                    BIC     #^C17,R0
 7964 047526  110037  002327                    MOVB    R0,$FILLS
 7965 047532                                    POP     R0
 7966 047534  000207                            RETURN
 7967
 7968 047536                            FCMD12: SUBTST  <<FS     COMMAND 12       ENTER KAMIKAZE MODE>>
                                        ;*****************************************************************************
                                        ;*SUBTEST        FS      COMMAND 12       ENTER KAMIKAZE MODE
                                        ;*****************************************************************************
 7969 047536                                    TYPE    MSG101           ;ENTERING KAMIKAZE MODE
 7970 047542                                    SET     KAMIKAZE,SKIPKAMI
 7971 047556  000207                            RETURN
 7972
 7973 047560                            FCMD13: SUBTST  <<FS     COMMAND 13       EXIT KAMIKAZE MODE>>
                                        ;*****************************************************************************
                                        ;*SUBTEST        FS      COMMAND 13       EXIT KAMIKAZE MODE
                                        ;*****************************************************************************
 7974 047560                                    TYPE    MSG102           ;LEAVING KAMIKAZE MODE
 7975 047564  005037  002004                    CLR     KAMIKAZE
 7976 047570                                    SET     SKIPKAMI
 7977 047576  000207                            RETURN
 7978
 7979 047600                            FCMD14: SUBTST  <<FS     COMMAND 14       TURN CACHE OFF>>
                                        ;*****************  *****************************************************************
                                        ;*SUBTEST        FS      COMMAND 14       TURN CACHE OFF
                                        ;*****************************************************************************
 7980 047600                                    TYPE    MSG106                   ;CACHE IS OFF
 7981 047604  104424                            CACHOFF                          ;TURN CACHE OFF
 7982 047606  013737  002514  002516            MOV     CACHKN,CACHKN+2          ;SAVE OLD CACHE ON STATE
 7983 047614  005037  002514                    CLR     CACHKN                   ;KEEP CACHE OFF
 7984 047620  000207                            RETURN
 7985
 7986 047622                            FCMD15: SUBTST  <<FS     COMMAND 15       TURN CACHE ON>>
                                        ;*****************************************************************************
                                        ;*SUBTEST        FS      COMMAND 15       TURN CACHE ON
                                        ;*****************************************************************************
 7987 047622                                    TYPE    MSG107                   ;CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)
 7988 047626  013737  002516  002514            MOV     CACHKN+2,CACHKN          ;RESTORE OLD CACHE ON STATE
 7989 047634  104423                            CACHON                           ;TURN CACHE ON
 7990 047636  000207                            RETURN
 7991
```

```
 8004
 8005 047640                        FCMD16: SUBTST  <<FS     COMMAND 16      TEST ONLY SELECTED BANKS>>
                                    ;*****************************************************************************
                                    ;*SUBTEST       FS       COMMAND 16      TEST ONLY SELECTED BANKS
                                    ;*****************************************************************************
 8006 047640                                TYPE    MSG105          ;ENTER BANKS IN OCTAL - USE NUMBER OUTSIDE RANGE TO TERMINAT
 8007 047644  004737  047734                CALL    CMD17A                  ;ERASE OLD SELECTIONS
 8008 047650                                BEGIN CMD16LOOP
 8009 047650                                  REPEAT
 8010 047650                                    TYPE        MSG030          ;BANK(0-177)?
 8011 047654  104413                           RDOCT                       ;READ AN OCTAL NUMBER ONTO THE STACK
 8012 047656                                   POP R1                      ;PUT IT IN R1
 8013 047660                                   IF R1 GT #177 OR R1 LT #0
 8014 047672                                     LEAVE CMD16LOOP
 8015 047674                                   END ;OF IF R1
 8016 047674  006301                           ASL  R1
 8017 047676  006301                           ASL  R1                     ;R1 <- R1 * 4
 8018 047700  052761  040000  002626           BIS #BIT14,CONFIG+2(R1)
 8019 047706                                  END ;OF REPEAT
 8020 047710                                END CMD16LOOP
 8021 047710                                TYPE    MSG110                  ;ONLY SELECTED BANKS WILL BE TESTED
 8022 047714                                SET     SELONLY
 8023 047722  000207                        RETURN
 8024
 8025 047724                        FCMD17: SUBTST  <<FS     COMMAND 17      RESUME TESTING ALL BANKS>>
                                    ;*****************************************************************************
                                    ;*SUBTEST       FS       COMMAND 17      RESUME TESTING ALL BANKS
                                    ;*****************************************************************************
 8026 047724                                TYPE    MSG111                  ;ALL BANKS WILL BE TESTED
 8027 047730  005037  002000                CLR     SELONLY
 8028
 8029                                       ;ENTRY POINT FROM CMD16
 8030 047734  013702  002526        CMD17A: MOV     LASTBANK,R2
 8031 047740  006302                        ASL     R2
 8032 047742  006302                        ASL     R2
 8033 047744                                FOR R1 := #0 TO R2 BY #4
 8034 047746  042761  040000  002626          BIC   #BIT14,CONFIG+2(R1)
 8035 047754                                END ;OF FOR R1
 8036 047764  000207                        RETURN
```

```
 8039 047766                        WHICHCSR:SUBTST <<SUBR  DETERMINE CORRECT CSR>>
                                    ;*****************************************************************************
                                    ;*SUBTEST       SUBR    DETERMINE CORRECT CSR
                                    ;*****************************************************************************
 8040 047766  013700  002216               MOV     TOTCSRS,R0      ;GET CSR'S FLAG
 8041 047772  022700  100000               CMP     #BIT15,R0       ;CSR 0?
 8042 047776  001003                       BNE     1$              ;NO - SKIP
 8043 050000  005037  002146               CLR     CSRNO           ;YES - SET IT UP
 8044 050004  000207                       RETURN
 8045
 8046 050006                        1$:     TYPE    MSG022          ;WHICH CSR(0-F)
 8047 050012  104412                       RDLIN                   ;GET CHARACTER
 8048 050014                               POP     R0              ;PUT IN R0
 8049 050016  011000                       MOV     (R0),R0         ;PUT CHAR IN R0
 8050 050020  020027  000106               CMP     R0,#106         ;CHECK LIMIT
 8051 050024  101370                       BHI     1$              ;IF BAD LOOP TILL HE TYPES IT RIGHT
 8052 050026  022700  000101               CMP     #'A,R0
 8053 050032  103002                       BHIS    2$
 8054 050034  162700  000007               SUB     #7,R0
 8055 050040  162700  000060        2$:    SUB     #60,R0
 8056 050044  006300                       ASL     R0
 8057 050046  010037  002146               MOV     R0,CSRNO
 8058 050052  000207                       RETURN
```

```
 8607                                   .SBTTL  ERROR DATA (SUPERVISOR) SETUP STUFF
 8608 050054                    $PER25: LET ADDRESS := R1 - #2
 8609 050066                            IF ABORTFLAG IS FALSE
 8610 050074                              TESTAREA                  ;ENTER TEST MODE
 8611 050102                              LET BAD := -2(R1)
 8612 050110  104417                      KERNEL                    ;ENTER KERNEL MODE
 8613 050112                            END ;OF IF ABORTFLAG
 8614 050112                            IF 177654 EQ #0
 8615 050120                              LET GOOD := R2
 8616 050124                            ELSE
 8617 050126                              LET GOOD := R3
 8618 050132                            END ;OF IF
 8619 050132  000137  053230            JMP     PERRAW
 8620
 8621 050136                    PERRA3: SUBTST  <<DATA WAS 3 WORDS>>
                                ;*****************************************************************************
                                ;*SUBTEST       DATA WAS 3 WORDS
                                ;*****************************************************************************
 8622 050136                            IF BADPC EQ #0 THEN $CALL BADSTACK
 8623 050150                            PUSH    R0
 8624 050152  005037  002144            CLR     CSR               ;MAKE SURE CSR BIT HOLDER IS CLEAR
 8625 050156  104505                    CHK1DIS                   ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
 8626 050160                            TESTAREA
 8627 050166  005711                    TST     (R1)              ;READ LOCATION TO READ CHECKBITS INTO CSR
 8628 050170  104417                    KERNEL
 8629 050172  104426                    READCSR                   ;GET CSR CONTENTS
 8630 050174  013700  002144            MOV     CSR,R0            ;SAVE CSR CONTENTS IN R0
 8631 050200  104503                    CLR1CSR                   ;RETURN CSR TO NORMAL MODE
 8632 050202  072027  177773            ASH     #-5,R0            ;MOVE CHECK BITS TO BOTTOM OF WORD
 8633 050206  042700  177600            BIC     #^C177,R0         ;CLEAR OFF EXTRANEOUS GARBAGE
 8634 050212                            LET ADDRESS := R1         ;SAVE VIRTUAL ADDRESS FOR PRINTOUT
 8635 050216  005037  002042            CLR     GOOD              ;FIRST TEST WORD WRITTEN SHOULD ALWAYS BE ZERO
 8636 050222                            TESTAREA                  ;ENTER TEST MODE
 8637 050230  011137  002050            MOV     (R1),BAD          ;GET BAD DATA FROM MUT - FIRST WORD
 8638 050234  011437  002052            MOV     (R4),BAD2         ;AND SECOND WORD
 8639 050240  104417                    KERNEL                    ;ENTER KERNEL MODE
 8640 050242  110037  002054            MOVB    R0,BAD3           ;MOVE BAD CHECKBITS FOR PRINTOUT
 8641 050246  105037  002055            CLRB    BAD3+1            ;CLEAR OFF THE OTHER UNUSED BITS
 8642 050252  004737  053464            CALL    PERBNK            ;MARK BANK AS BAD IN CONFIG TABLE
 8643 050256  104033                    ERROR   +33
 8644 050260                            POP     R0                ;RESTORE R0
 8645 050262                            IF #SW0 SET.IN @SWR
 8646 050272  104506                      ENASBE                  ;TRAP ON SINGLE BIT ERRORS
 8647 050274                            ELSE
 8648 050276  104472                      ECCINIT                 ;TRAP ON UNCORRECTABLE ERRORS
 8649 050300                            END; OF IF #SW0
 8650 050300  000002                    RTI
```

```
 8653 050302                            $PER30: LET GOOD := R1
 8654 050306                                    LET ADDRESS := (SP) - 16
 8655 050320                                    IF ABORTFLAG IS FALSE
 8656 050326                                      TESTAREA              ;ENTER TEST MODE
 8657 050334                                      LET BAD := @ADDRESS
 8658 050342  104417                              KERNEL                ;ENTER KERNEL MODE
 8659 050344                                    END ;OF IF ABORTFLAG
 8660 050344  000137  053230                    JMP PERRAW
 8661
 8662 050350                            GETDATA:SUBTST  <<GET DATA FROM ABORTED AREA IF POSSIBLE>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       GET DATA FROM ABORTED AREA IF POSSIBLE
                                        ;*****************************************************************************
 8663 050350                                    PUSH    R0,4,114
 8664 050362  010637  050446                    MOV     SP,GETDA1
 8665 050366  012737  050426  000004            MOV     #1$,4
 8666 050374  012737  050426  000114            MOV     #1$,114
 8667 050402  013700  002032                    MOV     ADDRESS,R0
 8668 050406                                    TESTAREA
 8669 050414  011037  002050                    MOV     (R0),BAD
 8670 050420  104417                            KERNEL
 8671 050422  005037  002140                    CLR     ABORTFLAG
 8672 050426  013706  050446            1$:     MOV     GETDA1,SP             ;RESTORE KNOWN GOOD STACK POINTER
 8673 050432                                    POP     114,4,R0
 8674 050444  000207                            RETURN
 8675 050446  000000                    GETDA1: 0
```

```
                                        .SBTTL  POWER FAIL AUTO RESTART
                                        .SBTTL  ROUTINE POWER DOWN AND UP
                                ;*******************************************************************************
                                ;POWER DOWN ROUTINE
050450                          $PWRDN:
                                        ;SAVE CACHE STATUS
050450  005737  002514                  TST     CACHKN
050454  001403                          BEQ     5$
050456                                  PUSH    CONTRL
050462  104423                          CACHON                          ;TURN CACHE ON
050464  012737  051416  000024  5$:     MOV     #$ILLUP,PWRVEC ;;SET FOR FAST UP
050472  012737  000340  000026          MOV     #340,PWRVEC+2   ;;PRIO:7
050500                                  PUSH    R0,R1,R2,R3,R4,R5,CSRNO
                                        ;SAVE USER PAR'S & PDR7
050520  012700  177700                  MOV     #177700,R0
050524  012701  000021                  MOV     #17.,R1
050530                          1$:     PUSH    -(R0)
050532  077102                          SOB     R1,1$
                                        ;SAVE SUPERVISOR PAR'S
050534  005737  002426                  TST     NOSUPER
050540  001013                          BNE     PD1
050542  012700  172300                  MOV     #172300,R0
050546  012701  000020                  MOV     #16.,R1
050552                          2$:     PUSH    -(R0)
050554  077102                          SOB     R1,2$
050556                                  IF RLFLAG IS TRUE THEN $CALL WOOPS
                                        ;COPY KERNEL MAP TO USER & SUPERVISOR
050570  012700  172300          PD1:    MOV     #KIPDR0,R0
050574  012701  177600                  MOV     #UIPDR0,R1
050600  012702  172200                  MOV     #SIPDR0,R2
050604  012703  000040                  MOV     #32.,R3
050610  011021                  3$:     MOV     (R0),(R1)+
050612  012022                          MOV     (R0)+,(R2)+
050614  077303                          SOB     R3,3$
```

```
 8720                                           ;SAVE USER & SUPERVISOR STACK POINTERS
 8721 050616                                    USER
 8722 050624  010600                            MOV     USP,R0
 8723 050626  104417                            KERNEL                  ;ENTER KERNEL MODE
 8724 050630                                    PUSH    R0
 8725 050632  005737  002426                    TST     NOSUPER
 8726 050636  001006                            BNE     7$
 8727 050640                                    SUPERVISOR              ;ENTER SUPERVISOR MODE
 8728 050646  010600                            MOV     SSP,R0
 8729 050650  104417                            KERNEL                  ;ENTER KERNEL MODE
 8730 050652                                    PUSH    R0
 8731                                           ;SAVE ECC REGISTERS
 8732 050654  013701  002216          7$:       MOV     TOTCSRS,R1      ;GET CSR'S
 8733 050660                                    BEGIN   LCSRSAVE
 8734 050660                                      FOR CSRNO := #0 TO #36 BY #2
 8735 050664  006301                                ASL         R1
 8736 050666                                        ON.ERROR
 8737 050670  104426                                  READCSR
 8738 050672                                          PUSH      CSR
 8739 050676                                        END ;OF ON.ERROR
 8740 050676                                      IF R1 EQ #0 THEN LEAVE LCSRSAVE
 8741 050702                                      END ;OF FOR CSRNO
 8742 050720                                    END LCSRSAVE
 8743                                           ;SAVE MMR0,1,2,3
 8744 050720                                    PUSH    MMR0,MMR1,MMR2
 8745 050734  005737  002426                    TST     NOSUPER
 8746 050740  001002                            BNE     8$
 8747 050742                                    PUSH    MMR3
 8748                                           ;SAVE KERNEL PAR'S
 8749 050746  012700  172400          8$:       MOV     #172400,R0
 8750 050752  012701  000020                    MOV     #16.,R1
 8751 050756                          4$:       PUSH    -(R0)
 8752 050760  077102                            SOB     R1,4$
 8753                                           ;SAVE UNIBUS MAP REGISTERS
 8754 050762  005737  002424                    TST     NO22BIT
 8755 050766  001004                            BNE     9$
 8756 050770                                    PUSH    MAPH0,MAPL0
 8757                                           ;SAVE POSSIBLE SOFTWARE SWITCH REGISTER
 8758 051000                          9$:       PUSH    @SWR
 8759                                           ;SAVE STACK POINTER
 8760 051004  010637  051422                    MOV     SP,$SAVR6       ;;SAVE SP
 8761                                           ;NOW SET UP REAL VECTOR
 8762 051010  012737  051022  000024            MOV     #$PWRUP,PWRVEC  ;;SET UP VECTOR
 8763 051016  000000                  $DOWN:    HALT
 8764 051020  000776                            BR      $DOWN           ;;HANG UP
```

```
8767                                    ;*****************************************************************************
8768                                    ;POWER UP ROUTINE
8769 051022                             $PWRUP:
8773 051022  012737  051416  000024             MOV     #$ILLUP,PWRVEC ;;SET FOR FAST DOWN
8774                                    ;RESTORE STACK POINTER
8775 051030  013706  051422             MOV     $SAVR6,SP       ;;GET SP
8776 051034  005037  051422             CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
8777 051040  005237  051422     1$:     INC     $SAVR6          ;;WAIT FOR THE INC
8778 051044  001375                     BNE     1$              ;;OF A WORD
8779                                    ;RESTORE POSSIBLE SOFTWARE SWITCH REGISTER
8780 051046                             POP     @SWR
8781                                    ;RESTORE UNIBUS MAP
8782 051052  005737  002424             TST     NO22BIT
8783 051056  001006                     BNE     10$
8784 051060                             POP     MAPLO,MAPHO
8785 051070  004737  042304             CALL    LOWMAP          ;SETUP LOWER 16K OF UNIBUS MAP
8786                                    ;RESTORE KERNEL PAR'S & PDR'S
8787 051074  012700  172340     10$:    MOV     #172340,R0
8788 051100  012702  172300             MOV     #KIPDR0,R2
8789 051104  012701  000020             MOV     #16.,R1
8790 051110                     6$:     POP     (R0)+
8791 051112  012722  077406             MOV     #77406,(R2)+
8792 051116  077104                     SOB     R1,6$
8793                                    ;RESTORE MMR3,2,1,0
8794 051120  005737  002426             TST     NOSUPER
8795 051124  001002                     BNE     11$
8796 051126                             POP     MMR3
8797 051132                     11$:    POP     MMR2,MMR1,MMR0
8798                                    ;RESTORE ECC REGISTERS
8799 051146  013701  002216             MOV     TOTCSRS,R1      ;GET CSR'S
8800 051152  042701  177400             BIC     #177400,R1
8801 051156                             BEGIN LCSRRESTORE
8802 051156                               FOR CSRNO := #36 DOWNTO #0 BY #2
8803 051164  006201                         ASR         R1
8804 051166                                 ON.ERROR
8805 051170                                   POP       CSR
8806 051174  104425                           LOADCSR
8807 051176                                 END ;OF ON.ERROR
8808 051176                               IF R1 EQ #0 THEN LEAVE LCSRRESTORE
8809 051202                               END ;OF FOR CSRNO
8810 051220                             END LCSRRESTORE
8811                                    ;COPY KERNEL MAP TO USER & SUPERVISOR
8812 051220  012700  172300             MOV     #KIPDR0,R0
8813 051224  012701  177600             MOV     #UIPDR0,R1
8814 051230  012702  172200             MOV     #SIPDR0,R2
8815 051234  012703  000040             MOV     #32.,R3
8816 051240  011021             3$:     MOV     (R0),(R1)+
8817 051242  012022                     MOV     (R0)+,(R2)+
8818 051244  077303                     SOB     R3,3$
```

```
8820                                        ;RESTORE SUPERVISOR & USER STACK POINTERS
8821 051246  005737  002426                 TST     NOSUPER
8822 051252  001006                         BNE     13$
8823 051254                                 POP     R0
8824 051256                                 SUPERVISOR              ;ENTER SUPERVISOR MODE
8825 051264  010006                         MOV     R0,SSP
8826 051266  104417                         KERNEL                  ;ENTER KERNEL MODE
8827 051270                         13$:    POP     R0
8828 051272                                 USER
8829 051300  010006                         MOV     R0,USP
8830 051302  104417                         KERNEL                  ;ENTER KERNEL MODE
8831                                        ;RESTORE SUPERVISOR PAR'S
8832 051304  012700  172240                 MOV     #172240,R0
8833 051310  012701  000020                 MOV     #16.,R1
8834 051314                         7$:     POP     (R0)+
8835 051316  077102                         SOB     R1,7$
8836                                        ;RESTORE USER PAR'S & PDR7
8837 051320  012700  177636                 MOV     #177636,R0
8838 051324  012701  000021                 MOV     #17.,R1
8839 051330                         8$:     POP     (R0)+
8840 051332  077102                         SOB     R1,8$
8841                                        ;RESTORE POSSIBLE SOFTWARE DISPLAY REGISTER
8842 051334  013777  002010  131236         MOV     $PATMAR,@DISPLAY
8843 051342  013737  002010  000174         MOV     $PATMAR,DISPREG
8844 051350                                 POP     CSRNO,R5,R4,R3,R2,R1,R0
8845 051370  012737  050450  000024         MOV     #$PWRDN,PWRVEC  ;;SET UP THE POWER DOWN VECTOR
8846 051376                                 TYPE    MSG051          ;REPORT THE POWER FAILURE
8847                                        ;RESTORE CACHE STATUS
8848 051402  005737  002514                 TST     CACHKN
8849 051406  001402                         BEQ     9$
8850 051410                                 POP     CONTRL
8851 051414  000002                 9$:     RTI
8852 051416  000000                 $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
8853 051420  000776                         BR      $ILLUP          ;; BEFORE THE POWER DOWN WAS COMPLETE
8854 051422  000000                 $SAVR6: 0                       ;;PUT THE SP HERE
8855                                        .EVEN
```

```
 8867 051424                                    WOOPS:  SUBTST  <<POWER FAIL WHILE RELOCATED>>
                                                ;*****************************************************************************
                                                ;*SUBTEST       POWER FAIL WHILE RELOCATED
                                                ;*****************************************************************************
 8868 051424                                            PUSH    BANK
 8869 051430  005037  002100                            CLR     BANK
 8870 051434                                            MAP     BANK                    ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
 8871 051450                                            SUPERVISOR                      ;ENTER SUPERVISOR MODE
 8872 051456  013737  060024  052022                    MOV     FIRST+PWRVEC,WOOPSAV
 8873 051464  013737  060026  052024                    MOV     FIRST+PWRVEC+2,WOOPSAV+2
 8874 051472                                            BMOV    FIRST+WOOPUP,WOOPSAV+4,WOOPEND-WOOPUP/2+12.
 8875 051504  012737  051610  060024                    MOV     #WOOPUP,FIRST+PWRVEC
 8876 051512  012737  000340  060026                    MOV     #340,FIRST+PWRVEC+2
 8877 051520                                            BMOV    WOOPUP,FIRST+WOOPUP,WOOPEND-WOOPUP/2
 8878 051532  012700  172340                            MOV     #KIPAR0,R0
 8879 051536  012701  131772                            MOV     #FIRST+WOOPEND,R1
 8880 051542  012702  000010                            MOV     #8.,R2
 8881 051546  012021                            1$:     MOV     (R0)+,(R1)+
 8882 051550  077202                                    SOB     R2,1$
 8883 051552  005737  002426                            TST     NOSUPER
 8884 051556  001002                                    BNE     2$
 8885 051560  013721  172516                            MOV     MMR3,(R1)+
 8886 051564  013721  177576                    2$:     MOV     MMR2,(R1)+
 8887 051570  013721  177574                            MOV     MMR1,(R1)+
 8888 051574  013721  177572                            MOV     MMR0,(R1)+
 8889 051600  104417                                    KERNEL                          ;ENTER KERNEL MODE
 8890 051602                                            POP     BANK
 8891 051606  000207                                    RETURN
```

```
8894 051610                         WOOPUP: SUBTST  <<POWER UP FROM BANK 0 TO RELOCATION>>
                                    ;*****************************************************************************
                                    ;*SUBTEST       POWER UP FROM BANK 0 TO RELOCATION
                                    ;*****************************************************************************
8895 051610  012700  051772                MOV     #WOOPEND,R0
8896 051614  012701  172340                MOV     #KIPAR0,R1
8897 051620  012703  172300                MOV     #KIPDR0,R3
8898 051624  012702  000010                MOV     #8.,R2
8899 051630  012021                1$:     MOV     (R0)+,(R1)+
8900 051632  012723  077406                MOV     #77406,(R3)+
8901 051636  077204                        SOB     R2,1$
8902 051640  005737  002426                TST     NOSUPER
8903 051644  001002                        BNE     3$
8904 051646  012037  172516                MOV     (R0)+,MMR3
8905 051652  012037  177576          3$:   MOV     (R0)+,MMR2
8906 051656  012037  177574                MOV     (R0)+,MMR1
8907 051662  012037  177572                MOV     (R0)+,MMR0
8908 051666  013706  051422                MOV     $SAVR6,SP
8909 051672                                PUSH    BANK
8910 051676  005037  002100                CLR     BANK
8911 051702                                MAP     BANK                    ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
8912 051716                                SUPERVISOR                      ;ENTER SUPERVISOR MODE
8913 051724  013737  052022  060024        MOV     WOOPSAV,FIRST+PWRVEC
8914 051732  013737  052024  060026        MOV     WOOPSAV+2,FIRST+PWRVEC+2
8915                                        ;SIMULATE THE FOLLOWING BLOCK MOV BUT WITH NO STACK ACCESSES
8916                                        ;BMOV   WOOPSAV+4,FIRST+WOOPUP,WOOPEND-WOOPUP/2+12.
8917 051740  012700  052026                MOV     #WOOPSAV+4,R0
8918 051744  012701  000105                MOV     #WOOPEND-WOOPUP/2+12.,R1
8919 051750  012702  131610                MOV     #FIRST+WOOPUP,R2
8920 051754  012022                2$:     MOV     (R0)+,(R2)+
8921 051756  077102                        SOB     R1,2$
8922
8923 051760  104417                        KERNEL                          ;ENTER KERNEL MODE
8924 051762                                POP     BANK
8925 051766  000137  051022                JMP     $PWRUP
8926 051772  000014          WOOPEND:.REPT   12.
8929 052022  000107          WOOPSAV:.REPT   WOOPEND-WOOPUP/2+12.+2
```

```
                                        .SBTTL  IO SUBROUTINES
                                        .SBTTL  ROUTINE TYPE
                                ;*******************************************************************************
                                ;*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
                                ;*

052240  105737  002330  $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
052244  100407                  BMI     6$              ;BR IF NO
052246  010046          1$:     MOV     R0,-(SP)        ;;SAVE R0
052250  017600  000002          MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
052254  112046          4$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
052256  001005                  BNE     7$              ;;BR IF IT ISN'T THE TERMINATOR
052260  005726                  TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
052262  012600          5$:     MOV     (SP)+,R0        ;;RESTORE R0
052264  062716  000002  6$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
052270  000002                  RTI                     ;;RETURN
052272  122716  000011  7$:     CMPB    #HT,(SP)        ;BRANCH IF NOT <HT>
052276  001002                  BNE     11$
052300  112716  000040          MOVB    #' ,(SP)        ;REPLACE TAB WITH SPACE
052304  122716  000200  11$:    CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
052310  001006                  BNE     8$
052312  005726                  TST     (SP)+           ;;POP  <CR><LF> EQUIV
052314                          TYPE                    ;;TYPE A CR AND LF
052316  002620                  $CRLF
052320  105037  052452          CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
052324  000753                  BR      4$              ;;GET NEXT CHARACTER
052326  004737  052364  8$:     CALL    $TYPEC          ;;GO TYPE THIS CHARACTER
052332  123726  002612  9$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
052336  001346                  BNE     4$              ;;IF NO GO GET NEXT CHAR.
052340  013746  002326          MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                        ;;AND THE NULL CHAR.
052344  105366  000001  10$:    DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
052350  002770                  BLT     9$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
052352  004737  052364          CALL    $TYPEC          ;;GO TYPE A NULL
052356  105337  052452          DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
052362  000770                  BR      10$             ;;LOOP
052364          $TYPEC: PUSH    R1
052366  116601  000004          MOVB    4(SP),R1
052372  005737  002514          TST     CACHKN
052376  001402                  BEQ     2$
052400                          PUSH    CONTRL
052404          2$:     PUSH    R0
052406  104424                  CACHOFF                 ;TURN CACHE OFF
9014 052410  105777  130172  3$:      TSTB    @$TPS           ;;WAIT UNTIL PRINTER I^ READY
```

```
 9015 052414  100375                          BPL     3$
 9016 052416  110177  130166                  MOVB    R1,@$TPB        ;;LOAD CHAR TO BE TYPED INTO DATA REG.
 9020 052422  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
 9021 052430  001003                          BNE     1$              ;;BRANCH IF NO
 9022 052432  105037  052452                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
 9023 052436  000406                          BR      $TYPEX          ;;EXIT
 9024 052440  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
 9025 052446  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
 9026 052450  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
 9027 052452  000000                $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
 9028 052454                        $TYPEX: POP     R0
 9029 052456  005737  002514                  TST     CACHKN          ;IS THERE A CACHE?
 9030 052462  001402                          BEQ     2$              ;BRANCH IF NOT
 9031 052464                                  POP     CONTRL          ;POP CACHE STATUS
 9032 052470                        2$:     POP     R1
 9033 052472  000207                          RETURN
 9034 052474                        SUPLIMIT:;.!!.!!!!!.!!THIS IS THE LIMIT ON SUPERVISOR MAPPED TO MUT SPACE
```

```
                        .SBTTL  ERROR DATA SETUP
;
;USE THIS       IF THIS CONDITION DISCRIBES THE ERROR
;
;PERR01         TRAP
;               BAD DATA IN R0 UNLESS ABORTED
;                       THEN BAD DATA IS POINTED TO BY -(R4)
;               GOOD DATA IN R5
;
;PERR02         TRAP
;               BAD DATA IN R1 UNLESS ABORTED
;                       THEN BAD DATA IS POINTED TO BY -(R4)
;               GOOD DATA IN R2
;
;PERR03         TRAP
;               BAD DATA IS POINTED TO BY -(R1)
;               GOOD DATA IN R4
;
;PERR04         TRAP
;               BAD DATA IN R4 UNLESS ABORTED
;                       THEN BAD DATA IS POINTED TO BY -2(R0)
;               GOOD DATA IN R2
;
;PERR05         JSR     PC
;               BAD DATA IS POINTED TO BY -(R0)
;               GOOD DATA IN R2
;               RETURN AFTER SETTING UP GOOD,BAD,ADDRESS
;
;PERR06         JSR     PC
;               BAD DATA IS POINTED TO BY -(R0)
;               GOOD DATA IS ZERO
;               RETURN AFTER SETTING UP GOOD,BAD,ADDRESS
;
;PERR07         TRAP
;               BAD DATA IN R2 UNLESS ABORTED
;                       THEN BAD DATA IS POINTED TO BY (R1)
;               GOOD DATA IN DATBUF
;
;PERR10         TRAP
;               BAD DATA IN R2 UNLESS ABORTED
;                       THEN BAD DATA IS POINTED TO BY 2(R1)
;               GOOD DATA IN DATBUF+2
;
;PERR11         TRAP
;               BYTE TEST
;               BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
;                       THEN BAD DATA IS POINTED TO BY (R1)
;               GOOD DATA IS A ZERO BYTE
;
;PERR12         TRAP
;               BYTE TEST
;               BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
;                       THEN BAD DATA IS POINTED TO BY (R1)
;               GOOD DATA IS A BYTE OF ONES
;
;PERR13         TRAP
;               BAD DATA IN R0 UNLESS ABORTED
```

```
                                        THEN BAD DATA IS POINTED TO BY (R1)
                                GOOD DATA IS ZERO

PERR14          TRAP
                BAD DATA IN R0 UNLESS ABORTED
                                        THEN BAD DATA IS POINTED TO BY (R1)
                GOOD DATA IS ONES

PERR15          TRAP
                BAD DATA IN R0 UNLESS ABORTED
                                        THEN BAD DATA IS POINTED TO BY (R1)
                GOOD DATA IN TSTDAT

PERR16          TRAP
                BAD DATA IN R0 UNLESS ABORTED
                                        THEN BAD DATA IS POINTED TO BY (R1)
                GOOD DATA IN TSTDAT+2

PERR17          TRAP
                BAD DATA IN R0 UNLESS ABORTED
                                        THEN BAD DATA IS POINTED TO BY (R1)
                GOOD DATA IN R2

PERR20          TRAP
                BAD DATA IN R0 UNLESS ABORTED
                                        THEN BAD DATA IS POINTED TO BY (R1)
                GOOD DATA IN R3

PERR21          TRAP
                7 BIT BYTE TEST
                BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
                                        THEN BAD DATA IS POINTED TO BY (R1)
                GOOD DATA IS A 7 BIT BYTE ON ONES

PERR22          TRAP
                BAD DATA IN R2 UNLESS ABORTED
                                        THEN BAD DATA IS POINTED TO BY (R1)
                GOOD DATA IN R0

PERR23          TRAP
                BAD DATA IN R0 UNLESS ABORTED
                                        THEN BAD DATA IS POINTED TO BY (R1)
                GOOD DATA IN R4

PERR24          TRAP
                BAD DATA IN R0 UNLESS ABORTED
                                        THEN BAD DATA IS POINTED TO BY (R2)
                GOOD DATA IN R3

PERR25          TRAP
                BAD DATA POINTED TO BY -(R1)
                GOOD DATA IN R2 UNLESS LOC V177654 IS SET
                                        THEN GOOD DATA IS IN R3

PERR26          TRAP
                BAD DATA IS DOUBLE WORD POINTED TO BY R1 AND IN LOW 7 BITS OF R0
                GOOD DATA IS 000000,,100000,,100
```

```
9826
9827                    ;PERR27         TRAP
9828                    ;               BAD DATA IS DOUBLE WORD POINTED TO BY R1 AND IN LOW 7 BITS OF R0
9829                    ;               GOOD DATA IS 000000,,000000,,077
9830
9831                    ;PERR30         TRAP
9832                    ;               BAD DATA IS POINTED TO BY -16(SP)
9833                    ;               GOOD DATA IS IN R1
9834
9835                    ;PERR31         TRAP
9836                    ;               SPECIAL ECC FAILURE HANDLER
9837
9838                    ;PERR32         TRAP
9839                    ;               SPECIAL ECC FAILURE HANDLER
9840
9841                    ;PERR33         TRAP
9842                    ;               SPECIAL ECC FAILURE HANDLER
9843
9844                    ;PERR34         TRAP
9845                    ;               SPECIAL ECC FAILURE HANDLER
9846
9847                    ;PERR35         TRAP
9848                    ;               SPECIAL BRANCH GOBBLE FAILURE HANDLER
9849
9850                    ;       CALLING SEQUENCE FOR TRAP TYPES
9851                    ;       BEQ     2$              ;NO - ERROR,BRANCH FOR CARD
9852                    ;       PERRXX                  ;TRAP TO ERROR ROUTINE
9853                    ;2$:    NEXT    INSTRUCTION     ;CONTINUE TESTING
```

```
 9856 052474  010437  002032          $PER01: MOV     R4,ADDRESS
 9857 052500  162737  000002  002032          SUB     #2,ADDRESS
 9858 052506  010037  002050                  MOV     R0,BAD
 9859 052512  010537  002042                  MOV     R5,GOOD
 9860 052516  000137  053230                  JMP PERRAW
 9861
 9862 052522  010437  002032          $PER02: MOV     R4,ADDRESS
 9863 052526  162737  000002  002032          SUB     #2,ADDRESS
 9864 052534  010137  002050                  MOV     R1,BAD
 9865 052540  010237  002042                  MOV     R2,GOOD
 9866 052544  000137  053230                  JMP PERRAW
 9867
 9868 052550  010137  002032          $PER03: MOV     R1,ADDRESS
 9869 052554  162737  000002  002032          SUB     #2,ADDRESS
 9870 052562  010437  002042                  MOV     R4,GOOD
 9871 052566  016137  177776  002050          MOV     -2(R1),BAD
 9872 052574  000137  053230                  JMP PERRAW
 9873
 9874 052600  010037  002032          $PER04: MOV     R0,ADDRESS
 9875 052604  162737  000002  002032          SUB     #2,ADDRESS
 9876 052612  010437  002050                  MOV     R4,BAD
 9877 052616  010237  002042                  MOV     R2,GOOD
 9878 052622  000137  053230                  JMP     PERRAW
 9879
 9880 052626  010237  002042          PERR05: MOV     R2,GOOD
 9881 052632  014037  002050          PERA05: MOV     -(R0),BAD
 9882 052636  010037  002032                  MOV     R0,ADDRESS
 9883 052642  062700  000002                  ADD     #2,R0           ;RFSTORE R0
 9884 052646  004737  036626                  CALL    BADSTACK
 9885 052652  000207                          RETURN
 9886
 9887 052654  005037  002042          PERR06: CLR     GOOD
 9888 052660  000764                          BR      PERA05
 9889
 9890 052662  010137  002032          $PER07: MOV     R1,ADDRESS
 9891 052666  010237  002050                  MOV     R2,BAD
 9892 052672  013737  002234  002042          MOV     DATBUF,GOOD
 9893 052700  000137  053230                  JMP     PERRAW
 9894
 9895 052704                          $PER10: LET ADDRESS := R1 + #2
 9896 052716                                  LET BAD := R2
 9897 052722                                  LET GOOD := DATBUF+2
 9898 052730  000137  053230                  JMP     PERRAW
 9899
 9900 052734                          $PER11: LET ADDRESS := R1
 9901 052740                                  LET BAD := R0
 9902 052744                                  LET GOOD := #0
 9903 052750  000137  053302                  JMP     PERRAB
 9904
 9905 052754                          $PER12: LET ADDRESS := R1
 9906 052760                                  LET BAD := R0
 9907 052764                                  LET GOOD := #377
 9908 052772  000137  053302                  JMP     PERRAB
```

```
9911 052776                          $PER13: LET ADDRESS := R1
9912 053002                                  LET BAD := R0
9913 053006                                  LET GOOD := #0
9914 053012  000137  053230                  JMP     PERRAW
9915
9916 053016                          $PER14: LET ADDRESS := R1
9917 053022                                  LET BAD := R0
9918 053026                                  LET GOOD := ONES
9919 053034  000137  053230                  JMP     PERRAW
9920
9921 053040                          $PER15: LET ADDRESS := R1
9922 053044                                  LET BAD := R0
9923 053050                                  LET GOOD := TSTDAT
9924 053056  000137  053230                  JMP     PERRAW
9925
9926 053062                          $PER16: LET ADDRESS := R1
9927 053066                                  LET BAD := R0
9928 053072                                  LET GOOD := TSTDAT+2
9929 053100  000453                          BR      PERRAW
9930
9931 053102                          $PER17: LET ADDRESS := R1
9932 053106                                  LET BAD := R0
9933 053112                                  LET GOOD := R2
9934 053116  000444                          BR      PERRAW
9935
9936 053120                          $PER20: LET ADDRESS := R1
9937 053124                                  LET BAD := R0
9938 053130                                  LET GOOD := R3
9939 053134  000435                          BR      PERRAW
9940
9941 053136                          $PER21: LET ADDRESS := R1
9942 053142                                  LET BAD := R0
9943 053146                                  LET GOOD := #177
9944 053154  000477                          BR      PERRA7
9945
9946 053156                          $PER22: LET ADDRESS := R1
9947 053162                                  LET BAD := R2
9948 053166                                  LET GOOD := R0
9949 053172  000416                          BR      PERRAW
9950
9951 053174                          $PER23: LET ADDRESS := R1
9952 053200                                  LET BAD := R0
9953 053204                                  LET GOOD := R4
9954 053210  000407                          BR      PERRAW
9955
9956 053212                          $PER24: LET ADDRESS := R2
9957 053216                                  LET BAD := R0
9958 053222                                  LET GOOD := R3
9959 053226  000400                          BR      PERRAW
```

```
 9961 053230                            PERRAW: SUBTST  <<DATA WAS A WORD>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       DATA WAS A WORD
                                        ;*****************************************************************************
 9962 053230  004737  053464                       CALL    PERBNK
 9963 053234                                       IF ABORTFLAG IS TRUE THEN $CALL GETDATA
 9964 053246                                       IF BADPC EQ #0 THEN $CALL BADSTACK
 9965 053260  004737  053440                       CALL    PERXOR
 9966 053264                                       IF ABORTFLAG IS FALSE
 9967 053272  104011                                 ERROR +11
 9968 053274                                       ELSE
 9969 053276  104012                                 ERROR +12
 9970 053300                                       END ;OF IF ABORTFLAG
 9971 053300  000002                               RTI
 9972
 9973 053302                            PERRAB: SUBTST  <<DATA WAS A BYTE>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       DATA WAS A BYTE
                                        ;*****************************************************************************
 9974 053302  004737  053464                       CALL    PERBNK
 9975 053306                                       IF ABORTFLAG IS TRUE THEN $CALL GETDATA
 9976 053320                                       IF BADPC EQ #0 THEN $CALL BADSTACK
 9977 053332  004737  053440                       CALL    PERXOR
 9978 053336                                       IF ABORTFLAG IS FALSE
 9979 053344  104014                                 ERROR +14
 9980 053346                                       ELSE
 9981 053350  104015                                 ERROR +15
 9982 053352                                       END ;OF IF ABORTFLAG
 9983 053352  000002                               RTI
```

```
 9986 053354                                PERRA7: SUBTST  <<DATA WAS A 7 BIT BYTE>>
                                            ;*****************************************************************************
                                            ;*SUBTEST       DATA WAS A 7 BIT BYTE
                                            ;*****************************************************************************
 9987 053354                                        IF BADPC EQ #0 THEN $CALL BADSTACK
 9988 053366  004737  053440                        CALL    PERXOR
 9989 053372  004737  053464                        CALL    PERBNK
 9990 053376  104022                                ERROR   +22
 9991 053400  000002                                RTI
 9992
 9993 053402                                $PER26: LET GOOD2 := #100000
 9994 053410                                        LET GOOD3 := #100
 9995 053416  000137  050136                        JMP PERRA3
 9996
 9997 053422  005037  002044                $PER27: CLR     GOOD2
 9998 053426                                        LET GOOD3 := #077
 9999 053434  000137  050136                        JMP PERRA3
10000
10001 053440                                PERXOR: SUBTST  <<DETERMINE XOR OF GOOD & BAD>>
                                            ;*****************************************************************************
                                            ;*SUBTEST       DETERMINE XOR OF GOOD & BAD
                                            ;*****************************************************************************
10002 053440                                        PUSH    R0
10003 053442  013700  002042                        MOV     GOOD,R0
10004 053446  013737  002050  002056                MOV     BAD,BADXOR
10005 053454  074037  002056                        XOR     R0,BADXOR
10006 053460                                        POP     R0
10007 053462  000207                                RETURN
```

```
10010 053464                                PERBNK: SUBTST<<LOG ERROR ON BAD BANK>>
                                            ;*****************************************************************************
                                            ;*SUBTEST       LOG ERROR ON BAD BANK
                                            ;*****************************************************************************
10011                                               ;WHILE WE'RE HERE LET'S MARK THE BAD BANK IN THE CONFIGURATION TABLE
10012 053464                                        PUSH    R0,R1
10013 053470  013701  002100                        MOV     BANK,R1
10014 053474  006301                                ASL     R1
10015 053476  006301                                ASL     R1
10016 053500  052761  000001  002624                BIS     #BIT0,CONFIG(R1)
10017 053506  105261  002626                        INCB    CONFIG+2(R1)            ;BUMP BANK COUNTER
10018 053512  001002                                BNE     12$                     ;NO OVERFLOW - SKIP
10019 053514  105361  002626                        DECB    CONFIG+2(R1)            ;SET BACK TO 255.
10020 053520  126137  002626  002524  12$:        CMPB    CONFIG+2(R1),ERRMAX     ;IS IT PAST MAX?
10021 053526  101403                                BLOS    11$                     ;NO - SKIP
10022 053530                                        SET     TOOMANY                 ;YES
10023 053536                                11$:    POP     R1,R0
10024 053542  000207                                RETURN
10025
10026 053544  010037  002050                PERECC: MOV     R0,BAD
10027 053550                                        IF ADDRESS EQ TESTADD
10028 053560  013737  002240  002042                  MOV TSTDAT,GOOD
10029 053566                                        ELSE
10030 053570  013737  002242  002042                  MOV   TSTDAT+2,GOOD
10031 053576                                        END ;OF IF (R1)
10032 053576  004737  053440                        CALL    PERXOR
10033 053602                                        SET     HEADER
10034 053610  000207                                RETURN
10035
10036 053612                                $PER31: IF BADPC EQ #0 THEN $CALL BADSTACK
10037 053624  004737  053544                        CALL    PERECC
10038 053630                                        IF REALPAT EQ #11
10039 053640  104037                                  ERROR +37
10040 053642                                        END ;OF IF REALPAT
10041 053642                                        IF REALPAT EQ #14
10042 053652  104042                                  ERROR +42
10043 053654                                        END ;OF IF REALPAT
10044 053654                                        IF REALPAT EQ #15
10045 053664  104043                                  ERROR +43
10046 053666                                        END ;OF IF REALPAT
10047 053666                                        IF REALPAT EQ #16
10048 053676  104044                                  ERROR +44
10049 053700                                        END ;OF IF REALPAT
10050 053700                                        SET HEADER
10051 053706  000002                                RTI
```

```
10054 053710                                $PER32: IF BADPC EQ #0 THEN $CALL BADSTACK
10055 053722  010137  002032                        MOV     R1,ADDRESS
10056 053726  010037  002050                        MOV     R0,BAD
10057 053732  010237  002042                        MOV     R2,GOOD
10058 053736                                        SET     HEADER
10059 053744  104040                                ERROR   +40
10060 053746                                        SET     HEADER
10061 053754  000002                                RTI
10062
10063 053756                                $PER33: IF BADPC EQ #0 THEN $CALL BADSTACK
10064 053770  010137  002032                        MOV     R1,ADDRESS
10065 053774  010037  002050                        MOV     R0,BAD
10066 054000  105037  002051                        CLRB    BAD+1
10067 054004  012737  000377  002042                MOV     #377,GOOD
10068 054012  004737  053440                        CALL    PERXOR
10069 054016                                        SET     HEADER
10070 054024  104041                                ERROR   +41
10071 054026                                        SET     HEADER
10072 054034  000002                                RTI
10073
10074 054036                                $PER34:IF BADPC EQ #0 THEN $CALL BADSTACK
10075 054050                                        IF #BIT15!BIT4 OFF.IN CSR
10076 054060  104016                                  ERROR +16                     ;NO SBE OR DBE
10077 054062                                        ELSE
10078 054064  104001                                  ERROR +1                      ;EXPECTED SBE SO DBE MUST HAVE GOTTEN SET
10079 054066                                        END ;OF IF #BIT15.BIT4
10080 054066  000002                                RTI
10081
10082                                               ;DURING BRANCH GOBBLE THE CONDITION CODES WERE WRONG
10083 054070  004737  053464                $PER35: CALL    PERBNK
10084 054074  004737  036626                        CALL    BADSTACK
10085 054100  013737  002030  002050                MOV     BADPSW,BAD
10086 054106  012737  000012  002042                MOV     #12,GOOD
10087 054114  104047                                ERROR   +47
10088 054116  062706  000004                        ADD     #4,SP                 ;FIX STACK FROM TRAP
10089 054122  000207                                RETURN                        ;ABORTING TEST
10090
10091 054124  010037  002042                $PER36: MOV     R0,GOOD
10092 054130  010137  002050                        MOV     R1,BAD
10093 054134                                        SET     HEADER
10094 054142  104023                                ERROR   +23
10095 054144                                        SET     HEADER
10096 054152  000002                                RTI
```

```
10099                                   .SBTTL  ROUTINE SCOPE HANDLER
10100                                ;**********************************************************************************
10101                                ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
10102                                ;*AND LOAD THE DISPLAY DATA INTO THE DISPLAY REGISTER
10103                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
10104                                ;*SW14=1        LOOP ON TEST
10105                                ;*SW9=1 LOOP ON ERROR
10106                                ;*CALL
10107                                ;*      SCOPE           ;;SCOPE=IOT
10108 054154  005237  061266          $SCOPE: INC     $DEVCT          ;TELL APT WE ARE ALIVE
10109 054160                                  IF RESULT IS LT
10110 054162  005037  061266                    CLR   $DEVCT
10111 054166  105237  061270                    INCB  $UNIT
10112 054172                                  END ;OF IF RESULT
10113 054172  104410                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
10122 054174                                  IF STOPOK IS TRUE AND #SW8 SET.IN @SWR
10123 054212  005037  002370                    CLR   STOPOK
10124 054216  000137  043534                    JMP   EXIT
10125 054222                                  END ;OF IF STOPOK
10126 054222                                  IF NOSCOPE IS TRUE
10127 054230  000002                            RTI
10128 054232                                  END ;OF IF NOSCOPE
10129 054232                          1$:     IF #SW14 SET.IN @SWR THEN GOTO $OVER
10130                                 ;#####START OF CODE FOR THE XOR TESTER#####
10131 054242  000424                  $XTSTR: BR      2$              ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
10132                                                                 ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
10133 054244  013746  000004                  MOV     ERRVEC,-(SP)    ;;SAVE THE CONTENTS OF THE ERROR VECTOR
10134 054250  012737  054270  000004          MOV     #1$,ERRVEC      ;;SET FOR TIMEOUT
10135 054256  005737  177060                  TST     177060  ;;TIME OUT ON XOR?
10136 054262  012637  000004                  MOV     (SP)+,ERRVEC    ;;RESTORE THE ERROR VECTOR
10137 054266  000427                          BR      $SVLAD          ;;GO TO THE NEXT TEST
10138 054270  062706  000004          1$:     ADD     #4,SP           ;FIX STACK FROM TRAP
10139 054274  005737  002424                  TST     NO22BIT         ;IS THIS AN 11/44?
10140 054300  001002                          BNE     6$              ;BRANCH IF NOT
10141 054302  005037  177766                  CLR     CPUERR          ;RESET CPU ERROR REGISTER
10142 054306  012637  000004          6$:     MOV     (SP)+,ERRVEC    ;;RESTORE THE ERROR VECTOR
10143 054312  000407                          BR      4$              ;;LOOP ON THE PRESENT TEST
10144 054314                          2$:;#####END OF CODE FOR THE XOR TESTER#####
10145 054314  105737  002012          3$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
10146 054320  001412                          BEQ     $SVLAD          ;;BR IF NO
10147 054322  032777  001000  126246          BIT     #SW9,@SWR       ;;LOOP ON ERROR?
10148 054330  001404                          BEQ     5$              ;;BR IF NO
10149 054332  013737  002564  002562  4$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
10150 054340  000410                          BR      $OVER
10151 054342  105037  002012          5$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
10152 054346  011637  002562          $SVLAD: MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
10153 054352  011637  002564                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
10154 054356  005037  002332                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
10155 054362  004737  054374          $OVER:  CALL    GETDIS
10156 054366  013716  002562                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
10157 054372  000002                          RTI                     ;;FIXES PS
```

```
 10159 054374                           GETDIS: SUBTST  <<SUBR  DISPLAY>>
                                        ;*****************************************************************************
                                        ;*SUBTEST        SUBR    DISPLAY
                                        ;*****************************************************************************
 10160 054374  113737  002100  002011          MOVB    BANK,$BANK
 10161 054402  113737  002260  002010          MOVB    REALPAT,$PATMAR
 10162 054410                                  PUSH    R0
 10163 054412  005737  002124                  TST     RLFLAG                  ;ARE WE RELOCATED?
 10164 054416  001403                          BEQ     1$                      ;NO - SKIP
 10165 054420  052737  100000  002010          BIS     #BIT15,$PATMAR          ;YES - SET MSB
 10166 054426                           1$:
 10170 054426  013777  002010  126144          MOV     $PATMAR,@DISPLAY
 10171 054434  013737  002010  000174          MOV     $PATMAR,DISPREG         ;SOFTWARE DISPLAY REGISTER
 10172 054442                                  POP     R0
 10173 054444  000207                          RETURN
```

```
10176                                   .SBTTL  ROUTINE ERROR HANDLER
10177                                 ;*****************************************************************************
10178                                 ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
10179                                 ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
10180                                 ;*AND GO TO $ERRTYP ON ERROR
10181                                 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
10182                                 ;*SW15=1        HALT ON ERROR
10183                                 ;*SW13=1        INHIBIT ERROR TYPEOUTS
10184                                 ;*SW10=1        BELL ON ERROR
10185                                 ;*SW9=1 LOOP ON ERROR
10186                                 ;*CALL
10187                                 ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
10188
10189                                   .ENABL  LSB
10190 054446                          $ERROR: IF NOERROR IS FALSE
10191 054454  104410                      CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
10192 054456  105237  002012          1$:   INCB  $ERFLG            ;;SET THE ERROR FLAG
10193 054462  001775                      BEQ   1$                ;;DON'T LET THE FLAG GO TO ZERO
10194 054464  004737  054374              CALL  GETDIS            ;SETUP DISPLAY STUFF
10195 054470  013737  002010  061262      MOV   $PATMAR,$TESTN    ;FOR APT
10196 054476  032777  002000  126072      BIT   #SW10,@SWR        ;;BELL ON ERROR?
10197 054504  001404                      BEQ   2$                ;;NO - SKIP
10198 054506                              TYPE  $BELL             ;;RING BELL
10199 054512                              TYPE  MSG014            ;CONTROL Z
10200 054516  005237  002570          2$:   INC   $ERTTL            ;;COUNT THE NUMBER OF ERRORS
10201 054522                              IF RESULT IS MI
10202 054524  012737  077777  002570          MOV #77777,$ERTTL
10203 054532                              END ;OF IF RESULT
10204 054532                          END ;OF IF NOERROR
10205 054532  011637  002016          MOV     (SP),ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
10206 054536  162737  000002  002016  SUB     #2,ERRPC
10207 054544  010637  002022          MOV     SP,ERRSP
10208 054550  016637  000002  002026  MOV     2(SP),ERRPSW
10209 054556  117737  125234  002013  MOVB    @ERRPC,$ITEMB   ;;STRIP AND SAVE THE ERROR ITEM CODE
10210 054564                          IF NOERROR IS FALSE
10211 054572                            IF BADPC NE #0
10212 054600  013737  002020  002016      MOV BADPC,ERRPC
10213 054606  162737  000002  002016      SUB #2,ERRPC
10214 054614  013737  002024  002022      MOV BADSP,ERRSP
10215 054622  013737  002030  002026      MOV BADPSW,ERRPSW
10216 054630  005037  002020              CLR BADPC
10217 054634                            END ;IF
10218 054634  013737  002016  061260      MOV   ERRPC,$FATAL    ;FOR APT
10219 054642                            IF #SW13 SET.IN @SWR
10220 054652  000412                      BR  3$
10221 054654                            END ;OF IF #SW13
10227 054654                            IF #SW5 SET.IN @SWR AND TOOMANY IS TRUE
10228 054672                              GOTO 3$
10229 054674                            END ;OF IF #SW5
10230 054674                          END ;OF IF NOERROR
10231 054674  004737  055110          CALL    $ERRTYP         ;;GO TO USER ERROR ROUTINE
```

```
10233 054700                                3$:     IF NOERROR IS FALSE
10234 054706  005777  125664                         TST   @SWR              ;;HALT ON ERROR
10235 054712  100002                                BPL   7$                ;;SKIP IF CONTINUE
10236 054714  000000                        $HALT:  HALT                    ;;HALT ON ERROR!
10237 054716  104410                                CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
10238 054720                                7$:     IF NOSCOPE IS FALSE AND #SW9 SET.IN @SWR
10239 054736  013716  002564                           MOV $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
10240 054742                                        END ;OF IF NOSCOPE
10241 054742  005737  002332                        TST   $ESCAPE           ;;CHECK FOR AN ESCAPE ADDRESS
10242 054746  001402                                BEQ   9$                ;;BR IF NONE
10243 054750  013716  002332                        MOV   $ESCAPE,(SP)      ;;FUDGE RETURN ADDRESS FOR ESCAPE
10244 054754                                9$:     IF DETFLAG IS FALSE
10245 054762  005737  002424                          TST NO22BIT
10246 054766  001002                                  BNE 11$
10247 054770  005037  177766                          CLR CPUERR
10248 054774                                      11$: IF ACTFLAG IS TRUE OR APTFLAG IS TRUE OR FATAL$ IS TRUE
10249 055016  012737  000001  061256                    MOV     #1,$MSGTY       ;FOR APT
10250 055024  000137  043534                          JMP     EXIT
10251 055030                                        END ;OF IF ACTFLAG
10252 055030                                        IF XXDPCHAIN IS TRUE AND $ERTTL HI #20
10253 055046                                          TYPE    MSG066          ;ERROR COUNT EXCEEDED 20 - ABORTING FOR XXDP CHAIN
10254 055052  013700  000042                        MOV     42,R0
10255 055056  005037  000042                        CLR     42
10256 055062  000137  013306                        JMP     $ZAP42
10257 055066                                        END ;OF IF XXDPCHAIN
10258 055066                                      END ;OF IF DETFLAG
10259 055066                                    ELSE
10260 055070                                      SET   HEADER
10261 055076                                    END ;OF IF NOERROR
10262 055076                                10$:    CLEAR   TOOMANY,NOERROR
10263 055106  000002                                RTI                     ;;RETURN
10264                                               .DSABL  LSB
```

```
                                .SBTTL  ROUTINE ERROR MESSAGE TYPEOUT

                        ;*******************************************************************************
                        ;*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 APPROPRIATF INFORMATION CONCERNING THE ERROR.

                        $ERRTYP:SAVREG
                                TYPE    $CRLF           ;;''CARRIAGE RETURN'' & 'LINE FEED''
                                CLR     R0              ;;PICKUP THE ITEM INDEX
                                BISB    $ITEMB,R0
                                BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                        ;;TYPE THE PC OF THE ERROR
                                TYPOCT  ERRPC,<ERROR ADDRESS>
                                BR      11$             ;;GET OUT
                        1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
                                ASL     R0              ;;      WORK FOR THE ERROR TABLE
                                ASL     R0
                                ASL     R0
                                ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
                                MOV     (R0)+,3$        ;;PICKUP 'ERROR MESSAGE'' POINTER
                                BEQ     4$              ;;SKIP TYPEOUT IF NO POINTER
                                TST     NOERROR         ;IS THIS REALLY AN ERROR?
                                BNE     12$             ;YES - SKIP
                                TST     HEADER          ;TYPE HEADER?
                                BPL     4$              ;NO - SKIP
                        12$:    TST     FATAL$          ;WAS IT A FATAL ERROR?
                                BEQ     2$              ;NO - SKIP
                                TYPE    MSG067          ;FATAL
                        2$:     TYPE                    ;;TYPE THE 'ERROR MESSAGE''
                        3$:     .WORD   0               ;;''ERROR MESSAGE'' POINTER GOES HERE
                                TYPE    $CRLF           ;;''CARRIAGE RETURN'' & 'LINE FEED''
                        4$:     MOV     (R0)+,5$        ;;PICKUP 'DATA HEADER'' POINTER
                                BEQ     6$              ;;SKIP TYPEOUT IF 0
                                TST     NOERROR         ;IS THIS REALLY AN ERROR?
                                BNE     13$             ;YES - SKIP
                                TST     HEADER          ;TYPE HEADER?
                                BPL     6$              ;NO - SKIP
                        13$:    TYPE                    ;;TYPE THE 'DATA HEADER''
                        5$:     .WORD   0               ;;''DATA HEADER'' POINTER GOES HERE
                                TYPE    $CRLF           ;;''CARRIAGE RETURN'' & 'LINE FEED''
                        6$:     MOV     (R0)+,R1        ;;PICKUP 'DATA TABLE'' POINTER
                                BEQ     10$             ;;BR IF NO DATA TO BE TYPED
                                MOV     (R0)+,R2        ;;PICKUP 'DATA FORMAT'' POINTER
```

```
10313 055256  112203                7$:     MOVB    (R2)+,R3
10314 055260  006303                        ASL     R3              ;MAKE IT A WORD ADDRESS
10315 055262  004773  055270                CALL    @8$(R3)
10316 055266  000412                        BR      9$
10317 055270  055404                8$:     TAG70$
10318 055272  055414                        TAG71$
10319 055274  055424                        TAG72$
10320 055276  055474                        TAG73$
10321 055300  055534                        TAG74$
10322 055302  055546                        TAG75$
10323 055304  055560                        TAG76$
10324 055306  055624                        TAG77$
10325 055310  055632                        TAG78$
10326 055312  055712                        TAG79$
10331 055314  062701  000002          9$:     ADD     #2,R1           ;UPDATE DATA TABLE POINTER
10332 055320  005711                        TST     (R1)            ;;IS THERE ANOTHER NUMBER?
10333 055322  001403                        BEQ     10$             ;;BR IF NO
10334 055324                                TYPE    MSG018          ;TYPE 2 SPACES
10335 055330  000752                        BR      7$              ;;LOOP
10336
10337 055332  005737  002106          10$:    TST     MUT             ;IS THERE A MEMORY UNDER TEST
10338 055336  001402                        BEQ     11$             ;NO - SKIP
10339 055340  005237  002552                INC     HEADER          ;YES - BUMP HEADER FLAG
10340 055344  104416                11$:    RESREG
10341 055346                                IF #SW7 SET.IN @SWR AND DETFLAG IS FALSE AND NOERROR IS FALSE
10342 055372  004737  055734                  CALL  DETAIL
10343 055376                                END ;OF IF #SW7
10344 055376                                TYPE    MSG104          ;CONTROL Z
10345 055402  000207                        RETURN
```

```
10348                                   ;*****************************************************************************
10349                                   ;*** OCTAL ***
10350                                   ;*****************************************************************************
10351 055404                            TAG70$: TYPOCT  @(R1)           ;;TYPE AN OCTAL NUMBER
10352 055412  000207                            RETURN
10353
10354                                   ;*****************************************************************************
10355                                   ;*** DECIMAL ***
10356                                   ;*****************************************************************************
10357 055414                            TAG71$: TYPDEC  @(R1)           ;;TYPE A DECIMAL NUMBER
10358 055422  000207                            RETURN
10359
10360                                   ;*****************************************************************************
10361                                   ;*** INTERLEAVE ***
10362                                   ;*****************************************************************************
10363 055424                            TAG72$: PUSH    R1,R5
10364 055430  013701  002100                    MOV     BANK,R1
10365 055434  070127  000004                    MUL     #4,R1
10366 055440                                    SET     NOTAB                   ;INDICATE NO TABLE TO BE PRINTED - NOW
10367 055446                                    TYPE    MSG014
10368 055452  004737  035714                    CALL    TCFIG1
10369 055456  005037  002342                    CLR     NOTAB
10370 055462                                    POP     R5,R1
10371 055466                                    TYPE    MSG014                  ;1 SPACE
10372 055472  000207                            RETURN
10373
10374                                   ;*****************************************************************************
10375                                   ;*** CSR ***
10376                                   ;*****************************************************************************
10377 055474                            TAG73$: PUSH    R1,R5
10378 055500  013701  002100                    MOV     BANK,R1
10379 055504  070127  000004                    MUL     #4,R1
10380 055510                                    SET     NOTAB
10381 055516  004737  036234                    CALL    TCFIG3
10382 055522  005037  002342                    CLR     NOTAB
10383 055526                                    POP     R5,R1
10384 055532  000207                            RETURN
10385
10386                                   ;*****************************************************************************
10387                                   ;*** PATTERN ***
10388                                   ;*****************************************************************************
10389 055534                            TAG74$: TYPOCS  REALPAT,<TYPE (0-77)>,2,Z
10390 055544  000207                            RETURN
10391
10392                                   ;*****************************************************************************
10393                                   ;*** BANK ***
10394                                   ;*****************************************************************************
10395 055546                            TAG75$: TYPOCS  BANK,<TYPE (0-167)>,3
10396 055556  000207                            RETURN
```

```
10398                                   ;*****************************************************************************
10399                                   ;*** MTYPE ***
10400                                   ;*****************************************************************************
10401 055560                            TAG76$: PUSH    R1,R5
10402 055564  013701  002100                    MOV     BANK,R1
10403 055570  070127  000004                    MUL     #4,R1
10404 055574                                    SET     NOTAB
10405 055602                                    TYPE    MSG019
10406 055606  004737  036046                    CALL    TCFIG2
10407 055612  005037  002342                    CLR     NOTAB
10408 055616                                    POP     R5,R1
10409 055622  000207                            RETURN
10410
10411                                   ;*****************************************************************************
10412                                   ;*** UNKNOWN DATA ***
10413                                   ;*****************************************************************************
10414 055624                            TAG77$: TYPE    MSG061
10415 055630  000207                            RETURN
10416
10417                                   ;*****************************************************************************
10418                                   ;*** PHYSICAL ADDRESS ***
10419                                   ;*****************************************************************************
10420 055632  013737  002032  002036  TAG78$: MOV     ADDRESS,PHYADD
10421 055640  162737  060000  002036          SUB     #FIRST,PHYADD
10422 055646  013737  002100  002040          MOV     BANK,PHYADD+2
10423 055654  006237  002040                  ASR     PHYADD+2
10424 055660  103003                          BCC     1$
10425 055662  052737  100000  002036          BIS     #BIT15,PHYADD
10426 055670  012746  002036          1$:     MOV     #PHYADD,-(SP)           ;POINTER TO DOUBLE WORD ON STACK
10427 055674  004737  061136                  CALL    $DB2O                   ;CALL DOUBLE PRECISION CONVERSION ROUTINE
10428 055700  062706  000002                  ADD     #2,SP                   ;FIX STACK
10429 055704                                    TYPE    $OCT8
10430 055710  000207                            RETURN
10431
10432                                   ;*****************************************************************************
10433                                   ;*** OCTAL BYTE ***
10434                                   ;*****************************************************************************
10435 055712                            TAG79$: TYPE    MSG018          ;2 SPACES
10436 055716                                    TYPOCS  @(R1),<TYPE BYTE>,3,Z
10437 055726                                    TYPE    MSG014          ;SPACE
10438 055732  000207                            RETURN
```

```
10482 055734                                DETAIL: SUBTST  <<SUBR  DETAILED ERROR REPORT>>
                                            ;*****************************************************************************
                                            ;*SUBTEST       SUBR    DETAILED ERROR REPORT
                                            ;*****************************************************************************
10483 055734  005237  002212                        INC     DETFLAG
10484 055740  022737  000003  002212                CMP     #3,DETFLAG
10485 055746  101472                                BLOS    4$
10486 055750  022737  000002  002212                CMP     #2,DETFLAG
10487 055756  001435                                BEQ     2$
10488 055760                                        PUSH    HEADER,MUT
10489 055770                                        SET     HEADER
10490 055776  005037  002106                        CLR     MUT
10491 056002  010037  002172                        MOV     R0,DETR0
10492 056006  012700  002174                        MOV     #DETR1,R0
10493 056012  010120                                MOV     R1,(R0)+
10494 056014  010220                                MOV     R2,(R0)+
10495 056016  010320                                MOV     R3,(R0)+
10496 056020  010420                                MOV     R4,(R0)+
10497 056022  010520                                MOV     R5,(R0)+
10498 056024  013720  002022                        MOV     ERRSP,(R0)+
10499 056030  013720  002026                        MOV     ERRPSW,(R0)+
10500 056034  013700  002172                        MOV     DETR0,R0
10501 056040                                        SET     NOERROR
10502 056046  104013                                ERROR   +13
10503 056050  000422                                BR      1$
10504 056052                                2$:     PUSH    HEADER,MUT
10505 056062                                        SET     HEADER
10506 056070  005037  002106                        CLR     MUT
10507 056074                                        SET     NOERROR
10508 056102  104031                                ERROR   +31
10509 056104  005737  002424                        TST     NO22BIT
10510 056110  001002                                BNE     1$
10511 056112  005037  177766                        CLR     CPUERR
10512 056116                                1$:     POP     MUT,HEADER
10513                                               ;WARNING RECURSIVE
10514 056126  004737  055734                        CALL    DETAIL
10515 056132  000207                                RETURN
```

```
10518                                           ;SIMULATE CONTROL 'T'
10519 056134  004737  057572          4$:       CALL    CONTT                   ;DISPLAY 'DISPLAY'' INFO
10520
10521                                           ;TYPE CONTENTS OF ALL CSR'S
10522 056140                                    PUSH    CSR,CSRNO,R1
10523 056152                                    TYPE    MSG058
10524 056156                                    TYPE    $CRLF
10525 056162  013701  002216                    MOV     TOTCSRS,R1
10526 056166                                    BEGIN DUMPCSRLOOP
10527 056166                                      FOR CSRNO := #0 TO #36 BY #2
10528 056172  006301                                ASL       R1
10529 056174                                        ON.ERROR
10530 056176  104426                                  READCSR
10531 056200                                          TYPOCT    CSR
10532 056206                                          TYPE      MSG018          ;2 SPACES
10533 056212                                        END ;OF ON.ERROR
10534 056212                                        IF R1 EQ #0 THEN LEAVE DUMPCSRLOOP
10535 056216                                      END ;OF FOR CSRNO
10536 056234                                    END DUMPCSRLOOP
10537 056234                                    POP     R1,CSRNO,CSR
10538
10539                                           ;TYPE STACKS
10540 056246                                    PUSH    R0,R1
10541 056252                                    TYPE    MSG088          ;KERNEL STACK
10542 056256  013701  002534                    MOV     KSTACK,R1
10543 056262  162701  000002                    SUB     #2,R1
10544 056266                                    FOR     R0 := SP TO R1 BY #2
10545 056270                                      TYPE  $CRLF
10546 056274                                      TYPOCT        R0
10547 056300                                      TYPE          MSG018          ;2 SPACES
10548 056304                                      TYPOCT        (R0)
10549 056310                                    END ;OF FOR R0
10550                                           ;SET PREVIOUS MODE TO SUPERVISOR
10551 056320  005737  002426                    TST     NOSUPER
10552 056324  001036                            BNE     DET1
10553 056326  042737  030000  177776            BIC     #BIT13!BIT12,PSW
10554 056334  052737  010000  177776            BIS     #BIT12,PSW
10555 056342  006506                            MFPI    SSP
10556 056344                                    POP     R1,R0
10557 056350                                    TYPE    MSG089          ;SUPERVISOR STACK
10558 056354                                    IF R0 LT #SUPSTK
10559 056362                                      FOR R0 := R0 TO #SUPSTK-2 BY #2
10560 056362                                        TYPE        $CRLF
10561 056366                                        TYPOCT      R0
10562 056372                                        TYPE        MSG018          ;2 SPACES
10563 056376                                        TYPOCT      (R0)
10564 056402                                      END ;OF FOR R0
10565 056414                                    ELSE
10566 056416                                      TYPE  MSG091          ;IS EMPTY
10567 056422                                    END ;OF IF R0
10568                                           ;SET PREVIOUS MODE TO USER
10569 056422  052737  030000  177776  DET1:     BIS     #BIT13!BIT12,PSW
10570 056430  006506                            MFPI    USP
10571 056432                                    POP     R0
10572 056434                                    TYPE    MSG090          ;USER STACK
10573 056440                                    IF R0 LT #USESTK
10574 056446                                      FOR R0 := R0 TO #USESTK-2 BY #2
```

```
10575 056446                                        TYPE        $CRLF
10576 056452                                        TYPOCT      R0
10577 056456                                        TYPE        MSG018          ;2 SPACES
10578 056462                                        TYPOCT      (R0)
10579 056466                                      END ;OF FOR R0
10580 056500                                    ELSE
10581 056502                                      TYPE  MSG091          ;IS EMPTY
10582 056506                                    END ;OF IF R0
10583 056506                                    TYPE    $CRLF
10584 056512  005037  002212                      CLR     DETFLAG
10585 056516                                    POP     R0
10586 056520  000207                            RETURN
```

```
                                        .SBTTL  ROUTINE 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

056522  017646  000000         $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
056526  116637  000001  056745         MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
056534  112637  056747                 MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
056540  062716  000002                 ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
056544  000406                         BR      $TYPON
056546  112737  000001  056745 $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
056554  112737  000006  056747         MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
056562  112737  000005  056744 $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
056570  010346                         MOV     R3,-(SP)        ;;SAVE R3
056572  010446                         MOV     R4,-(SP)        ;;SAVE R4
056574  010546                         MOV     R5,-(SP)        ;;SAVE R5
056576  113704  056747                 MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
056602  005404                         NEG     R4
056604  062704  000006                 ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
056610  110437  056746                 MOVB    R4,$OMODE       ;;SAVE IT FOR USE
056614  113704  056745                 MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
056620  016605  000012                 MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
056624  005003                         CLR     R3              ;;CLEAR THE OUTPUT WORD
056626  006105                 1$:     ROL     R5              ;;ROTATE MSB INTO "C"
056630  000404                         BR      3$              ;;GO DO MSB
056632  006105                 2$:     ROL     R5              ;;FORM THIS DIGIT
056634  006105                         ROL     R5
056636  006105                         ROL     R5
056640  010503                         MOV     R5,R3
056642  006103                 3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
056644  105337  056746                 DECB    $OMODE          ;;TYPE THIS DIGIT?
056650  100016                         BPL     6$              ;;BR IF NO
056652  042703  177770                 BIC     #177770,R3      ;;GET RID OF JUNK
056656  001002                         BNE     4$              ;;TEST FOR 0
056660  005704                         TST     R4              ;;SUPPRESS THIS 0?
056662  001403                         BEQ     5$              ;;BR IF YES
056664  005204                 4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
```

```
10681 056666  052703  000060              BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
10682 056672  052703  000040        5$:   BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
10683 056676  110337  056742              MOVB    R3,8$           ;;SAVE FOR TYPING
10684 056702                              TYPE    8$              ;;GO TYPE THIS DIGIT
10685 056706  105337  056744        6$:   DECB    $OCNT           ;;COUNT BY 1
10686 056712  003347                      BGT     2$              ;;BR IF MORE TO DO
10687 056714  002402                      BLT     7$              ;;BR IF DONE
10688 056716  005204                      INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
10689 056720  000744                      BR      2$              ;;GO DO THE LAST DIGIT
10690 056722  012605                7$:   MOV     (SP)+,R5        ;;RESTORE R5
10691 056724  012604                      MOV     (SP)+,R4        ;;RESTORE R4
10692 056726  012603                      MOV     (SP)+,R3        ;;RESTORE R3
10693 056730  016666  000002  000004      MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
10694 056736  012616                      MOV     (SP)+,(SP)
10695 056740  000002                      RTI                     ;;RETURN
10696 056742     000                8$:   .BYTE   0               ;;STORAGE FOR ASCII DIGIT
10697 056743     000                      .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
10698 056744     000             $OCNT:   .BYTE   0               ;;OCTAL DIGIT COUNTER
10699 056745     000             $OFILL:  .BYTE   0               ;;ZERO FILL SWITCH
10700 056746  000000             $OMODE:  .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
10702                                           .SBTTL  ROUTINE CONVERT BINARY TO DECIMAL AND TYPE
10703                                   ;*******************************************************************************
10704                                   ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
10705                                   ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
10706                                   ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
10707                                   ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
10708                                   ;*REPLACED WITH SPACES.
10709                                   ;*CALL:
10710                                   ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
10711                                   ;*      TYPDS                   ;;GO TO THE ROUTINE
10712 056750                           $TYPDS: PUSH    R0,R1,R2,R3,R5
10713 056762  012746  020200                   MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
10714 056766  016605  000020                   MOV     20(SP),R5       ;;GET THE INPUT NUMBER
10715 056772  100004                           BPL     1$              ;;BR IF INPUT IS POS.
10716 056774  005405                           NEG     R5              ;;MAKE THE BINARY NUMBER POS.
10717 056776  112766  000055  000001           MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
10718 057004  005000                   1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
10719 057006  012703  057164                   MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
10720 057012  112723  000040                   MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
10721 057016  005002                   2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
10722 057020  016001  057154                   MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
10723 057024  160105                   3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
10724 057026  002402                           BLT     4$              ;;BR IF DONE
10725 057030  005202                           INC     R2              ;;INCREASE THE BCD DIGIT BY 1
10726 057032  000774                           BR      3$
10727 057034  060105                   4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
10728 057036  005702                           TST     R2              ;;CHECK IF BCD DIGIT=0
10729 057040  001002                           BNE     5$              ;;FALL THROUGH IF 0
10730 057042  105716                           TSTB    (SP)            ;;STILL DOING LEADING 0'S?
10731 057044  100407                           BMI     7$              ;;BR IF YES
10732 057046  106316                   5$:     ASLB    (SP)            ;;MSD?
10733 057050  103003                           BCC     6$              ;;BR IF NO
10734 057052  116663  000001  177777           MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
10735 057060  052702  000060           6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
10736 057064  052702  000040           7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
10737 057070  110223                           MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
10738 057072  005720                           TST     (R0)+           ;;JUST INCREMENTING
10739 057074  020027  000010                   CMP     R0,#10          ;;CHECK THE TABLE INDEX
10740 057100  002746                           BLT     2$              ;;GO DO THE NEXT DIGIT
10741 057102  003002                           BGT     8$              ;;GO TO EXIT
10742 057104  010502                           MOV     R5,R2           ;;GET THE LSD
10743 057106  000764                           BR      6$              ;;GO CHANGE TO ASCII
10744 057110  105726                   8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
10745 057112  100003                           BPL     9$              ;;BR IF NO
10746 057114  116663  177777  177776           MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
10747 057122  105013                   9$:     CLRB    (R3)            ;;SET THE TERMINATOR
10748 057124                                   POP     R5,R3,R2,R1,R0
10749 057136                                   TYPE    $DBLK           ;;NOW TYPE THE NUMBER
10750 057142  016666  000002  000004           MOV     2(SP),4(SP)     ;;ADJUST THE STACK
10751 057150  012616                           MOV     (SP)+,(SP)
10752 057152  000002                           RTI                     ;;RETURN TO USER
10753 057154  023420                   $DTBL:  10000.
10754 057156  001750                           1000.
10755 057160  000144                           100.
10756 057162  000012                           10.
10757 057164  000000  000000  000000   $DBLK:  .WORD   0,0,0,0
      057172  000000
```

```
                                        .SBTTL  ROUTINE TTY INPUT
                                ;*******************************************************************************
                                ;*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.
                                        .ENABLE LSB
057174                          $CKSWR:
057174  105777  123402                  TSTB    @$TKS           ;;CHAR THERE?
057200  100135                         BPL     12$             ;;IF NO, DON'T WAIT AROUND
057202  117746  123376                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
057206  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
057212  022716  000006          CONTS1: CMP     #6,(SP)         ;IS IT CONTROL F?
057216  001002                         BNE     1$              ;NO SKIP
057220  004737  044004                  CALL    FIELDSERVICE
057224  022716  000024          1$:     CMP     #24,(SP)        ;IS IT CONTROL T?
057230  001002                         BNE     16$             ;NO - SKIP
057232  004737  057572                  CALL    CONTT           ;YES - CALL CONTROL T ROUTINE
057236  022716  000003          16$:    CMP     #3,(SP)         ;IS IT CONTROL C?
057242  001461                         BEQ     5$              ;YES EXIT *****NOTE***** STACK IS SCREWED UP!
057244  022716  000004                  CMP     #4,(SP)         ;IS IT CONTROL D?
057250  001002                         BNE     2$              ;NO - SKIP
057252  000137  062114          CONTP:  JMP     @#0.ODT         ;YES - GO TO ODT
057256  022716  000023          2$:     CMP     #23,(SP)        ;IS IT CONTROL S?
057262  001002                         BNE     17$             ;NO - SKIP
057264  004737  057646                  CALL    CONTS           ;YES - CALL CONTROL S ROUTINE
057270  022716  000013          17$:    CMP     #13,(SP)        ;IS IT CONTROL K?
057274  001005                         BNE     6$              ;NO - SKIP
057276                                 TYPE    $CNTLK          ;TYPE A ^K
057302  013706  002142                  MOV     CTLKVEC,SP      ;RESET KSP TO AFTER PATTERN EXEC ROUTINE
057306  000207                         RETURN                  ;RETURN TO PATTERN EXEC ROUTINE
057310  022737  000176  002576  6$:     CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
057316  001067                         BNE     CKEND           ;;BRANCH IF NO
057320  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
057324  001064                         BNE     CKEND           ;;NO, RETURN TO USER
057326  005737  002060                  TST     $AUTO           ;ARE WE RUNNING IN AUTO-MODE?
057332  001061                         BNE     CKEND           ;BRANCH IF YES
057334                                 TYPE    $CNTLG          ;;ECHO THE CONTROL-G (^G)
057340                          $GTSWR: TYPE    $MSWR           ;;TYPE CURRENT CONTENTS
057344                                 TYPOCT  @SWR            ;;OF THE SWR
057352                                 TYPE    $MNEW           ;;PROMPT FOR NEW SWR
057356  005046                  3$:     CLR     -(SP)           ;;CLEAR COUNTER
057360  005046                         CLR     -(SP)           ;;THE NEW SWR
057362  105777  123214          4$:     TSTB    @$TKS           ;;CHAR THERE?
057366  100375                         BPL     4$              ;;IF NOT TRY AGAIN
057370  117746  123210                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
057374  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
057400  021627  000003                  CMP     (SP),#3         ;;IS IT A CONTROL-C?
057404  001006                         BNE     7$              ;;BRANCH IF NOT
057406                          5$:     TYPE    $CNTLC          ;;YES, ECHO CONTROL-C (^C)
057412  062706  000006                  ADD     #6,SP           ;;CLEAN UP STACK
057416  000137  043426                  JMP     BOOT            ;;CONTROL-C RESTART
057422  021627  000025          7$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
057426  001005                         BNE     9$              ;;BRANCH IF NOT
057430                                 TYPE    $CNTLU          ;;YES, ECHO CONTROL-U (^U)
057434  062706  000006          8$:     ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
057440  000746                         BR      3$              ;;LET'S TRY IT AGAIN
```

```
10821 057442  021627  000015          9$:     CMP     (SP),#15        ;;IS IT A <CR>?
10822 057446  001016                          BNE     13$             ;;BRANCH IF NO
10823 057450  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
10824 057454  001403                          BEQ     10$             ;;BRANCH IF YES
10825 057456  016677  000002  123112          MOV     2(SP),@SWR      ;;SAVE NEW SWR
10826 057464  062706  000006          10$:    ADD     #6,SP           ;;CLEAR UP STACK
10827 057470                                  TYPE    $CRLF           ;;ECHO <CR> AND <LF>
10828 057474  000002                  12$:    RTI                     ;;RETURN
10829 057476  062706  000002          CKEND:  ADD     #2,SP           ;FIX STACK
10830 057502  000002                          RTI                     ;RETURN
10831 057504  004737  052364          13$:    CALL    $TYPEC          ;;ECHO CHAR
10832 057510  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
10833 057514  002420                          BLT     15$             ;;BRANCH IF YES
10834 057516  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
10835 057522  003015                          BGT     15$             ;;BRANCH IF YES
10836 057524  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
10837 057530  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
10838 057534  001403                          BEQ     14$             ;;BRANCH IF YES
10839 057536  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
10840 057540  006316                          ASL     (SP)            ;;    CHAR OVER TO MAKE
10841 057542  006316                          ASL     (SP)            ;;    ROOM FOR NEW ONE
10842 057544  005266  000002          14$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
10843 057550  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
10844 057554  000702                          BR      4$              ;;GET THE NEXT ONE
10845 057556                          15$:    TYPE    $QUES           ;;TYPE ?<CR><LF>
10846 057562  000724                          BR      8$              ;;SIMULATE CONTROL-J
10847 057564     136     113     015  $CNTLK: .ASCIZ   /^K/<15><12>   ;CONTROL K ASCII STRING
      057567     012     000
10848                                         .EVEN
10849                                         .DSABL   LSB
```

```
10852 057572                            CONTT:  SUBTST  <<CONTROL T>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       CONTROL T
                                        ;*****************************************************************************
10853 057572                                    PUSH    R0
10854 057574                                    TYPE    $CRLF
10864 057600                                      IF RLFLAG IS TRUE
10865 057606                                        TYPE        MSG092          ;RELOCATED
10866 057612                                      END ;OF IF RLFLAG
10867 057612                                      TYPE  MSG093                  ;BANK=
10868 057616                                      TYPOCS        BANK,,3         ;TYPE 3 DIGITS
10869 057626                                      TYPE  MSG095                  ;PAT=
10870 057632                                      TYPOCS        REALPAT,,2      ;TYPE 2 DIGITS
10874 057642                                    POP     R0
10875 057644  000207                            RETURN
10876
10877 057646                            CONTS:  SUBTST  <<CONTROL S & CONTROL Q>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       CONTROL S & CONTROL Q
                                        ;*****************************************************************************
10878 057646                                    POP     R0              ;GET RID OF RETURN ADDRESS FROM STACK
10879 057650  105777  122726            CONTS2: TSTB    @$TKS           ;WAIT FOR CHARACTER
10880 057654  100375                            BPL     CONTS2
10881 057656  117716  122722                    MOVB    @$TKB,(SP)      ;REPLACE OVER OLD CHARACTER ON STACK
10882 057662  042716  177600                    BIC     #^C177,(SP)     ;STRIP ALL BUT ASCII
10883 057666                                    IF (SP) EQ #21          ;IF IT IS A CONTROL Q
10884 057674  000137  057212                      JMP   CONTS1
10885 057700                                    ELSE
10886 057702  000762                              BR    CONTS2
10887 057704                                    END ;OF IF (SP)
```

```
10889                                   ;*******************************************************************************
10890                                   ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
10891                                   ;*CALL:
10892                                   ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
10893                                   ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
10894                                   ;*                              ;;WITH PARITY BIT STRIPPED OFF
10895                                   ;
10896
10897 057704  011646                   $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
10898 057706  016666  000004  000002            MOV     4(SP),2(SP)     ;;SAVE THE PS
10899 057714  105777  122662           1$:     TSTB    @$TKS           ;;WAIT FOR
10900 057720  100375                           BPL     1$              ;;A CHARACTER
10901 057722  117766  122656  000004           MOVB    @$TKB,4(SP)     ;;READ THE TTY
10902 057730  042766  177600  000004           BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
10903 057736  026627  000004  000023           CMP     4(SP),#23       ;;IS IT A CONTROL-S?
10904 057744  001013                           BNE     3$              ;;BRANCH IF NO
10905 057746  105777  122630           2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
10906 057752  100375                           BPL     2$              ;;LOOP UNTIL ITS THERE
10907 057754  117746  122624                   MOVB    @$TKB,-(SP)     ;;GET CHARACTER
10908 057760  042716  177600                   BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
10909 057764  022627  000021                   CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
10910 057770  001366                           BNE     2$              ;;IF NOT DISCARD IT
10911 057772  000750                           BR      1$              ;;YES, RESUME
10912 057774  026627  000004  000140   3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
10913 060002  002407                           BLT     4$              ;;BRANCH IF YES
10914 060004  026627  000004  000175           CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
10915 060012  003003                           BGT     4$              ;;BRANCH IF YES
10916 060014  042766  000040  000004           BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
10917 060022  000002                   4$:     RTI                     ;;GO BACK TO USER
10918                                   ;*******************************************************************************
10919                                   ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
10920                                   ;*CALL:
10921                                   ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
10922                                   ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
10923                                   ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
10924 060024  010346                   $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
10925 060026  005046                           CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
10926 060030  012703  060322           1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
10927 060034  022703  060346           2$:     CMP     #$TTYIN+20.,R3  ;;BUFFER FULL?
10928 060040  101477                           BLOS    8$              ;;BR IF YES
10929 060042  104411                           RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
10930 060044  112613                           MOVB    (SP)+,(R3)      ;;GET CHARACTER
10931 060046  122713  000003                   CMPB    #3,(R3)         ;;IS IT A CONTROL-C?
10932 060052  001016                           BNE     3$              ;;BRANCH IF NO
10933 060054                                   TYPE    $CNTLC          ;;TYPE A CONTROL-C (^C)
10934 060060  005726                           TST     (SP)+           ;;CLEAN RUBOUT KEY OFF OF THE STACK
10935 060062  012603                           MOV     (SP)+,R3        ;;RESTORE R3
10936 060064  032777  000400  122504           BIT     #BIT8,@SWR      ;;IS THERE A HALT FLAG SET IN THE SWR?
10937 060072  001404                           BEQ     11$             ;;BRANCH IF NOT TO BOOT ROUTINE
10938 060074  005037  002370                   CLR     STOPOK          ;;GET READY TO HALT PROGRAM
10939 060100  000137  043534                   JMP     EXIT            ;;GO HALT PROGRAM
10940 060104  000137  043426           11$:    JMP     BOOT            ;;GOTO CONTROL-C RESTART
10941 060110  122713  000177           3$:     CMPB    #177,(R3)       ;;IS IT A RUBOUT
10942 060114  001022                           BNE     5$              ;;BR IF NO
10943 060116  005716                           TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
10944 060120  001007                           BNE     4$              ;;BR IF NO
10945 060122  112737  000134  060320           MOVB    #'\,10$         ;;TYPE A BACK SLASH
```

```
10946 060130                                        TYPE    10$
10947 060134  012716  177777                        MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
10948 060140  005303                        4$:     DEC     R3              ;;BACKUP BY ONE
10949 060142  020327  060322                        CMP     R3,#$TTYIN      ;;STACK EMPTY?
10950 060146  103434                                BLO     8$              ;;BR IF YES
10951 060150  111337  060320                        MOVB    (R3),10$                ;;SETUP TO TYPEOUT THE DELETED CHAR.
10952 060154                                        TYPE    10$             ;;GO TYPE
10953 060160  000725                                BR      2$              ;;GO READ ANOTHER CHAR.
10954 060162  005716                        5$:     TST     (SP)            ;;RUBOUT KEY SET?
10955 060164  001406                                BEQ     6$              ;;BR IF NO
10956 060166  112737  000134  060320                MOVB    #'\,10$         ;;TYPE A BACK SLASH
10957 060174                                        TYPE    10$
10958 060200  005016                                CLR     (SP)            ;;CLEAR THE RUBOUT KEY
10959 060202  122713  000025                6$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
10960 060206  001003                                BNE     7$              ;;BR IF NO
10961 060210                                        TYPE    $CNTLU          ;;TYPE A CONTROL 'U'
10962 060214  000705                                BR      1$              ;;GO START OVER
10963 060216  122713  000022                7$:     CMPB    #22,(R3)        ;;IS CHARACTER A '^R'?
10964 060222  001011                                BNE     9$              ;;BRANCH IF NO
10965 060224  105013                                CLRB    (R3)            ;;CLEAR THE CHARACTER
10966 060226                                        TYPE    $CRLF           ;;TYPE A 'CR' & 'LF'
10967 060232                                        TYPE    $TTYIN          ;;TYPE THE INPUT STRING
10968 060236  000676                                BR      2$              ;;GO PICKUP ANOTHER CHACTER
10969 060240                                8$:     TYPE    $QUES           ;;TYPE A '?'
10970 060244  000671                                BR      1$              ;;CLEAR THE BUFFER AND LOOP
10971 060246  111337  060320                9$:     MOVB    (R3),10$                ;;ECHO THE CHARACTER
10972 060252                                        TYPE    10$
10973 060256  122723  000015                        CMPB    #15,(R3)+       ;;CHECK FOR RETURN
10974 060262  001264                                BNE     2$              ;;LOOP IF NOT RETURN
10975 060264  105063  177777                        CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
10976 060270                                        TYPE    $LF             ;;TYPE A LINE FEED
10977 060274  005726                                TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
10978 060276  012603                                MOV     (SP)+,R3        ;;RESTORE R3
10979 060300  011646                                MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
10980 060302  016666  000004  000002                MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
10981 060310  012766  060322  000004                MOV     #$TTYIN,4(SP)
10982 060316  000002                                RTI                     ;;RETURN
10983 060320     000                    10$:        .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
10984 060321     000                                .BYTE   0               ;;TERMINATOR
10985 060322  000024                    $TTYIN:     .REPT   20.             ;;RESERVE SIZE BYTES FOR TTY INPUT
10988 060346     136     103     015    $CNTLC:     .ASCIZ  /^C/<15><12>    ;;CONTROL 'C'
      060351     012     000
10989 060353     136     125     015    $CNTLU:     .ASCIZ  /^U/<15><12>    ;;CONTROL 'U'
      060356     012     000
10990 060360     136     107     015    $CNTLG:     .ASCIZ  /^G/<15><12>    ;;CONTROL 'G'
      060363     012     000
10991 060365     015     012     123    $MSWR:      .ASCIZ  <15><12>/SWR = /
      060370     127     122     040
      060373     075     040     000
10992 060376     040     040     116    $MNEW:      .ASCIZ  /  NEW - /
      060401     105     127     040
      060404     075     040     000
10993                                               .EVEN
```

```
                                        .SBTTL  ROUTINE 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
011646                          $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
016666  000004  000002                  MOV     4(SP),2(SP)     ;;INPUT NUMBER
                                        PUSH    R0,R1,R2
104412                          1$:     RDLIN                   ;;READ AN ASCIZ LINE
012600                                  MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
010037  060536                          MOV     R0,5$           ;;AND SAVE IT
005001                                  CLR     R1              ;;CLEAR DATA WORD
005002                                  CLR     R2
112046                          2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
001420                                  BEQ     3$              ;;IF ZERO GET OUT
122716  000060                          CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
003026                                  BGT     4$              ;;IS AN OCTAL DIGIT
122716  000067                          CMPB    #'7,(SP)
002423                                  BLT     4$
006301                                  ASL     R1              ;;*2
006102                                  ROL     R2
006301                                  ASL     R1              ;;*4
006102                                  ROL     R2
006301                                  ASL     R1              ;;*8
006102                                  ROL     R2
042716  177770                          BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
062601                                  ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
000756                                  BR      2$              ;;LOOP
005726                          3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
010166  000012                          MOV     R1,12(SP)       ;;SAVE THE RESULT
010237  060556                          MOV     R2,$HIOCT
                                        POP     R2,R1,R0
000002                                  RTI                     ;;RETURN
005726                          4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
105010                                  CLRB    (R0)            ;;SET A TERMINATOR
                                        TYPE                    ;;TYPE UP THRU THE BAD CHAR.
000000                          5$:     .WORD   0
                                        TYPE    MSG062          ;INPUT MUST BE A
                                        TYPE    MSG063          ;N OCTAL
                                        TYPE    MSG064          ;NUMBER
000724                                  BR      1$              ;;TRY AGAIN
000000                          $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
                                        .SBTTL  ROUTINE 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
```

```
11052                                   ;*POSITIVE 32767 TO NEGATIVE 32768.
11053                                   ;*CALL:
11054                                   ;*      RDDEC                   ;;READ A DECIMAL NUMBER
11055                                   ;*      RETURN HERE             ;;NUMBER IS ON TOP OF THE STACK
11056                                   ;
11057
11058 060560  011646                   $RDDEC: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR
11059 060562  016666  000004  000002           MOV     4(SP),2(SP)     ;;THE INPUT NUMBER
11060 060570                                   PUSH    R0,R1,R2
11061 060576  104412                   1$:     RDLIN                   ;;READ AN ASCIZ LINE
11062 060600  012600                           MOV     (SP)+,R0        ;;ADDRESS OF 1ST CHAR.
11063 060602  010037  060726                   MOV     R0,6$           ;;SAVE INCASE OF BAD INPUT
11064 060606  005046                           CLR     -(SP)           ;;CLEAR DATA WORD
11065 060610  005002                           CLR     R2              ;;SIGN SET POSITIVE
11066 060612  122710  000055                   CMPB    #'-,(R0)        ;;SEE IF A MINUS SIGN WAS TYPED
11067 060616  001001                           BNE     2$              ;;BR IF NO MINUS SIGN
11068 060620  112002                           MOVB    (R0)+,R2        ;;SAVE FOR LATER USE
11069 060622  112001                   2$:     MOVB    (R0)+,R1        ;;PICKUP THIS CHARACTER
11070 060624  001424                           BEQ     3$              ;;GET OUT IF ZERO
11071 060626  122701  000060                   CMPB    #'0,R1          ;;MAKE SURE THIS CHARACTER
11072 060632  003032                           BGT     5$              ;;IS A DIGIT BETWEEN 0 & 9
11073 060634  122701  000071                   CMPB    #'9,R1
11074 060640  002427                           BLT     5$
11075 060642  032716  170000                   BIT     #^C7777,(SP)    ;;DON'T LET NUMBER GET TO BIG
11076 060646  001024                           BNE     5$              ;;BR IF NUMBER WOULD OVERFLOW
11077 060650  006316                           ASL     (SP)            ;;*2
11078 060652  011646                           MOV     (SP),-(SP)      ;;SAVE FOR LATER
11079 060654  006316                           ASL     (SP)            ;;*4
11080 060656  006316                           ASL     (SP)            ;;*8.
11081 060660  062616                           ADD     (SP)+,(SP)      ;;*10.
11082 060662  102416                           BVS     5$              ;;OVERFLOW ISN'T ALLOWED
11083 060664  162701  000060                   SUB     #'0,R1          ;;STRIP AWAY THE ASCII JUNK
11084 060670  060116                           ADD     R1,(SP)         ;;ADD IN THIS DIGIT
11085 060672  102412                           BVS     5$              ;;OVERFLOW ISN'T ALLOWED
11086 060674  000752                           BR      2$              ;;LOOP
11087 060676  005702                   3$:     TST     R2              ;;CHECK IF NUMBER IS NEG
11088 060700  001401                           BEQ     4$              ;;BR IF NO
11089 060702  005416                           NEG     (SP)            ;;YES--NEGATE THE NUMBER
11090 060704  012666  000012           4$:     MOV     (SP)+,12(SP)    ;;SAVE THE RESULT
11091 060710                                   POP     R2,R1,R0
11092 060716  000002                           RTI                     ;;RETURN
11093
11094 060720  005726                   5$:     TST     (SP)+           ;;CLEAN PARTIAL NUMBER FROM STACK
11095 060722  105010                           CLRB    (R0)            ;;SET A TERMINATOR
11096 060724                                   TYPE                    ;;TYPE THE INPUT UP TO BAD CHAR.
11097 060726  000000                   6$:     .WORD   0               ;;POINTER GOES HERE
11098 060730                                   TYPE    MSG062          ;INPUT MUSST BE A
11099 060734                                   TYPE    MSG065          ;DECIMAL
11100 060740                                   TYPE    MSG064          ;NUMBER
11101 060744  000714                           BR      1$              ;;TRY AGAIN
```

```
                                        .SBTTL  ROUTINE SAVE AND RESTORE R0-R5

                                ;*******************************************************************************
                                ;*SAVE R0-R5
                                ;*CALL:
                                ;*      SAVREG
                                ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
                                ;*
                                ;*TOP---(+16)
                                ;* +2---(+18)
                                ;* +4---R5
                                ;* +6---R4
                                ;* +8---R3
                                ;*+10---R2
                                ;*+12---R1
                                ;*+14---R0

060746                          $SAVREG:
060746                                  PUSH    R0,R1,R2,R3,R4,R5
060762  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
060766  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
060772  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
060776  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
061002  000002                          RTI

                                ;*RESTORE R0-R5
                                ;*CALL:
                                ;*      RESREG
061004                          $RESREG:
061004  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
061010  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
061014  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
061020  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
061024                                  POP     R5,R4,R3,R2,R1,R0
061040  000002                          RTI
```

```
                                        .SBTTL  ROUTINE RANDOM NUMBER GENERATOR

                                ;*************************************************************************************
                                ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
                                ;*WITH A RANGE OF 0 TO 2**(+33)-1.
                                ;*CALL:
                                ;*      CALL    $RAND           ;;CALL THE ROUTINE
                                ;*      RETURN                  ;;RETURN HERE THE RANDOM
                                ;*                              ;;NUMBER WILL BE IN
                                ;*                              ;;$HINUM,$LONUM

061042                          $RAND:  PUSH    R0,R1,R2
061050  013700  002544                  MOV     SEEDLO,R0       ;SET R0 WITH LOW
061054  013701  002542                  MOV     SEEDHI,R1       ;SET R1 WITH HIGH
061060  012702  000007                  MOV     #7,R2           ;SET SHIFT COUNT
061064  006300                  1$:     ASL     R0              ;;SHIFT R0 LEFT AND
061066  006101                          ROL     R1              ;;ROTATE CARRY INTO R1 AND
061070  077203                          SOB     R2,1$
061072  063700  002544                  ADD     SEEDLO,R0       ;ADD NUMBER TO MAKE X 129
061076  005501                          ADC     R1              ;;PROPOGATE CARRY
061100  063701  002542                  ADD     SEEDHI,R1       ;ADD NUMBER TO MAKE X 129
061104  062700  001057                  ADD     #1057,R0        ;;ADD LOW CONSTANT
061110  005501                          ADC     R1              ;;PROPOGATE CARRY
061112  062701  047401                  ADD     #47401,R1       ;;ADD HIGH CONSTANT
061116  010037  002544                  MOV     R0,SEEDLO       ;SAVE R0
061122  010137  002542                  MOV     R1,SEEDHI       ;SAVE R1
061126                                  POP     R2,R1,R0
061134  000207                          RETURN
```

```
                                        .SBTTL  ROUTINE DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT
                                ;*******************************************************************************
                                ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
                                ;*UNSIGNED OCTAL ASCIZ NUMBER.
                                ;*CALL
                                ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
                                ;*      CALL    $DB2O           ;;CALL THE ROUTINE
                                ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK

061136  104415          $DB2O:  SAVREG                  ;;SAVE ALL REGISTERS
061140  016601  000002          MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
061144  012705  061255          MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
061150  012704  000014          MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
061154  012703  177770          MOV     #^C7,R3         ;;MASK
061160  012100                  MOV     (R1)+,R0        ;;LOWER WORD
061162  012101                  MOV     (R1)+,R1        ;;HIGH WORD
061164  005002                  CLR     R2              ;;TERMINATOR
061166  110245          1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
061170  010002                  MOV     R0,R2           ;;GET THIS DIGIT
061172  005304                  DEC     R4              ;;COUNT THIS CHARACTER
061174  003007                  BGT     3$              ;;BR IF NOT THE LAST DIGIT
061176  001405                  BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
061200  005205                  INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
061202  010566  000002          MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
061206  104416                  RESREG                  ;;RESTORE ALL REGISTERS
061210  000207                  RETURN                  ;;RETURN TO USER
061212  006203          2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
061214  006001          3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
061216  006000                  ROR     R0              ;;      THE NEXT OCTAL DIGIT
061220  006001                  ROR     R1
061222  006000                  ROR     R0
061224  006001                  ROR     R1
061226  006000                  ROR     R0
061230  040302                  BIC     R3,R2           ;;MASK OUT ALL JUNK
061232  062702  000060          ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
061236  000753                  BR      1$              ;;GO PUT IT IN THE DATA TABLE
061240  000016          $OCTVL: .REPT   14.             ;;RESERVE DATA TABLE
061244          $OCT8=$OCTVL+4                          ;POINTER TO 11 DIGIT NUMBER
```

```
                                        .SBTTL  TABLES
                                        .SBTTL  APT MAILBOX-ETABLE
061256                          $MAIL:
061256  000000                  $MSGTY: .WORD   0       ;;MESSAGE TYPE CODE
061260  000000                  $FATAL: .WORD   0       ;;FATAL ERROR NUMBER (ERROR PC)
061262  000000                  $TESTN: .WORD   0       ;;TEST PATTERN NUMBER
061264  000000                  $PASS:  .WORD   0       ;;PASS COUNT
061266  000000                  $DEVCT: .WORD   0       ;;DEVICE COUNT
061270  000000                  $UNIT:  .WORD   0       ;;I/O UNIT NUMBER
061272  000000                  $MSGAD: .WORD   0       ;;MESSAGE ADDRESS
061274  000000                  $MSGLG: .WORD   0       ;;MESSAGE LENGTH
061276                          $ETABLE:                ;;APT ENVIRONMENT TABLE
061276      000                 $ENV:   .BYTE   0       ;;ENVIRONMENT BYTE      ;SET TO A 1 FOR APT AUTO MODE
                                ;NOTE:  IF BIT #7 IS SET IN $ENVM THE TABLE BELOW (BEGINNING AT $MAMS1 AND
                                ;       ENDING AT $MADR4) MUST BE FILLED IN TO INDICATE THE PROPER AMOUNT OF
                                ;       EACH TYPE OF MEMORY.
061277      000                 $ENVM:  .BYTE   0       ;ENVIRONMENT MODE
                                                        ;BIT7(200)=USE APT SIZE INFO ;BIT5(40)=NO CONSOLE
061300  000101                  $SWREG: .WORD   101     ;;APT SWITCH REGISTER
061302  000000                  $USWR:  .WORD   0       ;USED TO LIMIT THE NUMBER OF PASSES
061304  000000                  $CPUOP: .WORD   0       ;;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
061306      001                 $MAMS1: .BYTE   1       ;;HIGH ADDRESS,M.S. BYTE ;DEFAULT = 64K
061307      004                 $MTYP1: .BYTE   4       ;;MEM. TYPE,BLK#1
                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
                                ;*                              900 NSEC CORE=001
                                ;*                              300 NSEC BIPOLAR=002
                                ;*                              PARITY MOS=003
                                ;*                              ERROR CORRECTING MOS=004
061310  177776                  $MADR1: .WORD   177776  ;;HIGH ADDRESS,BLK#1
                                ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE'' ABOVE
061312      000                 $MAMS2: .BYTE   0       ;;HIGH ADDRESS,M.S. BYTE
061313      000                 $MTYP2: .BYTE   0       ;;MEM.TYPE,BLK#2
061314  000000                  $MADR2: .WORD   0       ;;MEM.LAST ADDRESS,BLK#2
061316      000                 $MAMS3: .BYTE   0       ;;HIGH ADDRESS,M.S.BYTE
061317      000                 $MTYP3: .BYTE   0       ;;MEM.TYPE,BLK#3
061320  000000                  $MADR3: .WORD   0       ;;MEM.LAST ADDRESS,BLK#3
061322      000                 $MAMS4: .BYTE   0       ;;HIGH ADDRESS,M.S.BYTE
061323      000                 $MTYP4: .BYTE   0       ;;MEM.TYPE,BLK#4
061324  000000                  $MADR4: .WORD   0       ;;MEM.LAST ADDRESS,BLK#4
061326  000000                  $VECT1: .WORD   0       ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
061330  000000                  $VECT2: .WORD   0       ;;INTERRUPT VECTOR#2BUS PRIORITY#2
061332  000000                  $BASE:  .WORD   0       ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
061334  000000                  $DEVM:  .WORD   0       ;;DEVICE MAP

061336  000000                  $CDW1:  .WORD   0
061340  000000                  $CDW2:  .WORD   0
```

```
11265                                   ;THE FOLLOWING LOCATIONS SPECIFY WHICH PATTERNS
11266                                   ;ARE TO BE RUN FOR PARTICULAR MEMORIES
11267                                   ;
11268                                   ;REFERENCE THE TABLE LISTED BELOW TO RELATE BITS TO PATTERNS.
11269                                   ;BIT0 SET WILL RUN THE FIRST ENTRY IN THE TABLE, BIT0 SET
11270                                   ;IN THE SECOND WORD WILL RUN THE 17TH ENTRY IN THE TABLE ...
11271                                   ;
11272                                   ;NOTE** NULL TESTS DO NOT TAKE ANY TIME
11273                                   ;                                   ;FIELD SERVICE VALUE
11274 061342  177777                   $DDW0:  .WORD   177777  ;ECC CSR TESTS          177777          TABLE = MKCSRT:
11275 061344  177777                   $DDW1:  .WORD   177777  ;ECC CSR TESTS          177777          TABLE = MKCSRT:
11276 061346  177777                   $DDW2:  .WORD   177777  ;ECC PATTERNS           103777          TABLE = MKPAT:
11277 061350  177777                   $DDW3:  .WORD   177777  ;ECC PATTERNS           177777          TABLE = MKPAT:
11278 061352  177777                   $DDW4:  .WORD   177777  ;PARITY PATTERNS        003777          TABLE = MJPAT:
11279 061354  177777                   $DDW5:  .WORD   177777  ;PARITY PATTERNS        177774          TABLE = MJPAT:
11283 061356                           $ETEND:
11284                                   ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
11285                                   ;INTERFACE SPEC.
11286
11287 061356                           $APTHD:
11288 061356  000000                   $HIBTS: .WORD   0               ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
11289 061360  061256                   $MBADR: .WORD   $MAIL           ;;ADDRESS OF APT MAILBOX (BITS 0-15)
11290 061362  000043                   $TSTM:  .WORD   35.             ;;RUN TIM OF LONGEST TEST
11291 061364  001274                   $PASTM: .WORD   700.            ;;RUN TIME IN SECS. OF 1ST PASS ON 128K (QUICK VERIFY)
11292 061366  000000                   $UNITM: .WORD   0.              ;;EXTRA RUN TIME OF A PASS FOR EACH ADDITIONAL 128K (QV)
11293 061370  000040                           .WORD   $ETEND-$MAIL/2  ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
11295                                   .SBTTL  ROUTINE TRAP DECODER
11296
11297                           ;*******************************************************************************
11298                           ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE ''TRAP'' INSTRUCTION
11299                           ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
11300                           ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
11301                           ;*GO TO THAT ROUTINE.
11302
11303 061372  010046            $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
11304 061374  016600  000002            MOV     2(SP),R0        ;;GET TRAP ADDRESS
11305 061400  005740                    TST     -(R0)           ;;BACKUP BY 2
11306 061402  111000                    MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
11307 061404  006300                    ASL     R0              ;;POSITION FOR INDEXING
11308 061406  016000  061434            MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
11309 061412  000200                    RTS     R0              ;;GO TO ROUTINE
11310
11311
11312                           ;;THIS IS USE TO HANDLE THE ''GETPRI'' MACRO
11313
11314 061414  011646            $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
11315 061416  016666  000004  000002    MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
11316 061424  000002                    RTI                     ;;RESTORE THE PSW
11317
11318 061426                    $NOTRAP:TYPE    MSG006          ;UNDEFINED TRAP INSTRUCTION
11319 061432  000000            $HALT2: HALT
```

```
11322                                           .SBTTL  TRAP TABLE
11323
11324                                   ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
11325                                   ;*BY THE 'TRAP'' INSTRUCTION.
11326
11327                                   ;       ROUTINE
11328                                   ;       -------
11329 061434  061414                   $TRPAD: .WORD   $TRAP2
11330 061436  052240                           $TYPE   ;CALL=TYPEIT    TRAP+1(104401)  TTY TYPEOUT ROUTINE
11331 061440  056546                           $TYPOC  ;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
11332 061442  056522                           $TYPOS  ;CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
11333 061444  061426                           $NOTRAP;$TYPON  ;CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
11334 061446  056750                           $TYPDS  ;CALL=TYPDS     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
11335 061450  061426                           $NOTRAP;$TYPBN  ;CALL=TYPBN     TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER
11336
11337 061452  057340                           $GTSWR  ;CALL=GTSWR     TRAP+7(104407)  GET SOFT-SWR SETTING
11338 061454  057174                           $CKSWR  ;CALL=CKSWR     TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
11339
11340 061456  057704                           $RDCHR  ;CALL=RDCHR     TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
11341 061460  060024                           $RDLIN  ;CALL=RDLIN     TRAP+12(104412) TTY TYPEIN STRING ROUTINE
11342 061462  060410                           $RDOCT  ;CALL=RDOCT     TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
11343 061464  060560                           $RDDEC  ;CALL=RDDEC     TRAP+14(104414) READ A DECIMAL NUMBER FROM TTY
11344
11345 061466  060746                           $SAVREG ;CALL=SAVREG    TRAP+15(104415) SAVE R0-R5 ROUTINE
11346 061470  061004                           $RESREG ;CALL=RESREG    TRAP+16(104406) RESTORE R0-R5 ROUTINE
11347
11348 061472  036656                           $KERNEL  ;CALL=KERNEL   TRAP+17(104417) ENTER KERNEL MODE
11349 061474  036666                           $ENERGIZE;CALL=ENERGIZETRAP+20(104420) TURN ON MEMORY MANAGEMENT & TRAPS
11350 061476  036676                           $DEENERGI;CALL=DEENERGITRAP+21(104421) TURN OFF MEMORY MENAGEMENT & TRAPS
11351
11352 061500  041062                           $KMAP    ;CALL=KMAP     TRAP+22(104422) MAP KERNEL 1 TO 1
11353
11354 061502  036706                           $CACHN  ;CALL=CACHON    TRAP+23(104423) TURN CACHE ON
11355 061504  036732                           $CACHF  ;CALL=CACHOFF   TRAP+24(104424) TURN CACHE OFF
11356
11357 061506  036750                           $LOADC  ;CALL=LOADCSR   TRAP+25(104425) LOAD CORRECT CSR
11358 061510  037044                           $READC  ;CALL=READCSR   TRAP+26(104426) READ CORRECT CSR
11359
11360 061512  052474                           $PER01  ;CALL=PERR01    TRAP+27(104427) PROGRAM DETECTED ERROR
11361 061514  052522                           $PER02  ;CALL=PERR02    TRAP+30(104430) PROGRAM DETECTED ERROR
11362 061516  052550                           $PER03  ;CALL=PERR03    TRAP+31(104431) PROGRAM DETECTED ERROR
11363 061520  052600                           $PER04  ;CALL=PERR04    TRAP+32(104432) PROGRAM DETECTED ERROR
11364 061522  052662                           $PER07  ;CALL=PERR07    TRAP+33(104433) PROGRAM DETECTED ERROR
11365 061524  052704                           $PER10  ;CALL=PERR10    TRAP+34(104434) PROGRAM DETECTED ERROR
11366 061526  052734                           $PER11  ;CALL=PERR11    TRAP+35(104435) PROGRAM DETECTED ERROR
11367 061530  052754                           $PER12  ;CALL=PERR12    TRAP+36(104436) PROGRAM DETECTED ERROR
11368 061532  052776                           $PER13  ;CALL=PERR13    TRAP+37(104437) PROGRAM DETECTED ERROR
11369 061534  053016                           $PER14  ;CALL=PERR14    TRAP+40(104440) PROGRAM DETECTED ERROR
11370 061536  053040                           $PER15  ;CALL=PERR15    TRAP+41(104441) PROGRAM DETECTED ERROR
11371 061540  053062                           $PER16  ;CALL=PERR16    TRAP+42(104442) PROGRAM DETECTED ERROR
11372 061542  053102                           $PER17  ;CALL=PERR17    TRAP+43(104443) PROGRAM DETECTED ERROR
11373 061544  053120                           $PER20  ;CALL=PERR20    TRAP+44(104444) PROGRAM DETECTED ERROR
11374 061546  053136                           $PER21  ;CALL=PERR21    TRAP+45(104445) PROGRAM DETECTED ERROR
11375 061550  053156                           $PER22  ;CALL=PERR22    TRAP+46(104446) PROGRAM DETECTED ERROR
11376 061552  053174                           $PER23  ;CALL=PERR23    TRAP+47(104447) PROGRAM DETECTED ERROR
11377 061554  053212                           $PER24  ;CALL=PERR24    TRAP+50(104450) PROGRAM DETECTED ERROR
11378 061556  050054                           $PER25  ;CALL=PERR25    TRAP+51(104451) PROGRAM DETECTED ERROR
```

```
 11379 061560  053402                $PER26  ;CALL=PERR26    TRAP+52(104452) PROGRAM DETECTED ERROR
 11380 061562  053422                $PER27  ;CALL=PERR27    TRAP+53(104453) PROGRAM DETECTED ERROR
 11381 061564  050302                $PER30  ;CALL=PERR30    TRAP+54(104454) PROGRAM DETECTED ERROR
 11382 061566  053612                $PER31  ;CALL=PERR31    TRAP+55(104455) PROGRAM DETECTED ERROR
 11383 061570  053710                $PER32  ;CALL=PERR32    TRAP+56(104456) PROGRAM DETECTED ERROR
 11384 061572  053756                $PER33  ;CALL=PERR33    TRAP+57(104457) PROGRAM DETECTED ERROR
 11385 061574  054036                $PER34  ;CALL=PERR34    TRAP+60(104460) PROGRAM DETECTED ERROR
 11386 061576  054070                $PER35  ;CALL=PERR35    TRAP+61(104461) PROGRAM DETECTED ERROR
 11387 061600  054124                $PER36  ;CALL=PERR36    TRAP+62(104462) PROGRAM DETECTED ERROR
 11388 061602  061426                $NOTRAP ;CALL=PERR37    TRAP+63(104463) PROGRAM DETECTED ERROR
 11389 061604  061426                $NOTRAP ;CALL=PERR40    TRAP+64(104464) PROGRAM DETECTED ERROR
 11390 061606  061426                $NOTRAP ;CALL=PERR41    TRAP+65(104465) PROGRAM DETECTED ERROR
 11391 061610  061426                $NOTRAP ;CALL=PERR42    TRAP+66(104466) PROGRAM DETECTED ERROR
 11392 061612  061426                $NOTRAP ;CALL=PERR43    TRAP+67(104467) PROGRAM DETECTED ERROR
 11393
 11394 061614  037266                $ECCDIS ;CALL=ECCDIS    TRAP+70(104470) DISABLE ECC ON ALL CSR'S
 11395 061616  037302                $ECC1DIS;CALL=ECC1DIS   TRAP+71(104471) DISABLE ECC ON 1 SELECTED CSR
 11396 061620  037314                $ECCINIT;CALL=ECCINIT   TRAP+72(104472) INITIALIZE ALL MK11 CSR'S
 11397 061622  037330                $ECC1INIT;CALL=ECC1INIT TRAP+73(104473) INITIALIZE 1 SELECTED MK11 CSR
 11398 061624  037370                $CBCSR  ;CALL=CBCSR     TRAP+74(104474) WRITE GENERATED CHECKBITS IN ALL CSR'S
 11399 061626  037412                $CB1CSR ;CALL=CB1CSR    TRAP+75(104475) WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
 11400 061630  037432                $WASSBE ;CALL=WASSBE    TRAP+76(104476) WAS THERE A SBE ON ANY CSR?
 11401 061632  037546                $WAS1SBE;CALL=WAS1SBE   TRAP+77(104477) WAS THERE A SBE ON 1 SELECTED CSR?
 11402 061634  037576                $WASDBE ;CALL=WASDBE    TRAP+100(104500)WAS THERE A DBE ON ANY CSR?
 11403 061636  037712                $WAS1DBE;CALL=WAS1DBE   TRAP+101(104501)WAS THERE A DBE ON 1 SELECTED CSR?
 11404 061640  037742                $CLRCSR ;CALL=CLRCSR    TRAP+102(104502)CLEAR ALL CSR'S
 11405 061642  037754                $CLR1CSR;CALL=CLR1CSR   TRAP+103(104503)CLEAR 1 SELECTED CSR
 11406 061644  037764                $CHKDIS ;CALL=CHKDIS    TRAP+104(104504)DISABLE ECC & WRITE CKBITS FROM ALL CSR'S
 11407 061646  040000                $CHK1DIS;CALL=CHK1DIS   TRAP+105(104505)DISABLE ECC & WRITE CKBITS FROM 1 CSR
 11408 061650  037342                $ENASBE ;CALL=ENASBE    TRAP+106(104506)ENABLE TRAPS ON SBE'S FROM ALL CSR'S
 11409 061652  037356                $ENA1SBE;CALL=ENA1SBE   TRAP+107(104507)ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
 11410 061654  037064                $TSTRD  ;CALL=TSTREAD   TRAP+110(104510)TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES
 11411 061656  040060                $INVALID;CALL=INVALID   TRAP+111(104511)INVALIDATE BACKGROUND PATTERN ON BANK
 11412 061660  040110                $ERRGEN ;CALL=ERRGEN    TRAP+114(104512)TEST ERROR ADDRESS
 11413 061662  061426                $NOTRAP
 11414 061664  061426                $NOTRAP
 11415 061666  061426                $NOTRAP
 11416 061670  061426                $NOTRAP
 11417 061672  061426                $NOTRAP
 11418 061674  061426                $NOTRAP
 11419 061676  061426                $NOTRAP
```

```
11422                                   .SBTTL ODT (CZMSD) DATA AREA
11423   177776          ST      =       177776          ;STATUS REGISTER
11424   000016          0.BKP   =       16              ;NUMBER OF BREAKPOINTS-1 MULT. BY 2
11425   000014          0.TVEC  =       14              ;TRT VECTOR LOCATION
11426   000340          0.STM   =       340             ;PRIORITY MASK - STATUS REGISTER
11427   000020          0.TBT   =       20              ;T-BIT MASK - STATUS REGISTER
11428   000003          TRT     =       000003          ;TRT INSTRUCTION
11429   177562          0.RDB   =       177562          ;R DATA BUFFER
11430   177560          0.RCSR  =       177560          ;R C/SR
11431   177566          0.TDB   =       177566          ;T DATA BUFFER
11432   177564          0.TCSR  =       177564          ;T C/SR
```

```
11435 061700  000000              O.CAD:  0                ; CURRENT ADDRESS
11436 061702  000000              O.DOT:  0                ; ORIGIN ADDRESS
11437 061704  000000              O.XXX:  .WORD   0        ;TEMPORARY STORAGE
11438 061706  000000              O.BW:   .WORD   0        ; =0 - ALL CLOSED
11439                                                      ; =1 - BYTE OPEN,
11440                                                      ; =2 - WORD OPEN
11441 061710      000             O.SEQ:  .BYTE   0        ;CHANGE SEQUENCE INDICATOR
11442 061711      000             O.S:    .BYTE   0        ;SINGLE INSTRUCTION FLAG
11443                                                      ;0 IF NOT ACTIVE
11444                                                      ;-1 IF ACTIVE
11445                                                      ;NO BREAK BOINTS MAY BE SET WHILE IN
11446                                                      ;SINGLE INSTRUCTION MODE
11447 061712      000             O.T:    .BYTE   0        ;  T-BIT FLAG
11448 061713      000             O.P:    .BYTE   0        ;PROCEED FLAG = -2 IF MANUAL ENTRY
11449                                                      ;                -1 IF NO PROCEED ALLOWED
11450                                                      ;                0-7 IF PCEED ALLOWED
11451 061714      000             O.CSR1: .BYTE   0        ;SAVE CELL - R C/SR
11452 061715      000             O.CSR2: .BYTE   0        ;SAVE CELL - T C/SR
11453 061716  000000              O.FIL:  0                ;ILL COUNTER
11454 061720      000             O.CMFD: .BYTE   0        ;COMMA FOUND SWITCH, =0 NO COMMA FOUND
11455                                                      ;                    =1 COMMA FOUND
11456 061721      000             O.SMFD: .BYTE   0        ;SEMICOLON FOUND SWITCH
11457                                                      ;=0 NO SEMICOLON FOUND
11458                                                      ;=1 SEMICOLON FOUND
11459 061722      000             O.SCRN: .BYTE   0        ;FLAG; 1=PASS SPACES ON FROM TTY
11460                                                      ;ALSO, IF =1, <LF> IS ECHOED
11461 061723      000             O.MINS: .BYTE   0        ;MINUS SIGN TYPED  (SWITCH)
11462                                                      ;0=NO MINUS TYPED;  1=MINUS SIGN TYPED
11463 061724  000000              O.BIAS: .WORD   0        ;CURRENT RELOCATION BIAS
```

```
                                        ;THE ORDER OF THE FOLLOWING ENTRIES IS CRITICAL
                        000026                  .REPT   26
000000                                  O.UR0:  0       ;USER R0
000000                                          0       ;      R1
000000                                          0       ;      R2
000000                                          0       ;      R3
000000                                          0       ;      R4
000000                                          0       ;      R5
000000                                  O.USP:  0       ;USER SP
000000                                  O.UPC:  0       ;USER PC
000000                                  O.UST:  0       ;USER ST
000007                                  O.PRI:  7       ;ODT PRIORITY
                                        ; BREAK POINT LISTS, ADR1 = ADDRESS OF BREAKPOINT,CT = COUNT,
                                        ;   UIN = CONTENTS
                                        
000000                                  O.ADR1: .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                  O.CT:   .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                  O.UIN:  .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
000000                                          .WORD 0
```

```
11515                                   ; INITIALIZE ODT
11516                                   ;  USE 0.ODT FOR A NORMAL ENTRY
11517                                   ;  USE 0.ODT+2 TO RESTART ODT - WIPING OUT ALL BREAKPOINTS
11518                                   ;  USE 0.ODT+4 TO RE-ENTER (I.E. - FAKE A BREAKPOINT)
11519
11520 062114  000421                  0.ODT:  BR      2$              ;NORMAL ENTRY
11521 062116  000440                          BR      3$              ;RESTART
11522 062120  004737  065312          1$:     JSR     PC,0.RRST       ;RE-ENTER  --  SAVE STATUS
11523 062124  013746  000016                  MOV     0.TVEC+2,-(SP)  ;SET UP LOCAL STATUS
11524 062130  004737  065322                  JSR     PC,0.WST
11525 062134  010746                          MOV     PC,-(SP)        ;FAKE THE PC
11526 062136  012637  062020                  MOV     (SP)+,0.UPC
11527 062142  112737  177777  061713          MOVB    #-1,0.P         ;DISALLOW PROCEED
11528 062150  105037  061711                  CLRB    0.S
11529 062154  000137  064200                  JMP     0.BK1
11530
11531                                         ;RUN AT CURRENT STATUS
11532 062160  004537  064620          2$:     JSR     5,0.SVTT        ;SAVE TTY STATUS
11533 062164  012737  062120  057254          MOV     #1$,@#CONTP+2
11534 062172  004037  064500                  JSR     0,0.SVR         ;SAVE REGISTERS (MAINLY SP)
11535 062176  012706  062002                  MOV     #0.UR0,SP       ;SET UP STACK
11536 062202  012704  065420                  MOV     #0.ID,R4        ;TYPE ID
11537 062206  012703  065433                  MOV     #0.IDND,R3
11538 062212  004537  065246                  JSR     5,0.TYPE
11539 062216  000414                          BR      4$
11540 062220  004037  064500          3$:     JSR     0,0.SVR         ;SAVE REGISTERS
11541 062224  004537  064722                  JSR     5,0.REM         ;REMOVE ALL BREAKPOINTS
11542 062230  113704  062024                  MOVB    0.PRI,R4        ;GET ODT PRIORITY
11543 062234  106004                          RORB    R4              ;SHIFT
11544 062236  106004                          RORB    R4              ; INTO
11545 062240  106004                          RORB    R4              ;  POSITION
11546 062242  010446                          MOV     R4,-(SP)        ;STORE IN STATUS
11547 062244  004737  065322                  JSR     PC,0.WST
11548 062250  105037  061711          4$:     CLRB    0.S             ;DISABLE SINGLE INSTRUCTION FOR NOW
11549 062254  112737  177777  061713          MOVB    #-1,0.P         ;DISALLOW PROCEED
11550 062262  012737  000340  000016          MOV     #0.STM,0.TVEC+2 ;STATUS WORD TO TRT VECTOR+2
11551 062270  012737  064162  000014          MOV     #0.BRK,0.TVEC   ;PC TO TRT VECTOR
11552 062276  000137  063550                  JMP     0.RALL          ;CLEAR BRK PT TABLES
11553                                   ;
11554                                   ;0.CTLC
11555                                   ;       ^C PROCESSING. RETURN TO KEYBOARD MONITOR
11556                                   ;
11557                                   ;
11558 062302  013706  062016          0.CTLC: MOV     0.USP,SP        ;RESTORE USER STACK
11559 062306  012704  062326                  MOV     #1$,R4
11560 062312  012703  062327                  MOV     #2$,R3
11561 062316  004537  065246                  JSR     R5,0.TYPE       ;ECHO ''^C''
11562 062322  000137  043426                  JMP     BOOT
11563 062326     136                  1$:     .BYTE   '^
11564 062327     103                  2$:     .BYTE   'C
```

```
11567                                   ;CONTROL F PROCESSING
11568 062330  013706  062016            0.CTLF: MOV     0.USP,SP        ;RESTORE USER STACK
11569 062334  012704  062360                    MOV     #1$,R4
11570 062340  012703  062361                    MOV     #2$,R3
11571 062344  004537  065246                    JSR     R5,0.TYPE       ;ECHO ^F
11572 062350  004737  044004                    CALL    FIELDSERVICE
11573 062354  000137  062114                    JMP     0.ODT
11574 062360     136                    1$:     .BYTE   '^
11575 062361     106                    2$:     .BYTE   'F
11576
11577                                   ;CONTROL E PROCESSING
11578 062362  012704  062406            0.CTLE: MOV     #1$,R4
11579 062366  012703  062407                    MOV     #2$,R3
11580 062372  004537  065246                    JSR     R5,0.TYPE       ;ECHO ^E
11581 062376  012705  057256                    MOV     #CONTP+4,R5
11582 062402  000137  063744                    JMP     0.GOGO
11583 062406     136                    1$:     .BYTE   '^
11584 062407     105                    2$:     .BYTE   'E
11585
11586                                   ; SPECIAL NAME HANDLER
11587                                   ;  DEPENDS UPON THE EXPLICIT ORDER OF THE TWO TABLES 0.TL AND 0.URO
11588
11589 062410  004537  065126            0.REGT: JSR     5,0.GET         ;SPECIAL NAME, GET ONE MORE CHARACTER
11590 062414  012704  065466                    MOV     #0.TL,R4        ;TABLE START ADDRESS
11591 062420  120024                    1$:     CMPB    R0,(R4)+        ;IS THIS THE CORRECT CHARACTER?
11592 062422  001414                            BEQ     3$              ;JUMP IF YES
11593 062424  022704  065506                    CMP     #0.TL+0.LG,R4   ;IS THE SEARCH DONE?
11594 062430  101373                            BHI     1$              ;BRANCH IF NOT
11595 062432  042700  177770                    BIC     #177770,R0      ;MASK OFF OCTAL
11596 062436  010004                            MOV     R0,R4
11597 062440  006304                    2$:     ASL     R4
11598 062442  062704  062002                    ADD     #0.URO,R4       ;GENERATE ADDRESS
11599 062446  005202                            INC     R2              ;SET FOUND FLAG
11600 062450  000137  062674                    JMP     0.SCAN          ;GO FIND NEXT CHARACTER
11601 062454  162704  065457            3$:     SUB     #0.TL-7,R4      ;GO FIND NEXT CHARACTER
11602 062460  000767                            BR      2$
```

```
11605                                   ; 'BACKARROW'  HANDLER - OPEN INDEXED ON THE PC (BACK ARROW)
11606                                           .ENABL  LSB
11607 062462  004537  062536          O.ORPC:   JSR     5,2$              ;TEST WORD MODE AND CLOSE
11608 062466  061202                            ADD     @R2,R2            ;COMPUTE
11609 062470  005202                            INC     R2
11610 062472  005202                            INC     R2                ;   NEW ADDRESS
11611 062474  010237  061700          1$:       MOV     R2,O.CAD          ;UPDATE CAD
11612 062500  000137  063346                    JMP     O.OP2A            ;GO FINISH UP
11613 062504  004537  062536          O.ORAB:   JSR     5,2$              ;TEST WORD MODE AND CLOSE
11614 062510  011202                            MOV     @R2,R2            ;GET ABSOLUTE ADDRESS
11615 062512  000770                            BR      1$
11616 062514  004537  062536          O.ORRB:   JSR     5,2$              ;TEST AND CLOSE
11617 062520  011201                            MOV     @R2,R1            ;COMPUTE NEW ADDRESS
11618 062522  110101                            MOVB    R1,R1             ;EXTEND THE SIGN
11619 062524  006301                            ASL     R1                ;R2=2(@R2)
11620 062526  005201                            INC     R1                ;   +2
11621 062530  005201                            INC     R1
11622 062532  060102                            ADD     R1,R2             ;   +PC
11623 062534  000757                            BR      1$
11624 062536  004737  065334          2$:       JSR     PC,O.CLSE         ;CLOSE CURRENT CELL
11625 062542  022737  000002  061706            CMP     #2,O.BW           ;ONLY WORD MODE ALLOWED
11626 062550  001003                            BNE     3$                ;BRANCH IF ERROR
11627 062552  013702  061700                    MOV     O.CAD,R2          ;CURRENT ADDRESS IN R2
11628 062556  000205                            RTS     R5
11629 062560  005726                  3$:       TST     (SP)+
11630 062562  000137  062630          O.TCL2:   JMP     O.ERR             ;POP A WORD AND SHOW THE ERROR
11631                                           .DSABL  LSB
11632
11633                                   ;       PROCESS S - SINGLE INSTRUCTION MODE
11634 062566  105737  061721          O.SNGL:   TSTB    O.SMFD            ;DONT REACT IF ; NOT TYPED
11635 062572  001773                            BEQ     O.TCL2
11636 062574  005702                            TST     R2                ;SEE IF TURN ON OR TURN OFF
11637 062576  001004                            BNE     1$                ;BRANCH IF TURNING IT ON
11638 062600  105037  061711                    CLRB    O.S               ;CLEAR THE FLAG
11639 062604  000137  062640                    JMP     O.DCD             ;CONTINUE THE SCAN
11640 062610  112737  177777  061711  1$:       MOVB    #-1,O.S           ;SET THE FLAG
11641 062616  000137  062640                    JMP     O.DCD
```

```
11644                                   ;
11645                                   ;MINUS PROCESSING
11646                                   ;
11647 062622  105237  061723          O.MIN:  INCB    O.MINS          ;SET MINUS FOUND SWITCH ON
11648 062626  000420                          BR      O.DCD1
11649
11650                                   ; COMMAND DECODER - ODT11X
11651
11652                                   ;  ALL REGISTERS MAY BE USED (R0-R5),
11653
11654 062630  012700  000077          O.ERR:  MOV     #'?,R0          ;  ? TO BE TYPED
11655 062634  004537  065050                  JSR     5,O.FTYP        ; OUTPUT ?
11656 062640  005037  061706          O.DCD:  CLR     O.BW            ;CLOSE ALL
11657 062644  004537  065374                  JSR     5,O.CRLS        ;TYPE  <CR><LF>*
11658 062650  105037  061721          O.DCD3: CLRB    O.SMFD          ;SEMICOLON FOUND FLAG
11659 062654  105037  061720                  CLRB    O.CMFD          ;COMMA FOUND FLAG
11660 062660  105037  061723                  CLRB    O.MINS          ;MINUS SIGN FOUND FLAG
11661 062664  005003                          CLR     R3              ;R3 IS A SAVE REGISTER FOR R2
11662 062666  005005                          CLR     R5              ;R5 IS A SAVE REGISTER FOR R4
11663 062670  005004          O.DCD1: CLR     R4              ; R4 CONTAINS THE CONVERTED OCTAL
11664 062672  005002                          CLR     R2              ; R2 IS THE NUMBER FOUND FLAG
11665 062674  004537  065126          O.SCAN: JSR     5,O.GET         ;GET A CHAR, RETURN IN R0
11666 062700  022700  000060                  CMP     #'0,R0          ;COMPARE WITH ASCII 0
11667 062704  101013                          BHI     5$              ;CHECK LEGALITY IF NON-NUMERIC
11668 062706  022700  000067                  CMP     #'7,R0          ;COMPARE WITH ASCII 7
11669 062712  103410                          BLO     5$              ;CHECK LEGALITY IF NOT OCTAL
11670 062714  042700  177770                  BIC     #177770,R0      ;CONVERT TO BCD
11671 062720  006304                          ASL     R4              ; MAKE ROOM
11672 062722  006304                          ASL     R4              ;  IN
11673 062724  006304                          ASL     R4              ;    R4
11674 062726  060004                          ADD     R0,R4           ;PACK THREE BITS IN R4
11675 062730  005202                          INC     R2              ;R2 HAS NUMERIC FLAG
11676 062732  000760                          BR      O.SCAN          ;   AND TRY AGAIN
11677 062734  005001          5$:     CLR     R1              ;CLEAR INDEX
11678 062736  120061  065442          1$:     CMPB    R0,O.LGCH(R1)   ;DO THE CODES MATCH?
11679 062742  001405                          BEQ     2$              ;JUMP IF YES
11680 062744  005201                          INC     R1              ; SET INDEX FOR NEXT SEARCH
11681 062746  020127  000024                  CMP     R1,#O.CLGT      ;IS THE SEARCH DONE?
11682 062752  103326                          BHIS    O.ERR           ;    OOPS!
11683 062754  000770                          BR      1$              ;RE-LOOP
11684 062756  105737  061723          2$:     TSTB    O.MINS          ;IF MINUS WAS NOT TYPED
11685 062762  001401                          BEQ     4$              ;DO NOT NEGATE K
11686 062764  005404                          NEG     R4              ;OTHERWISE, TAKE 2'S COMPLEMENT.
11687 062766  105737  061720          4$:     TSTB    O.CMFD          ;IF A COMMA NOT TYPED,
11688 062772  001402                          BEQ     3$              ;SKIP NEXT INSTRUCTION.
11689 062774  063704  061724                  ADD     O.BIAS,R4       ;OTHERWISE, ADD RELOC. BIAS TO (R4)
11690 063000  105037  061723          3$:     CLRB    O.MINS          ;REINITIALIZE MINUS-TYPED SWITCH FOR NXT SCAN
11691 063004  006301                          ASL     R1              ;MULTIPLY BY TWO
11692 063006  000171  063012                  JMP     @6$(R1)         ;GO TO PROPER ROUTINE
```

```
 11695 063012  063062                6$:     O.SEMI  ;  ; SEMICOLON
 11696 063014  063100                        O.WRD   ;  /    OPEN WORD
 11697 063016  063112                        O.BYT   ;  \    OPEN BYTE -BACK SLASH
 11698 063020  063260                        O.CRET  ;  CARRIAGE RETURN    CLOSE
 11699 063022  062410                        O.REGT  ;  $  REGISTER OPS
 11700 063024  063716                        O.GO    ;  G  GO TO ADDRESS K
 11701 063026  063300                        O.OP1   ;  <LF>  MODIFY, CLOSE, OPEN NEXT
 11702 063030  062462                        O.ORPC  ;  'BACKARROW''  OPEN RELATED, INDEX - PC (BACK ARROW)
 11703 063032  063274                        O.OLD   ;  <  RETURN TO OLD SEQUENCE AND OPEN
 11704 063034  063422                        O.BACK  ;  ^  OPEN PREVIOUS (UP ARROW)
 11705 063036  063606                        O.OFST  ;  O  OFFSET
 11706 063040  063444                        O.BKPT  ;  B  BREAKPOINTS
 11707 063042  064052                        O.PROC  ;  P  PROCEED
 11708 063044  062504                        O.ORAB  ;  @  OPEN RELATED, ABSOLUTE
 11709 063046  062514                        O.ORRB  ;  >  OPEN RELATED, REL. BRANCH
 11710 063050  062566                        O.SNGL  ;  S  SINGLE INSTRUCTION MODE
 11711 063052  062622                        O.MIN   ;  - MINUS, NEGATES NUMBER TYPED IN
 11712 063054  062302                        O.CTLC  ;  ^C EXIT TO MONITOR
 11713 063056  062330                        O.CTLF  ;  ^F EXIT TO FIELD SERVICE MODE
 11714 063060  062362                        O.CTLE  ;  ^E PROCEED FROM ^D
```

```
11717
11718                                   ; SEMI-COLON PROCESSOR
11719
11720 063062  010203                    0.SEMI: MOV     R2,R3           ;A SEMI-COLON HAS BEEN RECEIVED
11721 063064  010405                            MOV     R4,R5           ;NUMERIC FLAG TO R3, CONTENTS TO R5
11722 063066  105237  061721                    INCB    0.SMFD          ;SET SEMICOLON FOUND FLAG
11723 063072  105037  061720                    CLRB    0.CMFD          ;RESET COMMA FOUND FLAG
11724 063076  000674                            BR      0.DCD1          ;GO BACK FOR MORE
11725
11726                                   ; PROCESS / AND \ - OPEN WORD OR BYTE
11727
11728                                   ;INPUT - IF R2 IS NON-ZERO A NEW REXP HAS BEEN TYPED IN
11729                                   ;INPUT - -ADDRESS OF WORD TO BE OPENED IS IN R4
11730                                           .ENABL  LSB
11731 063100  012737  000002  061706    0.WRD:  MOV     #2,0.BW         ;OPEN WORD
11732 063106  000404                            BR      11$
11733 063110  006104                    1$:     ROL     R4              ;GET THE ADDRESS BACK
11734 063112  012737  000001  061706    0.BYT:  MOV     #1,0.BW         ;OPEN BYTE
11735 063120  005702                    11$:    TST     R2              ;GET VALUE IF R2 IS NON-ZERO
11736 063122  001011                            BNE     14$             ;BRANCH IF NUMBER INPUT
11737 063124  105737  061720                    TSTB    0.CMFD          ;TEST FOR ","AND";"
11738 063130  001402                            BEQ     12$
11739 063132  000137  062630            13$:    JMP     0.ERR           ;ERROR IF PRESENT WITHOUT NUMBER.
11740 063136  105737  061721            12$:    TSTB    0.SMFD
11741 063142  001373                            BNE     13$
11742 063144  000404                            BR      0.WRD1          ;NO NUMBER - REOPEN PREVIOUS LOCATION
11743 063146  010437  061702            14$:    MOV     R4,0.DOT        ;PUT VALUE IN DOT
11744 063152  010437  061700                    MOV     R4,0.CAD        ;    ALSO IN CAD
11745 063156  022737  000001  061706    0.WRD1: CMP     #1,0.BW         ;CHECK BYTE MODE
11746 063164  001407                            BEQ     22$             ;JUMP IF BYTE
11747 063166  013704  061700                    MOV     0.CAD,R4
11748 063172  006204                            ASR     R4              ;MOVE ONE BIT TO CARRY
11749 063174  103745                            BCS     1$              ;JUMP IF ODD ADDRESS
11750 063176  017700  176476                    MOV     @0.CAD,R0       ;GET CONTENTS OF WORD
11751 063202  000402                            BR      23$
11752 063204  117700  176470            22$:    MOVB    @0.CAD,R0       ;GET CONTENTS OF BYTE
11753 063210  004537  064754            23$:    JSR     5,0.CADV        ;GO GET AND TYPE OUT @CAD
11754 063214  022737  000001  061706            CMP     #1,0.BW         ;CHECK IF BYTE MODE.
11755 063222  001212                            BNE     0.DCD3          ;IF NOT WE'RE DONE. ELSE:
11756 063224  112700  000075                    MOVB    #'=,R0          ;TYP "=" AND THEN THE ASCII BYTE
11757 063230  004537  065050                    JSR     5,0.FTYP
11758 063234  117700  176440                    MOVB    @0.CAD,R0
11759 063240  004537  065050                    JSR     5,0.FTYP
11760 063244  112700  000040                    MOVB    #' ,R0
11761 063250  004537  065050                    JSR     5,0.FTYP
11762 063254  000137  062650                    JMP     0.DCD3          ;GO BACK TO DECODER
11763                                           .DSABL  LSB
```

```
11766                                   ; PROCESS CARRIAGE RETURN
11767 063260  004737  065334            O.CRET: JSR     PC,O.CLSE       ;CLOSE LOCATION
11768 063264  000137  062640            O.DCDA: JMP     O.DCD           ;RETURN TO DECODER
11769
11770 063270  000137  062630            O.ERR3: JMP     O.ERR           ; INTERMEDIATE HELP
11771
11772                                   ; PROCESS <LF>, OPEN NEXT WORD
11773
11774 063274  105237  061710            O.OLD:  INCB    O.SEQ           ;SET FLAG TO LATER RESTORE CAD
11775 063300  005737  061706            O.OP1:  TST     O.BW            ;<LF> RECEIVED
11776 063304  001771                    O.ERR2: BEQ     O.ERR3          ;ERROR IF NOTHING IS OPEN
11777 063306  004737  065334                    JSR     PC,O.CLSE       ;CLOSE PRESENT CELL
11778 063312  105737  061710                    TSTB    O.SEQ           ;SHOULD CAD BE RESTORED?
11779 063316  001405                            BEQ     1$              ;BRANCH IF NOT
11780 063320  013737  061702  061700            MOV     O.DOT,O.CAD     ;RESTORE PREVIOUS SEQUENCE
11781 063326  105037  061710                    CLRB    O.SEQ           ;RESET FLAG; NO LONGER NEEDED
11782 063332  063737  061706  061700  1$:       ADD     O.BW,O.CAD      ;GENERATE NEW ADDRESS
11783 063340  013737  061700  061702  O.OP2:   MOV     O.CAD,O.DOT     ;INITIALIZE DOT
11784 063346  004537  065366            O.OP2A: JSR     5,O.CRLF        ;<CR><LF>
11785 063352  013746  061706                    MOV     O.BW,-(SP)      ;SAVE BW
11786 063356  012737  000002  061706            MOV     #2,O.BW         ;SET TO TYPE FULL WORD ADDRESS
11787 063364  013700  061700                    MOV     O.CAD,R0        ;NUMBER TO TYPE
11788 063370  004537  065412                    JSR     5,O.RORA        ; CHECK FORMAT
11789 063374  011637  061706                    MOV     @SP,O.BW        ;RESTORE BW
11790 063400  012700  027534                    MOV     #27534,R0       ;PUT "/\" IN R0
11791 063404  022726  000001                    CMP     #1,(SP)+        ;IS IT BYTE MODE?
11792 063410  001401                            BEQ     1$              ;JUMP IF YES
11793 063412  000300                            SWAB    R0              ;TYPE A /
11794 063414  004537  065050            1$:     JSR     5,O.FTYP        ;TYPE THE LOW BYTE OF R0
11795 063420  000656                            BR      O.WRD1          ;GO PROCESS IT
11796
```

```
11799
11800                                   ; PROCESS ^, OPEN PREVIOUS WORD
11801
11802 063422  005737  061706          O.BACK: TST     O.BW            ;   ^ RECEIVED
11803 063426  001726                          BEQ     O.ERR2          ;ERROR IF NOTHING OPEN
11804 063430  004737  065334                  JSR     PC,O.CLSE
11805 063434  163737  061706  061700        SUB     O.BW,O.CAD      ;GENERATE NEW ADDRESS
11806 063442  000736                          BR      O.OP2           ;GO DO THE REST
11807
11808                                   ; B HANDLER - SET AND REMOVE BREAKPOINTS
11809
11810 063444  012700  065510          O.BKPT: MOV     #O.TRTC,R0
11811 063450  006304                          ASL     R4              ;MULTIPLY NUMBER BY TWO
11812 063452  005703                          TST     R3
11813 063454  001423                          BEQ     3$              ;IF R3 IS ZERO GO REMOVE BREAKPOINT
11814 063456  006205                          ASR     R5              ;GET ONE BIT TO CARRY
11815 063460  103514                          BCS     O.ERR1          ;BADNESS IF ODD ADDRESS
11816 063462  006305                          ASL     R5              ;RESTORE ONE BIT
11817 063464  062704  062026                  ADD     #O.ADR1,R4
11818 063470  005702                          TST     R2
11819 063472  001007                          BNE     1$              ;JUMP IF SPECIFIC CELL
11820 063474  020014                  2$:     CMP     R0,@R4          ;IS THIS CELL FREE?
11821 063476  001405                          BEQ     1$              ;JUMP IF YES
11822 063500  020427  062044                  CMP     R4,#O.BKP+O.ADR1;ARE WE AT THE END OF OUR ROPE
11823 063504  103102                          BHIS    O.ERR1          ;YES, THERE IS NOTHING FREE
11824 063506  005724                          TST     (R4)+           ;INCREMENT BY TWO
11825 063510  000771                          BR      2$
11826 063512  020427  062044          1$:     CMP     R4,#O.BKP+O.ADR1
11827 063516  101075                          BHI     O.ERR1          ;ERROR IF TOO LARGE
11828 063520  010514                          MOV     R5,@R4          ;SET BREAKPOINT
11829 063522  000660                          BR      O.DCDA          ;RETURN
11830 063524  005702                  3$:     TST     R2
11831 063526  001410                          BEQ     O.RALL          ;GO REMOVE ALL
11832 063530  020427  000016                  CMP     R4,#O.BKP
11833 063534  101066                          BHI     O.ERR1          ;JUMP IF NUMBER TOO LARGE
11834 063536  010064  062026                  MOV     R0,O.ADR1(R4)   ;CLEAR BREAKPOINT
11835 063542  005064  062050                  CLR     O.CT(R4)        ;CLEAR COUNT ALSO
11836 063546  000646                          BR      O.DCDA
11837 063550  005004                  O.RALL: CLR     R4
11838 063552  012700  065510                  MOV     #O.TRTC,R0
11839 063556  020427  000020          1$:     CMP     R4,#O.BKP+2     ;ALL DONE?
11840 063562  101240                          BHI     O.DCDA          ;JUMP IF YES
11841 063564  010064  062026                  MOV     R0,O.ADR1(R4)   ;RESET BKPT
11842 063570  012764  000003  062072          MOV     #TRT,O.UIN(R4)  ;RESET CONTENTS OF TABLE
11843 063576  005064  062050                  CLR     O.CT(R4)        ;CLEAR COUNT
11844 063602  005724                          TST     (R4)+           ;INCREMENT BY TWO
11845 063604  000764                          BR      1$
11846
```

```
11849
11850                                   ; PROCESS O, COMPUTE OFFSET
11851
11852 063606  022737  000002  061706  O.OFST: CMP     #2,O.BW         ;CHECK WORD MODE
11853 063614  001036                          BNE     O.ERR1          ;ERROR IF NOT CORRECT MODE
11854 063616  012700  000040                  MOV     #' ,R0          ;TYPE ONE BLANK
11855 063622  004537  065050                  JSR     5,O.FTYP        ;  AS A SEPARATOR
11856 063626  005703                          TST     R3              ;WAS SEMI-COLON TYPED?
11857 063630  001430                          BEQ     O.ERR1          ;NO, CALL IT AN ERROR
11858 063632  163705  061700                  SUB     O.CAD,R5        ;COMPUTE
11859 063636  005305                          DEC     R5
11860 063640  005305                          DEC     R5              ;    16 BIT OFFSET
11861 063642  010500                          MOV     R5,R0
11862 063644  004537  064754                  JSR     5,O.CADV        ;NUMBER IN R0 - WORD MODE
11863 063650  010500                          MOV     R5,R0
11864 063652  006200                          ASR     R0              ;DIVIDE BY TWO
11865 063654  103414                          BCS     1$              ;ERROR IF ODD
11866 063656  022700  177600                  CMP     #-200,R0        ;COMPARE WITH -200
11867 063662  003011                          BGT     1$              ;DO NOT TYPE IF OUT OF RANGE
11868 063664  022700  000177                  CMP     #177,R0         ;COMPARE WITH +177
11869 063670  002406                          BLT     1$              ;DO NOT TYPE IF OUT OF RANGE
11870 063672  005337  061706                  DEC     O.BW            ;SET TEMPORARY BYTE MODE
11871 063676  004537  064754                  JSR     5,O.CADV        ;NUMBER IN R0 - BYTE MODE
11872 063702  005237  061706                  INC     O.BW            ;RESTORE WORD MODE
11873 063706  000137  062650          1$:     JMP     O.DCD3          ;ALL DONE
11874
11875 063712  000137  062630          O.ERR1: JMP     O.ERR           ;INTERMEDIATE HELP
```

```
 11878                                  ; PROCESS G - GO
 11879
 11880 063716  105737  061721           O.GO:   TSTB    O.SMFD          ;WAS ":" TYPED?
 11881 063722  001773                           BEQ     O.ERR1          ;BR IF NOT TYPED
 11882 063724  005703                           TST     R3              ;WAS K; TYPED?
 11883 063726  001771                           BEQ     O.ERR1          ; TYPE ?<CR,LF> IF NOT
 11884 063730  112737  000021  061713           MOVB    #O.BKP+3,O.P    ;CLEAR PROCEED
 11885 063736  006205                           ASR     R5              ;CHECK LOW ORDER BIT
 11886 063740  103764                           BCS     O.ERR1          ;ERROR IF ODD NUMBER
 11887 063742  006305                           ASL     R5              ;RESTORE WORD
 11888 063744  010537  062020           O.GOGO: MOV     R5,O.UPC        ;SET UP NEW PC
 11889 063750  012746  000340                   MOV     #O.STM,-(SP)    ;SET HIGH PRIORITY
 11890 063754  004737  065322                   JSR     PC,O.WST
 11891 063760  004537  064654                   JSR     5,O.RSTT        ;RESTORE TELETYPE
 11892 063764  105037  061712           O.TBIT: CLRB    O.T             ;CLEAR
 11893 063770  052737  000020  062022           BIS     #O.TBT,O.UST    ;     BOTH T-BIT FLAGS
 11894 063776  105737  061711                   TSTB    O.S             ;SEE IF WE NEED A T BIT
 11895 064002  001005                           BNE     O.GO2           ;IF NOT GO NOW
 11896 064004  042737  000020  062022           BIC     #O.TBT,O.UST    ;CLEAR THE T BIT
 11897 064012  004537  064570                   JSR     5,O.RSB         ;RESTORE BREAKPOINTS
 11898 064016  004037  064536           O.GO2:  JSR     0,O.RSR         ;RESTORE REGISTERS
 11899 064022  013746  062022                   MOV     O.UST,-(SP)     ;     AND STATUS
 11900 064026  013746  062020                   MOV     O.UPC,-(SP)     ;     AND PC
 11901 064032  000240                           NOP                     ; CHANGE TO HALT FOR DEBUGGING
 11902 064034  013746  062022                   MOV     O.UST,-(SP)     ; MOV IN STATUS FIRST W/O T BIT
 11903 064040  042716  177420                   BIC     #O.TBT!177400,(SP);  SO INTERRUPTS CAN HAPPEN BEFORE
 11904 064044  004737  065322                   JSR     PC,O.WST        ;  RTT TURNS ON THE T BIT.
 11905 064050  000006                           RTT
```

```
11908                                   ; PROCESS P - PROCEED
11909                                   ;   ONLY ALLOWED AFTER A BREAKPOINT
11910
11911 064052  105737  061721            O.PROC: TSTB    O.SMFD          ;WAS ";" TYPED?
11912 064056  001715                            BEQ     O.ERR1          ;BR IF NOT TYPED
11913 064060  113700  061713                    MOVB    O.P,R0
11914 064064  105700                            TSTB    R0              ;CHECK LEGALITY OF PROCEED
11915 064066  002711                            BLT     O.ERR1          ;NOT LEGAL
11916 064070  005702                            TST     R2              ;CHECK FOR ILLEGAL COUNT
11917 064072  001307                            BNE     O.ERR1          ;JUMP IF ILLEGAL
11918 064074  005703                            TST     R3              ;WAS COUNT SPECIFIED?
11919 064076  001402                            BEQ     O.PR1           ;NO
11920 064100  010560  062050                    MOV     R5,O.CT(R0)     ;YES, PUT AWAY COUNT
11921 064104  012746  000340            O.PR1:  MOV     #O.STM,-(SP)    ;FORCE HIGH PRIORITY
11922 064110  004737  065322                    JSR     PC,O.WST
11923 064114  004537  064654                    JSR     5,O.RSTT        ;RESTORE TTY
11924 064120  123727  061713  000016    O.C1:   CMPB    O.P,#O.BKP      ;SEE IF A REAL ONE OR A FAKE
11925 064126  003316                            BGT     O.TBIT          ;BRANCH IF FAKE
11926 064130  105737  061711                    TSTB    O.S             ;SEE IF SINGLE INSTRUCTION MODE
11927 064134  001313                            BNE     O.TBIT          ;IF SO EXIT NOW
11928 064136  012746  000340                    MOV     #O.STM,-(SP)    ;SET HIGH PRIORITY
11929 064142  004737  065322                    JSR     PC,O.WST
11930 064146  105237  061712                    INCB    O.T             ;SET T-BIT FLAG
11931 064152  052737  000020  062022            BIS     #O.TBT,O.UST    ;SET T-BIT
11932 064160  000716                            BR      O.GO2
```

```
11935                                   ; BREAKPOINT HANDLER
11936 064162  012637  062020          0.BRK:  MOV     (SP)+,0.UPC     ;PRIORITY IS 7 UPON ENTRY
11937 064166  012637  062022                  MOV     (SP)+,0.UST     ;SAVE STATUS AND PC
11938 064172  112737  000021  061713          MOVB    #0.BKP+3,0.P    ;TELL ;P THAT WE CAN CONTINUE
11939 064200  004037  064500          0.BK1:  JSR     0,0.SVR         ;SAVE VARIOUS REGISTERS
11940 064204  105737  061712                  TSTB    0.T             ;CHECK FOR T-BIT SET
11941 064210  001265                          BNE     0.TBIT          ;JUMP IF SET
11942 064212  004537  064722                  JSR     5,0.REM         ;REMOVE BREAKPOINTS
11943 064216  105737  062024                  TSTB    0.PRI           ;CHECK IF PRIORITY
11944 064222  100003                          BPL     2$              ; IS AS SAME AS USER PGM
11945 064224  113705  062022                  MOVB    0.UST,R5        ;PICK UP USER UST IF SO
11946 064230  000407                          BR      3$
11947 064232  113705  062024          2$:     MOVB    0.PRI,R5        ;OTHERWISE PICK UP ACTUAL PRIORITY
11948 064236  000257                          CCC                     ;CLEAR CARRY
11949 064240  106005                          RORB    R5              ;SHIFT LOW ORDER BITS
11950 064242  106005                          RORB    R5              ;  INTO
11951 064244  106005                          RORB    R5              ;    HIGH ORDER
11952 064246  106005                          RORB    R5              ;      POSITION
11953 064250  042705  177420          3$:     BIC     #0.TBT!177400,R5;CLEAR POSSIBLE T BIT (S/I MODE)
11954 064254  010546                          MOV     R5,-(SP)        ;PUT THE STATUS AWAY WHERE IT BELONGS
11955 064256  004737  065322                  JSR     PC,0.WST
11956 064262  013705  062020                  MOV     0.UPC,R5        ;GET PC, IT POINTS TO THE TRT
11957 064266  105737  061711                  TSTB    0.S             ;SEE IF IT WAS SINGLE INSTRUCTION FUN
11958 064272  100432                          BMI     14$             ;IF SO HANDLE THERE
11959 064274  005745                          TST     -(R5)
11960 064276  010537  062020                  MOV     R5,0.UPC
11961 064302  012704  000016                  MOV     #0.BKP,R4       ;GET A COUNTER
11962 064306  020564  062026          11$:    CMP     R5,0.ADR1(R4)   ;COMPARE WITH LIST
11963 064312  001426                          BEQ     12$             ;JUMP IF FOUND
11964 064314  005304                          DEC     R4
11965 064316  005304                          DEC     R4
11966 064320  002372                          BGE     11$             ;RE-LOOP UNTIL FOUND
11967 064322  004537  064620                  JSR     5,0.SVTT        ;SAVE TELETYPE STATUS
11968 064326  004537  065366                  JSR     5,0.CRLF
11969 064332  012704  065506                  MOV     #0.BD,R4        ;ERROR, NOTHING FOUND
11970 064336  012703  065507                  MOV     #0.BD+1,R3
11971 064342  004537  065246                  JSR     5,0.TYPE        ;OUTPUT 'BE'' FOR BAD ENTRY
11972 064346  010500                          MOV     R5,R0
11973 064350  062737  000002  062020          ADD     #2,0.UPC        ;POP OVER THE ADJUSTMENT ABOVE
11974 064356  000444                          BR      13$             ; OR CONTINUE
```

```
11977 064360  112704  000020          14$:   MOVB   #0.BKP+2,R4      ;SET BREAK POINT HIGH + 1
11978 064364  010564  062026                 MOV    R5,0.ADR1(R4)    ;STORE NEXT PC VALUE FOR TYPE OUT
11979 064370  110437  061713          12$:   MOVB   R4,0.P           ;ALLOW PROCEED
11980 064374  005364  062050                 DEC    0.CT(R4)
11981 064400  003247                        BGT    0.C1             ;JUMP IF REPEAT
11982 064402  012764  000001  062050        MOV    #1,0.CT(R4)      ;RESET COUNT TO 1
11983 064410  004537  064620                 JSR    5,0.SVTT         ;SAVE TELETYPE STATUS, R4 IS SAFE
11984 064414  012700  000102                 MOV    #'B,R0
11985 064420  004537  065050                 JSR    5,0.FTYP         ;TYPE 'B'
11986 064424  113700  061713                 MOVB   0.P,R0           ;CONVERT BREAKPOINT NUMBER TO ASCII
11987 064430  062700  000140                 ADD    #140,R0
11988 064434  006200                        ASR    R0
11989 064436  004537  065050                 JSR    5,0.FTYP
11990 064442  012700  000073                 MOV    #';,R0
11991 064446  004537  065050                 JSR    5,0.FTYP         ; TYPE
11992 064452  012737  000002  061706        MOV    #2,0.BW          ; SET WORD MODE
11993 064460  113704  061713                 MOVB   0.P,R4
11994 064464  016400  062026                 MOV    0.ADR1(R4),R0    ;GET ADDRESS OF BREAK
11995 064470  004537  065412          13$:   JSR    5,0.RORA         ; CHECK FORMAT
11996 064474  000137  062640                 JMP    0.DCD            ;GO TO DECODER
```

```
11999                                   ; SAVE REGISTERS R0-R6
12000                                   ;   INTERNAL STACK
12001
12002 064500  012637  061704      0.SVR:  MOV     (SP)+,0.XXX     ;PICK REGISTER FROM STACK AND SAVE
12003 064504  010637  062016              MOV     SP,0.USP        ;SAVE USER STACK ADDRESS
12004 064510  012706  062016              MOV     #0.USP,SP       ;SET TO INTERNAL STACK
12005 064514  010546                      MOV     R5,-(SP)        ;SAVE
12006 064516  010446                      MOV     R4,-(SP)        ; REGISTERS
12007 064520  010346                      MOV     R3,-(SP)        ;1
12008 064522  010246                      MOV     R2,-(SP)        ; THRU
12009 064524  010146                      MOV     R1,-(SP)        ;     5
12010 064526  013746  061704              MOV     0.XXX,-(SP)     ;PUT SAVED REGISTER ON STACK
12011 064532  005746                      TST     -(SP)
12012 064534  000200                      RTS     R0
12013
12014                                   ; RESTORE REGISTERS R0-R6
12015
12016 064536  005726              0.RSR:  TST     (SP)+           ;POP THE EXTRA CELL
12017 064540  012637  061704              MOV     (SP)+,0.XXX     ;GET R0 FROM STACK
12018 064544  012601                      MOV     (SP)+,R1        ;RESTORE
12019 064546  012602                      MOV     (SP)+,R2        ; REGISTERS
12020 064550  012603                      MOV     (SP)+,R3        ;   1
12021 064552  012604                      MOV     (SP)+,R4        ;   THRU
12022 064554  012605                      MOV     (SP)+,R5        ;     5
12023 064556  013706  062016              MOV     0.USP,SP        ;RESTORE USER STACK
12024 064562  013746  061704              MOV     0.XXX,-(SP)     ;PUT R0 ON USER STACK
12025 064566  000200                      RTS     R0
```

```
12028                                   ; RESTORE BREAKPOINTS 0-7
12029
12030 064570  012704  000016          0.RSB:  MOV     #0.BKP,R4       ;RESTORE ALL BREAKPOINTS
12031 064574  017464  062026  062072  1$:     MOV     @0.ADR1(R4),0.UIN(R4);SAVE CONTENTS
12032 064602  013774  065510  062026          MOV     0.TRTC,@0.ADR1(R4);REPLACE WITH TRAP
12033 064610  005304                          DEC     R4
12034 064612  005304                          DEC     R4
12035 064614  002367                          BGE     1$              ;RE-LOOP UNTIL DONE
12036 064616  000205                          RTS     R5              ;  THEN QUIT
12037
12038                                   ; SAVE TELETYPE STATUS
12039
12040 064620  113737  177560  061714  0.SVTT: MOVB    0.RCSR,0.CSR1   ;SAVE R C/SR
12041 064626  113737  177564  061715          MOVB    0.TCSR,0.CSR2   ;SAVE T C/SR
12042 064634  105037  177560                  CLRB    0.RCSR          ;CLEAR ENABLE AND MAINTENANCE
12043 064640  105037  177564                  CLRB    0.TCSR          ;  BITS IN BOTH C/SR
12044 064644  105737  177564          1$:     TSTB    0.TCSR          ;LOOP UNTIL READY BIT COMES ON
12045 064650  100375                          BPL     1$              ;BR IF BIT NOT ON
12046 064652  000205                          RTS     R5
12047                                   ; RESTORE TELETYPE STATUS
12048
12049 064654  004537  065366          0.RSTT: JSR     5,0.CRLF
12050 064660  105737  177564                  TSTB    0.TCSR          ;WAIT READY
12051 064664  100375                          BPL     .-4             ;  ON PRINTER
12052 064666  032737  004000  177560          BIT     #4000,0.RCSR    ;CHECK BUSY FLAG
12053 064674  001403                          BEQ     1$              ;SKIP READY LOOP IF NOT BUSY
12054 064676  105737  177560                  TSTB    0.RCSR          ;WAIT READY
12055 064702  100375                          BPL     .-4             ;    ON READER
12056 064704  113737  061714  177560  1$:     MOVB    0.CSR1,0.RCSR   ;RESTORE
12057 064712  113737  061715  177564          MOVB    0.CSR2,0.TCSR   ;  THE STATUS REGISTERS
12058 064720  000205                          RTS     R5
12059
12060                                   ; REMOVE BREAKPOINTS 0-7
12061                                   ;   IN THE OPPOSITE ORDER OF SETTING
12062
12063 064722  105737  061711          0.REM:  TSTB    0.S             ;SEE IF SINGLE INSTRUCTION IS GOING
12064 064726  001011                          BNE     2$              ;EXIT IF SO
12065 064730  005004                          CLR     R4              ;REMOVE ALL BREAKPOINTS
12066 064732  016474  062072  062026  1$:     MOV     0.UIN(R4),@0.ADR1(R4)   ;CLEAR BREAKPOINT
12067 064740  005204                          INC     R4
12068 064742  005204                          INC     R4
12069 064744  020427  000016                  CMP     R4,#0.BKP
12070 064750  003770                          BLE     1$              ;RE-LOOP UNTIL DONE
12071 064752  000205                  2$:     RTS     R5              ;THEN QUIT
```

```
12074                                   ; TYPE OUT CONTENTS OF WORD OR BYTE WITH ONE TRAILING SPACE
12075                                   ; WORD IS IN R0
12076
12077 064754  012703  000006          O.CADV: MOV     #6,R3           ;# OF DIGITS
12078 064760  012704  177776                  MOV     #-2,R4          ;# OF BITS FIRST-3
12079 064764  022737  000001  061706          CMP     #1,O.BW         ;SEE IF WORD MODE
12080 064772  001004                          BNE     3$              ;BRANCH IF SO
12081 064774  162703  000003                  SUB     #3,R3           ;ONLY DO 3 DIGITS
12082 065000  005204                          INC     R4              ;DO 2 BITS FIRST
12083 065002  000300                          SWAB    R0              ;AND TURN R0 AROUND
12084 065004  010046                  3$:     MOV     R0,-(SP)        ;SAVE R0
12085 065006  062704  000003          2$:     ADD     #3,R4           ;COMPUTE THE NUMBER OF BITS TO DO
12086 065012  005000                          CLR     R0
12087 065014  006116                  1$:     ROL     (SP)            ;GET A BIT
12088 065016  006100                          ROL     R0              ;STORE IT AWAY
12089 065020  005304                          DEC     R4              ;DECREMENT COUNTER
12090 065022  003374                          BGT     1$              ;LOOP IF MORE BITS NEEDED
12091 065024  062700  000060                  ADD     #'0,R0          ;CONVERT TO ASCII
12092 065030  004537  065050                  JSR     R5,O.FTYP       ;TYPE IT
12093 065034  005303                          DEC     R3              ;SEE IF MORE DIGITS TO DO
12094 065036  003363                          BGT     2$              ;LOOP IF SO
12095 065040  112700  000040                  MOVB    #' ,R0          ;SET UP FOR TRAILING SPACE
12096 065044  005726                          TST     (SP)+           ;GET RID OF JUNK
12097 065046  000400                          BR      O.FTYP
12098
12099
12100                                   ; TYPE ONLY ONE CHARACTER (CONTAINED IN R0)
12101
12102 065050  105737  177564          O.FTYP: TSTB    O.TCSR
12103 065054  100375                          BPL     O.FTYP
12104 065056  042700  177400                  BIC     #177400,R0      ;CLEAR HIGH BYTE,SHOULD NOT
12105                                                                 ;CONTAIN INPORTANT INFO.
12106 065062  001416                          BEQ     3$
12107 065064  022700  000176                  CMP     #176,R0         ;PRINT ? FOR 177;16-37
12108 065070  103411                          BLO     2$              ; 1-10 AND 200-377.
12109 065072  022700  000037                  CMP     #37,R0
12110 065076  103410                          BLO     3$
12111 065100  022700  000015                  CMP     #15,R0
12112 065104  103403                          BLO     2$
12113 065106  022700  000010                  CMP     #10,R0
12114 065112  103402                          BLO     3$
12115 065114  012700  000077          2$:     MOV     #'?,R0
12116 065120  110037  177566          3$:     MOVB    R0,O.TDB
12117 065124  000205                O.TYP1: RTS     R5
```

```
12120                                   ; GENERAL CHARACTER INPUT ROUTINE -- ODT11X
12121                                   ; CHARACTER INPUT GOES TO R0
12122
12123 065126  105737  177560      O.GET:  TSTB    O.RCSR          ;WAIT FOR
12124 065132  100375                      BPL     O.GET           ;  INPUT FROM KBD
12125 065134  113700  177562              MOVB    O.RDB,R0        ;GET CHARACTER - STRIP OFF PARITY
12126 065140  042700  177600              BIC     #177600,R0      ;STRIP OFF PARITY FROM CHARACTER
12127 065144  122700  000003              CMPB    #3,R0           ;IS IT ^C?
12128 065150  001435                      BEQ     3$              ;IF SO, DO NOT ECHO
12129 065152  122700  000006              CMPB    #6,R0           ;IS IT ^F?
12130 065156  001432                      BEQ     3$              ;IF SO, DO NOT ECHO
12131 065160  122700  000005              CMPB    #5,R0           ;IS IT ^E?
12132 065164  001427                      BEQ     3$              ;IF SO, DO NOT ECHO
12133 065166  105737  061722              TSTB    O.SCRN          ;SHOULD WE ECHO <LF>?
12134 065172  001003                      BNE     2$              ;BR IF YES
12135 065174  120027  000012              CMPB    R0,#012         ;SEE IF A <LF>
12136 065200  001421                      BEQ     3$              ;IF SO SAVE THE PAPER
12137 065202  120027  000173      2$:     CMPB    R0,#173
12138 065206  002005                      BGE     1$
12139 065210  122700  000140              CMPB    #140,R0         ;TEST FOR LOWER CASE
12140 065214  002002                      BGE      1$
12141 065216  162700  000040              SUB     #40,R0          ;CHAR IS LC-CONVERT TO UPPER CASE
12142 065222  004537  065050      1$:     JSR     5,O.FTYP        ;ECHO CHARACTER
12143 065226  001737                      BEQ     O.GET           ;IGNORE NULLS
12144 065230  105737  061722              TSTB    O.SCRN          ;SHOULD WE PASS ON SPACES?
12145 065234  001003                      BNE     3$              ;BR IF YES
12146 065236  122700  000040              CMPB    #40,R0          ;CHECK FOR SPACES
12147 065242  001731                      BEQ     O.GET           ;IGNORE SPACES
12148 065244  000205              3$:     RTS     R5
```

```
12151                                   ; GENERAL CHARACTER OUTPUT ROUTINE - ODT11X
12152                                   ;  ADDRESS OF FIRST BYTE IN R4,
12153                                   ;  ADDRESS OF LAST BYTE IN R3, (R3)>(R4)
12154                                   ; EXPECTS LOCS 56,57 TO BE INITIALIZED BY MONITOR FOR FILL
12155                                   ; CHARACTERISTICS OF TERMINAL
12156                                   ;  56=CHAR TO BE FILLED AFTER
12157                                   ;  57=# OF NULLS TO FILL WITH
12158
12159 065246  020304                   O.TYPE: CMP     R3,R4           ;CHECK FOR COMPLETION
12160 065250  103725                           BLO     O.TYP1          ; EXIT WHEN DONE
12161 065252  112400                           MOVB    (R4)+,R0        ;GET A CHARACTER
12162 065254  004537  065050                   JSR     5,O.FTYP        ;TYPE ONE CHARACTER
12163 065260  120037  002612                   CMPB    R0,@#$FILLC     ;COMPARE CHAR AGAINST FILL REQUIREMENT
12164 065264  001370                           BNE     O.TYPE          ;NO FILL NEEDED
12165 065266  113737  002327  061716           MOVB    @#$FILLS,O.FIL  ;FILL COUNT INTO TEMP
12166 065274  005000                           CLR     R0              ;FILL WITH NULLS
12167 065276  004537  065050           2$:     JSR     R5,O.FTYP       ;TYPE NULLS
12168 065302  105337  061716                   DECB    O.FIL           ;DECREASE COUNT
12169 065306  003373                           BGT     2$              ;BRANCH IF NOT DONE
12170 065310  000756                           BR      O.TYPE          ;LOOP UNTIL DONE
```

```
12173                                   ;SUBROUTINE TO READ THE PS INDEPENDENT OF MACHINE
12174
12175 065312  013737  177776  062022  O.RRST: MOV     ST,O.UST        ;STORE THE STATUS IN USER
12176                                                                 ;STATUS AREA.FOR AN LSI
12177                                                                 ;THIS IS CHANGED TO
12178                                                                 ;MFPS O.UST
12179 065320  000207                          RTS     PC
12180
12181                                   ;SUBROUTINE TO WRITE PS INDEPENDENT OF MACHINE
12182                                   ;CALL ROUTINE WITH PS VALUE
12183                                   ;ON THE STACK
12184
12185 065322  016637  000002  177776  O.WST:  MOV     2(SP),ST        ;STORE NEW PS VALUE
12186                                                                 ;THIS INSTRUCTION IS CHANGED FOR LSI
12187                                                                 ;TO MTPS 2(SP)
12188 065330  012616                          MOV     (SP)+,(SP)      ;PUT RETURN PC OVER SUB. ARGUMENT
12189 065332  000207                          RTS     PC              ;TO RETURN WITHOUT IT ON THE STACK
12190                                   ; CLOSE WORD OR BYTE AND EXIT,
12191                                   ; UPON ENTERING, R2 HAS NUMERIC FLAG, R4 HAS CONTENTS
12192
12193                                           .ENABL  LSB
12194 065334  005702                  O.CLSE: TST     R2              ;IF NO NUMBER WAS TYPED THERE IS
12195 065336  001412                          BEQ     1$              ;NO CHANGE TO THE OPEN CELL
12196 065340  022737  000001  061706          CMP     #1,O.BW
12197 065346  001404                          BEQ     2$              ;JUMP IF BYTE MODE
12198 065350  101005                          BHI     1$              ;JUMP IF ALREADY CLOSED
12199 065352  010477  174322                  MOV     R4,@O.CAD       ;STORE WORD
12200 065356  000402                          BR      1$
12201 065360  110477  174314          2$:     MOVB    R4,@O.CAD       ;STORE BYTE
12202 065364  000207                  1$:     RTS     PC
12203 065366  012703  065435          O.CRLF: MOV     #O.CR+1,R3      ;LWA <CR,LF>
12204 065372  000402                          BR      3$
12205 065374  012703  065441          O.CRLS: MOV     #O.CR+5,R3      ;LWA <CR,LF>ODT>
12206 065400  012704  065434          3$:     MOV     #O.CR,R4        ;FWA
12207 065404  004537  065246                  JSR     5,O.TYPE        ;TYPE SOMETHING
12208 065410  000205                          RTS     R5
12209                                           .DSABL  LSB
```

```
12212                                   ;SUBROUTINE O.RORA
12213                                   ;FUNCTION:                              THE
12214                                   ;ADDRESS WILL BE PRINTED IN ABSOLUTE FORM
12215                                   ;INPUT: THE ADDRESS TO BE PRINTED IS IN R0.
12216                                   ;DATA SAVED:  R0 CONTAINING THE ADDRESS TO BE
12217                                   ;PRINTED, AND LOCATION O.CAD CONTAINING
12218                                   ;THE CURRENT ADDRESS WERE SAVED AND RESTORED.
12219                                   ;CALLED:  JSR 5,O.RORA
12220
12221 065412  004537  064754           O.RORA: JSR     5,O.CADV        ;PRINT ABSOLUTE ADDRESS
12222 065416  000205                           RTS     R5
12223 065420     012                   O.ID:   .BYTE   012
12224 065421     015                           .BYTE   015
12225 065422     040                           .BYTE   40
12226 065423     103      132      115         .ASCII ''CZMSD ODT''
      065426     123      104      040
      065431     117      104      124
12227           065433                 O.IDND=.-1
12228 065434     015                   O.CR:   .BYTE   015     ;  <CR>
12229 065435     012                           .BYTE   012     ;  <LF>
12230 065436     117                           .BYTE   'O      ;  O
12231 065437     104                           .BYTE   'D      ;  D
12232 065440     124                           .BYTE   'T      ;  T
12233 065441     076                           .BYTE   '>      ;  >
12234
12235 065442     073                   O.LGCH: .BYTE   ';      ;
12236 065443     057                           .BYTE   '/      ;  /
12237 065444     134                           .BYTE   '\      ;  \     (BACK SLASH)
12238 065445     015                           .BYTE   015     ;  CARRIAGE RETURN
12239 065446     044                           .BYTE   '$      ;  $
12240 065447     107                           .BYTE   'G      ;  G
12241 065450     012                           .BYTE   012     ;  <LF>
12242 065451     137                           .BYTE   '_      ;       (BACK ARROW)
12243 065452     074                           .BYTE   '<      ;  <
12244 065453     136                           .BYTE   '^      ;  ^     (UP ARROW)
12245 065454     117                           .BYTE   'O      ;  O
12246 065455     102                           .BYTE   'B      ;  B
12247 065456     120                           .BYTE   'P      ;  P
12248 065457     100                           .BYTE   '@      ;  @
12249 065460     076                           .BYTE   '>      ;  >
12250 065461     123                           .BYTE   'S      ;  S
12251 065462     055                           .BYTE   '-      ;  -
12252 065463     003                           .BYTE   003     ;CTRL C
12253 065464     006                           .BYTE   006     ;CTRL F
12254 065465     005                           .BYTE   005     ;CTRL E
12255           000024                 O.CLGT  =       .-O.LGCH                ;TABLE LENGTH
12256
12257 065466     123                   O.TL:   .BYTE   'S      ;DO        1
12258 065467     120                           .BYTE   'P      ;NOT       2
12259 065470     115                           .BYTE   'M      ;CHANGE    3
12260 065471     000                           .BYTE   0       ;THE       4
12261 065472     000                           .BYTE   0       ;ORDER     5
12262 065473     103                           .BYTE   'C      ;          6
12263 065474     106                           .BYTE   'F      ;          7
12264 065475     122                           .BYTE   'R      ;         10
12265 065476     000                           .BYTE   0       ;         11
12266 065477     000                           .BYTE   0       ;         12
```

```
 12267 065500  000                        .BYTE   0       ;       13
 12268 065501  000                        .BYTE   0       ;       14
 12269 065502  000                        .BYTE   0       ;       15
 12270 065503  000                        .BYTE   0       ;       16
 12271 065504  000                        .BYTE   0       ;       17
 12272 065505  102                        .BYTE   'B      ;       20
 12273         000020          O.LG       =       .-O.TL
 12274                                    .EVEN
 12275 065506  042502          O.BD:     .WORD   'BE
 12276 065510  000003          O.TRTC:   TRT                     ;TRACE TRAP PROTOTYPE
```

```
12279                                   .SBTTL  TABLE   ERROR POINTER
12280
12281                           ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
12282                           ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
12283                           ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
12284                           ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS (ERRPC).
12285                           ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
12286
12287                           ;*      EM              ;;POINTS TO THE ERROR MESSAGE
12288                           ;*      DH              ;;POINTS TO THE DATA HEADER
12289                           ;*      DT              ;;POINTS TO THE DATA
12290                           ;*      DF              ;;POINTS TO THE DATA FORMAT
12291
12292
12293 065512                   $ERRTB: ;ERROR  1
12294 065512  070003                   EM24
12295 065514  072130                   DH13
12296 065516  066362                   DT13
12297 065520  066720                   DF11
12298                                  ;ERROR  2
12299 065522  066757                   EM2
12300 065524  071437                   DH1
12301 065526  066212                   DT1
12302 065530  066576                   DF2
12303                                  ;ERROR  3
12304 065532  067015                   EM3
12305 065534  071517                   DH3
12306 065536  066224                   DT3
```

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

CZMSDA0 MS11-L/M DIAGNOSTIC MACRO M1110 12-DEC-79 16:25
TABLE OF CONTENTS SEQ 0378

```
                        .TITLE  CZMSDA0 MS11-L/M DIAGNOSTIC
                        .REM    &


                                        IDENTIFICATION
                                        --------------

                          PRODUCT CODE:

                          PRODUCT NAME:         CZMSDA0 MS11-L/M MEMORY DIAGNOSTIC

                          PRODUCT DATE:         DECEMBER 1979

                          MAINTAINER:           DIAGNOSTIC ENGINEERING

                          THE INFORMATION  IN  THIS  DOCUMENT  IS  SUBJECT  TO  CHANGE
                          WITHOUT  NOTICE  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT
                          BY  DIGITAL  EQUIPMENT   CORPORATION.    DIGITAL   EQUIPMENT
                          CORPORATION  ASSUMES  NO  RESPONSIBILITY FOR ANY ERRORS THAT
                          MAY APPEAR IN THIS MANUAL.

                          THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO  THE
                          PURCHASER  UNDER  A  LICENSE  FOR  USE  ON A SINGLE COMPUTER
                          SYSTEM  AND  CAN  BE  COPIED  (WITH  INCLUSION  OF DIGITAL'S
                          COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY
                          OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

                          DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY  FOR
                          THE  USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS
                          NOT SUPPLIED BY DIGITAL.

                          COPYRIGHT (C)  1979 DIGITAL EQUIPMENT CORPORATION
                        &
```

```
                        ;                       REVISION HISTORY
                        ;                       ================
                        ;
                        ;  REVISION   VERSION     DATE            AUTHOR            CHANGES
                        ;  ========   =======   =========     ===============      =======
                        ;    CZMSDA     V00.00    1-DEC-79    MICHAEL D. BIBEAULT  NONE=NEW PROGRAM
                        ;
```

```
                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        .SBTTL ;SWITCH REGISTER DEFINITIONS
                        .SBTTL ;*
                        .SBTTL ;*       SWITCH              USE
                        .SBTTL ;*       ------      --------------------
                        .SBTTL ;*         15        HALT ON ERROR
                        .SBTTL ;*         14        LOOP ON TEST
                        .SBTTL ;*         13        INHIBIT ERROR TYPEOUTS
                        .SBTTL ;*         12        INHIBIT RELOCATION
                        .SBTTL ;*         11        QUICK VERIFY
                        .SBTTL ;*         10        BELL ON ERROR
                        .SBTTL ;*          9        LOOP ON ERROR
                        .SBTTL ;*          8        HALT PROGRAM (UNRELOCATED & RESTORE LOADERS)
                        .SBTTL ;*          7        DETAILED ERROR REPORTS
                        .SBTTL ;*          6        INHIBIT CONFIGURATION MAP
                        .SBTTL ;*          5        LIMIT MAX ERRORS PER BANK
                        .SBTTL ;*          4        FAT TERMINAL (132 COLUMNS OR BETTER)
                        .SBTTL ;*          3        TEST MODE - SEE DOCUMENT
                        .SBTTL ;*          2        TEST MODE - SEE DOCUMENT
                        .SBTTL ;*          1        TEST MODE - SEE DOCUMENT
                        .SBTTL ;*          0        DETECT SINGLE BIT ERRORS
```

```
     72 000000                          .ENABL  ABS
     73                                 .ENABL  AMA
     74                                 .DSABL  GBL
     75                                 ;NOTE:  CZMSDA.SML IS THE SUPER.MAC SOURCE AND IS RELEASED WITH
     76                                 ;THIS PROGRAM.  ALL THESE .MCALL STATEMENTS REFERENCE THAT FILE.
     77                                 .MCALL  SMACIT,..PUSH,..POP,..TAG,..BRAN,..EMIT,..EMITN,..EMITL,..EMITR
     78                                 .MCALL  .IFOPR,.IS,..GENBR,..OPADD,..OPSUB,CLEAR,SET,CLEARB,SETB
     79                                 .MCALL  RNE,REQ,RLT,RGE,RGT,RLE,RPL,RMI,RHI,RLOS,RHIS,RLO,RCS,RCC
     80                                 .MCALL  IF,..OR,..IFARI,..LEAVE,..GOTO,OR,AND,THEN,ELSE,WHILE,CASE
     81                                 .MCALL  FOR,TO,DOWNTO,REPEAT,UNTIL,THRU,END,BEGIN
     82                                 .MCALL  $$END,LEAVE,JUMPTO,GOTO,PUSH,POP,LET
     83                                 .MCALL  .SIMPLE,..ARITH,ORB,ANDB,IFB,UNTILB,WHILEB,ON.ERROR,ON.NOERROR
     84                                 .MCALL  $CALL,$RETURN
     85
     86                                 .NLIST  TTM                     ;I WANT FAT PAPER!
     87                                 .LIST   MC,SYM                  ;LIST MACRO CALLS, SYMBOL TABLE
     88                                 .NLIST  MD,CND,ME               ;DON'T LIST MACRO DEFS & CONDITIONALS & EXPANSIONS
     89                                 ;LST$$= 0                       ;DEFINED TO LIST SUPERMAC EXPANSIONS
     90          163000                 $SWR=   163000                  ;USE THESE SYSMAC SWITCHES
     91          000001                 $TN=    1                       ;FIRST TEST NUMBER TO ONE(1)
     92 000000                          SMACIT
```

```
                                .SBTTL  DEFINE  TRAPS
                        ;ALL ENTRIES HERE MUST HAVE A CORRESPONDING ENTRY IN THE
                        ;TRAP TABLE '$TRPAD'' (NEAR END OF PROGRAM).
                        ;*TRAP DEFINITIONS
                        ;
                        ;HERE IS HOW TRAPS WORK IN THIS PROGRAM
                        ;
                        ;ALL TRAPS EXECUTE A 'TRAP'' INSTRUCTION WHICH TAKES THE PROGRAM
                        ;TO SYMBOLIC LOCATION '$TRAP''
                        ;
                        ;AT $TRAP THE PROGRAM PICKS UP THE RIGHT BYTE OF THE TRAP INSTRUCTION
                        ;AND INDEXES INTO A TABLE AT LOCATION '$TRPAD'' WHICH SENDS THE PROGRAM TO
                        ;THE SPECIFIC ROUTINE TO HANDLE THAT SPECIFIC TRAPS TASK.
                        ;
                        ;THE ULTIMATE DESTINATION OF A TRAP INSTRUCTION CAN BE GUESSED AT AS FOLLOWS
                        ;
                        ;EXAMPLE:       NOP
                        ;               NOP
                        ;               NOP
                        ;               KERNEL                  ;ENTER KERNEL MODE
                        ;               NOP
                        ;
                        ;       ADD A DOLLAR SIGN TO THE SYMBOLIC NAME AND CHECK THE CRF FOR SOMETHING CLOSE
                        ;       IN THIS CASE THE CRF HAS $KERNE LISTED AS 032546
                        ;       AT LOCATION 32546 YOU FIND THE ROUTINE $KERNEL
                        ;
                        ;       NOTE THAT CRF SYMBOLS ARE TRUCNATED TO 6 CHARACTERS
                        ;       SYMBOLIC NAMES GREATER THAT 6 CHARACTERS ARE USED SO I CAN
                        ;       REMEMBER WHAT THEY MEAN!
                        ;
                        ;       ALL GOOD TRAP ROUTINES RETURN VIA AN 'RTI'' INSTRUCTION
104401                  TYPEIT= 104401          ;;TTY TYPEOUT ROUTINE
104402                  TYPOC=  104402          ;;TYPE OCTAL NUMBER (WITH LEADING ZEROS)
104403                  TYPOS=  104403          ;;TYPE OCTAL NUMBER (NO LEADING ZEROS)
                        ;TYPON= 104404          ;;TYPE OCTAL NUMBER (AS PER LAST CALL)
104405                  TYPDS=  104405          ;;TYPE DECIMAL NUMBER (WITH SIGN)
                        ;TYPBN= 104406          ;;TYPE BINARY (ASCII) NUMBER

104407                  GTSWR=  104407          ;;GET SOFT-SWR SETTING
104410                  CKSWR=  104410          ;;TEST FOR CHANGE IN SOFT-SWR

104411                  RDCHR=  104411          ;;TTY TYPEIN CHARACTER ROUTINE
104412                  RDLIN=  104412          ;;TTY TYPEIN STRING ROUTINE
104413                  RDOCT=  104413          ;;READ AN OCTAL NUMBER FROM TTY
104414                  RDDEC=  104414          ;;READ A DECIMAL NUMBER FROM TTY

104415                  SAVREG= 104415          ;;SAVE R0-R5 ROUTINE
104416                  RESREG= 104416          ;;RESTORE R0-R5 ROUTINE

104417                  KERNEL= 104417          ;ENTER KERNEL MODE

104420                  ENERGIZE=104420         ;TURN ON MEMORY MANAGEMENT & TRAPS
104421                  DEENERGIZE=104421       ;TURN OFF MEMORY MANAGEMENT & TRAPS
104422                  KMAP=   104422          ;MAP KERNEL 1 TO 1

104423                  CACHON= 104423          ;TURN ON CACHE
104424                  CACHOFF=104424          ;TURN OFF CACHE
```

```
    152
    153          104425                LOADCSR=104425          ;LOAD CORRECT CSR
    154          104426                READCSR=104426          ;READ CORRECT CSR
    155
    156          104427                PERR01= 104427          ;PROGRAM DETECTED ERROR
    157          104430                PERR02= 104430          ;PROGRAM DETECTED ERROR
    158          104431                PERR03= 104431          ;PROGRAM DETECTED ERROR
    159          104432                PERR04= 104432          ;PROGRAM DETECTED ERROR
    160          104433                PERR07= 104433          ;PROGRAM DETECTED ERROR
    161          104434                PERR10= 104434          ;PROGRAM DETECTED ERROR
    162          104435                PERR11= 104435          ;PROGRAM DETECTED ERROR
    163          104436                PERR12= 104436          ;PROGRAM DETECTED ERROR
    164          104437                PERR13= 104437          ;PROGRAM DETECTED ERROR
    165          104440                PERR14= 104440          ;PROGRAM DETECTED ERROR
    166          104441                PERR15= 104441          ;PROGRAM DETECTED ERROR
    167          104442                PERR16= 104442          ;PROGRAM DETECTED ERROR
    168          104443                PERR17= 104443          ;PROGRAM DETECTED ERROR
    169          104444                PERR20= 104444          ;PROGRAM DETECTED ERROR
    170          104445                PERR21= 104445          ;PROGRAM DETECTED ERROR
    171          104446                PERR22= 104446          ;PROGRAM DETECTED ERROR
    172          104447                PERR23= 104447          ;PROGRAM DETECTED ERROR
    173          104450                PERR24= 104450          ;PROGRAM DETECTED ERROR
    174          104451                PERR25= 104451          ;PROGRAM DETECTED ERROR
    175          104452                PERR26= 104452          ;PROGRAM DETECTED ERROR
    176          104453                PERR27= 104453          ;PROGRAM DETECTED ERROR
    177          104454                PERR30= 104454          ;PROGRAM DETECTED ERROR
    178          104455                PERR31= 104455          ;PROGRAM DETECTED ERROR
    179          104456                PERR32= 104456          ;PROGRAM DETECTED ERROR
    180          104457                PERR33= 104457          ;PROGRAM DETECTED ERROR
    181          104460                PERR34= 104460          ;PROGRAM DETECTED ERROR
    182          104461                PERR35= 104461          ;PROGRAM DETECTED ERROR
    183          104462                PERR36= 104462          ;PROGRAM DETECTED ERROR
    184          104463                PERR37= 104463          ;PROGRAM DETECTED ERROR
    185          104464                PERR40= 104464          ;PROGRAM DETECTED ERROR
    186          104465                PERR41= 104465          ;PROGRAM DETECTED ERROR
    187          104466                PERR42= 104466          ;PROGRAM DETECTED ERROR
    188          104467                PERR43= 104467          ;PROGRAM DETECTED ERROR
    189
    190          104470                ECCDIS= 104470          ;DISABLE ECC ON ALL CSR'S
    191          104471                ECC1DIS=104471          ;DISABLE ECC ON 1 SELECTED CSR
    192          104472                ECCINIT=104472          ;INITIALIZE ALL ECC CSR'S
    193          104473                ECC1INIT=104473         ;INITIALIZE 1 SELECTED ECC CSR
    194          104474                CBCSR=  104474          ;WRITE GENERATED CHECKBITS IN ALL CSR'S
    195          104475                CB1CSR= 104475          ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
    196          104476                WASSBE= 104476          ;WAS THERE A SBE ON ANY CSR?
    197          104477                WAS1SBE=104477          ;WAS THERE A SBE ON 1 SELECTED CSR?
    198          104500                WASDBE= 104500          ;WAS THERE A DBE ON ANY CSR?
    199          104501                WAS1DBE=104501          ;WAS THERE A DBE ON 1 SELECTED CSR?
    200          104502                CLRCSR= 104502          ;CLEAR ALL CSR'S
    201          104503                CLR1CSR=104503          ;CLEAR 1 SELECTED CSR
    202          104504                CHKDIS= 104504          ;DISABLE ECC & WRITE CHECKBITS FROM ALL CSR'S
    203          104505                CHK1DIS=104505          ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
    204          104506                ENASBE= 104506          ;ENABLE TRAPS ON SBE'S FROM ALL CSR'S
    205          104507                ENA1SBE=104507          ;ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
    206          104510                TSTREAD=104510          ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
    207          104511                INVALID=104511          ;INVALIDATE BACKGROUND PATTERN ON 'BANK''
    208          104512                ERRGEN =104512          ;CHECK ERROR ADDRESS
```

```
                                .SBTTL  DEFINE  BASIC PDP11 STUFF

                        ;*INITIAL ADDRESS OF THE STACK POINTER
002000                  STACK=  2000            ;;FIRST ADDRESS OF THE STACK
002000                  KERSTK= STACK           ;;KERNEL STACK
000740                  SUPSTK= 740             ;;SUPERVISOR STACK
000700                  USESTK= 700             ;;USER STACK
104000                  ERROR=EMT               ;;BASIC DEFINITION OF ERROR CALL
000004                  SCOPE=IOT               ;;BASIC DEFINITION OF SCOPE CALL
177776                  PSW=    177776          ;;PROCESSOR STATUS WORD
                        ;STKLMT=177774          ;;STACK LIMIT REGISTER
                        ;PIRQ=  177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
177546                  LKS=    177546          ;;LINE CLOCK (KW11-L) STATUS REGISTER

                        ;*MISCELLANEOUS DEFINITIONS
000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                  LF=     12              ;;CODE LINE FEED
000015                  CR=     15              ;;CODE CARRIAGE RETURN
000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED

                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
                        ;SP=R6                  ;;STACK POINTER
                        ;KSP=SP                 ;;KERNEL STACK POINTER
000006                  SSP=SP                  ;;SUPERVISOR STACK POINTER
000006                  USP=SP                  ;;USER STACK POINTER
                        ;PC=R7                  ;;PROGRAM COUNTER

                        ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW9=    1000
000400                  SW8=    400
000200                  SW7=    200
000100                  SW6=    100
000040                  SW5=    40
000020                  SW4=    20
000010                  SW3=    10
000004                  SW2=    4
000002                  SW1=    2
000001                  SW0=    1

                        ;*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                  BIT9=   1000
000400                  BIT8=   400
000200                  BIT7=   200
```

```
                        BIT6=   100
                        BIT5=   40
                        BIT4=   20
                        BIT3=   10
                        BIT2=   4
                        BIT1=   2
                        BIT0=   1

                        ;*BASIC ''CPU'' TRAP VECTOR ADDRESSES
000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
                        ;TBITVEC=14             ;;''T'' BIT
                        ;TRTVEC=        14              ;;TRACE TRAP
                        ;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
                        ;TPVEC= 64              ;;TTY PRINTER VECTOR
                        ;LKVEC= 100             ;;LINE CLOCK (KW11-L) VECTER
000114                  CACHVEC=114             ;;CACHE ERROR INTERRUPT VECTOR
000114                  PARVEC=CACHVEC
                        ;PIRQVEC=240            ;;PROGRAM INTERRUPT REQUEST VECTOR
000250                  MMVEC=  250             ;;MEMORY MANAGEMENT VECTOR
                                .SBTTL  DEFINE  CACHE REGISTERS
                        ;MEMERR = 177744                ;;CACHE ERROR REGISTER
177746                  CONTRL  = 177746                ;;MEMORY CONTROL REGISTER
177750                  MAINT   = 177750                ;;MEMORY MAINTENENCE REGISTER
                        ;HITMIS = 177752                ;;HIT MISS REGISTER ''1'' IMPLIES HIT IN CACHE
177754                  DATARG  = 177754                ;;DATA REGISTER

                                .SBTTL  DEFINE  CPU REGISTERS
177766                  CPUERR = 177766         ;;CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED

                                .SBTTL  DEFINE  MEMORY MANAGEMENT REGISTERS
                                ;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES
177572                  MMR0=   177572
177574                  MMR1=   177574
177576                  MMR2=   177576
172516                  MMR3=   172516

                                ;*USER ''I'' PAGE DESCRIPTOR REGISTERS
177600                  UIPDR0= 177600
                        ;UIPDR1=        177602
                        ;UIPDR2=        177604
                        ;UIPDR3=        177606
                        ;UIPDR4=        177610
                        ;UIPDR5=        177612
                        ;UIPDR6=        177614
                        ;UIPDR7=        177616

                                ;*USER ''D'' PAGE DESCRIPTOR REGISTORS
                        ;UDPDR0=        177620
                        ;UDPDR1=        177622
                        ;UDPDR2=        177624
                        ;UDPDR3=        177626
```

```
                                ;UDPDR4=        177630
                                ;UDPDR5=        177632
                                ;UDPDR6=        177634
                                ;UDPDR7=        177636

                                        ;*USER ''I'' PAGE ADDRESS REGISTERS
        177640                  FASTCITY=UIPAR0
        177640                  UIPAR0= 177640          ;PATTERN PROGRAM SPACE
        177642                  UIPAR1= 177642          ;PATTERN PROGRAM SPACE
        177644                  UIPAR2= 177644          ;PATTERN PROGRAM SPACE
        177646                  UIPAR3= 177646          ;PATTERN PROGRAM SPACE
        177650                  UIPAR4= 177650          ;PATTERN PROGRAM SPACE
        177652                  UIPAR5= 177652          ;PATTERN PROGRAM SPACE
        177654                  UIPAR6= 177654          ;PATTERN PROGRAM SPACE
                                ;UIPAR7=        177656          ;PATTERN PROGRAM SPACE

                                        ;*USER 'D'' PAGE ADDRESS REGISTERS
        177660                  UDPAR0= 177660          ;PATTERN PROGRAM SPACE
                                ;UDPAR1=        177662          ;PATTERN PROGRAM SPACE
                                ;UDPAR2=        177664          ;PATTERN PROGRAM SPACE
                                ;UDPAR3=        177666          ;PATTERN PROGRAM SPACE
                                ;UDPAR4=        177670          ;PATTERN PROGRAM SPACE
                                ;UDPAR5=        177672          ;PATTERN PROGRAM SPACE
                                ;UDPAR6=        177674          ;PATTERN PROGRAM SPACE
        177676                  UDPAR7= 177676          ;PATTERN PROGRAM SPACE

                                        ;*SUPERVISOR ''I'' PAGE DESCRIPTOR REGISTERS
        172200                  SIPDR0= 172200
                                ;SIPDR1=        172202
                                ;SIPDR2=        172204
                                ;SIPDR3=        172206
                                ;SIPDR4=        172210
                                ;SIPDR5=        172212
                                ;SIPDR6=        172214
                                ;SIPDR7=        172216

                                        ;*SUPERVISOR 'D'' PAGE DESCRIPTOR REGISTERS
                                ;SDPDR0=        172220
                                ;SDPDR1=        172222
                                ;SDPDR2=        172224
                                ;SDPDR3=        172226
                                ;SDPDR4=        172230
                                ;SDPDR5=        172232
                                ;SDPDR6=        172234
                                ;SDPDR7=        172236

                                        ;*SUPERVISOR ''I'' PAGE ADDRESS REGISTERS
        172240                  SIPAR0= 172240
                                ;SIPAR1=        172242
                                ;SIPAR2=        172244
        172246                  SIPAR3= 172246          ;TEST AREA
                                ;SIPAR4=        172250          ;TEST AREA
        172252                  SIPAR5= 172252          ;TEST AREA
        172254                  SIPAR6= 172254          ;TEST AREA
                                ;SIPAR7=        172256

                                        ;*SUPERVISOR 'D'' PAGE ADDRESS REGISTERS
```

```
                        SDPAR0= 172260
                        ;SDPAR1=        172262
                        ;SDPAR2=        172264
                        ;SDPAR3=        172266
                        ;SDPAR4=        172270
                        SDPAR5= 172272
                        SDPAR6= 172274
                        SDPAR7= 172276

                                ;*KERNEL ''I'' PAGE DESCRIPTOR REGISTERS
                        KIPDR0= 172300
                        ;KIPDR1=        172302
                        ;KIPDR2=        172304
                        ;KIPDR3=        172306
                        ;KIPDR4=        172310
                        ;KIPDR5=        172312
                        ;KIPDR6=        172314
                        ;KIPDR7=        172316

                                ;*KERNEL 'D'' PAGE DESCRIPTOR REGISTERS
                        ;KDPDR0=        172320
                        ;KDPDR1=        172322
                        ;KDPDR2=        172324
                        ;KDPDR3=        172326
                        ;KDPDR4=        172330
                        ;KDPDR5=        172332
                        ;KDPDR6=        172334
                        ;KDPDR7=        172336

                                ;*KERNEL ''I'' PAGE ADDRESS REGISTERS
                        KIPAR0= 172340
                        ;KIPAR1=        172342
                        ;KIPAR2=        172344
                        ;KIPAR3=        172346
                        KIPAR4= 172350
                        KIPAR5= 172352
                        KIPAR6= 172354
                        ;KIPAR7=        172356

                                ;*KERNEL 'D'' PAGE ADDRESS REGISTERS
                        KDPAR0= 172360
                        ;KDPAR1=        172362
                        ;KDPAR2=        172364
                        ;KDPAR3=        172366
                        ;KDPAR4=        172370
                        ;KDPAR5=        172372
                        KDPAR6= 172374
                        KDPAR7= 172376
```

```
                                .SBTTL  DEFINE  UNIBUS MAP REGISTERS
                                ;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
                                ;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'
170200                  MAPL0 = 170200
170202                  MAPH0 = 170202
170204                  MAPL1 = 170204
                        ;MAPH1 = 170206
                        ;MAPL2 = 170210
                        ;MAPH2 = 170212
                        ;MAPL3 = 170214
                        ;MAPH3 = 170216
                        ;MAPL4 = 170220
                        ;MAPH4 = 170222
                        ;MAPL5 = 170224
                        ;MAPH5 = 170226
                        ;MAPL6 = 170230
                        ;MAPH6 = 170232
                        ;MAPL7 = 170234
                        ;MAPH7 = 170236
                        ;MAPL10 = 170240
                        ;MAPH10 = 170242
                        ;MAPL11 = 170244
                        ;MAPH11 = 170246
                        ;MAPL12 = 170250
                        ;MAPH12 = 170252
                        ;MAPL13 = 170254
                        ;MAPH13 = 170256
                        ;MAPL14 = 170260
                        ;MAPH14 = 170262
                        ;MAPL15 = 170264
                        ;MAPH15 = 170266
                        ;MAPL16 = 170270
                        ;MAPH16 = 170272
                        ;MAPL17 = 170274
                        ;MAPH17 = 170276
                        ;MAPL20 = 170300
                        ;MAPH20 = 170302
                        ;MAPL21 = 170304
                        ;MAPH21 = 170306
                        ;MAPL22 = 170310
                        ;MAPH22 = 170312
                        ;MAPL23 = 170314
                        ;MAPH23 = 170316
                        ;MAPL24 = 170320
                        ;MAPH24 = 170320
                        ;MAPL25 = 170324
                        ;MAPH25 = 170326
                        ;MAPL26 = 170330
                        ;MAPH26 = 170332
                        ;MAPL27 = 170334
                        ;MAPH27 = 170336
                        ;MAPL30 = 170340
                        ;MAPH30 = 170342
                        ;MAPL31 = 170344
                        ;MAPH31 = 170346
                        ;MAPL32 = 170350
                        ;MAPH32 = 170352
```

```
                                ;MAPL33 = 170354
                                ;MAPH33 = 170356
                                ;MAPL34 = 170360
                                ;MAPH34 = 170362
                                ;MAPL35 = 170364
                                ;MAPH35 = 170366
                                ;MAPL36 = 170370
                                ;MAPH36 = 170372
                                ;MAPL37 = 170374
                                ;MAPH37 = 170376

                                        .SBTTL  DEFINE  SOFTWARE SWITCH & DISPLAY REGISTERS
000174                          DISPREG=174
000176                          SWREG=  176

                                        .SBTTL  DEFINE  CONTROL STATUS REGISTERS
172100                          CSRADD=172100

                                        .SBTTL  DEFINE  PARAMETERS
060000                          FIRST=60000             ;START OF THE 16K TEST PATTERN AREA
157776                          LAST=157776             ;END OF THE 16K TEST PATTERN AREA
040000                          SIZE=40000              ;SIZE OF THE 16K TEST PATTERN AREA (FOR SOB INSTRUCTIONS)
```

```
                        .LIST   MD                      ;BE NICE TO SEE MY DEFINITIONS
                        .SBTTL  MACRO   FATAL
                ;********************** FATAL ****************************
                ;
                ;FATAL IS USED TO REPORT FATAL ERRORS (ERRORS THAT PREVENT
                ;       THE PROGRAM FROM CONTINUING).
                ;
                ;*********************************************************
                        .MACRO  FATAL   ARG             ;***MACRO***MACRO***MACRO***
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        INC     FATAL$                  ;SET FATAL INDICATOR
                        ERROR   +ARG
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   FATAL

                        .SBTTL  MACRO   TYPE
                        .MACRO  TYPE    ARG
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        .IF B ARG
                        TYPEIT
                        .IFF
                        TYPEIT  ,ARG
                        .ENDC
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   TYPE
```

```
                        .SBTTL  MACRO   NEWTST
                ;********************** NEWTST **********************
                ;NEWTST IS USED AS THE FIRST INSTRUCTION OF A TEST.
                ;IT WILL:
                ;1)  GENERATE A TEST NUMBER FOR THE LABEL OF THIS TEST
                ;2)  PUT STARS BEFORE AND AFTER A MESSAGE
                ;ARGUMENTS
                ;1)  ASCII  --  THIS IS THE MESSAGE THAT WILL APPEAR
                ;               ON THE LISTING
                ;2)  ICOUNT --  IF NON-BLANK AND BIT 11 OF $SWR = 1 IT WILL BE
                ;               THE NUMBER Of ITERATIONS TO MAKE ON THIS TEST
                ;3)  RETURN --  IF NON-BLANK WILL BE THE ADDRESS TO
                ;               WHICH THE NEXT SCOPE STATEMENT WILL
                ;               LOOP BACK TO.
                ;4)  COMAND --  IF NON-BLANK WILL BE THE FIRST
                ;               INSTRUCTION OF THE TEST
                ;               IF BLANK SCOPE WILL BE THE
                ;               FIRST INSTRUCTION
                ;*****************************************************
                        .MACRO  NEWTST  ASCII,ICOUNT,RETURN,COMAND
                        $STN=1
                        $NWTST=0
                        .NLIST  MC
                        .IF B <COMAND>
                        $$NEWTEST       \$TN,<ASCII>,SCOPE
                        .IFF
                        $$NEWTEST       \$TN,<ASCII>,<COMAND>
                        .ENDC
                        .NLIST
                        .LIST   ME
                        .LIST
                        .IF NE 4000&$SWR
                        .IF NB ICOUNT
                        .IF LE <ICOUNT-1>
                        MOV     #1,$TIMES       ;;DO 1 ITERATION
                        .IFF
                        MOV     #ICOUNT,$TIMES  ;;DO ICOUNT ITERATIONS
                        .ENDC
                        .ENDC
                        .IF NB RETURN
                        MOV     #RETURN,$LPADR  ;;SET SCOPE LOOP ADDRESS
                        .ENDC
                        .ENDC
                        .NLIST
                        .LIST   MC
                        .LIST
                        .NLIST  ME
                        .ENDM   NEWTST
```

```
                        .SBTTL  MACRO   $$NEWTEST
                        .MACRO  $$NEWTEST       A,ASC,COMND
                        .IRP    ASCI,<ASC>
                        .IF EQ $NWTST
                        $NWTST=1
                        .SBTTL  T'A'    ASCI
                        .NLIST
                        .LIST   ME
                        .LIST
                ;*******************************************************************************
                ;*TEST A         ASCI
                        .IFF
                        ASCI
                        .ENDC
                        .ENDM
                ;*******************************************************************************
                TST'A:  COMND
                        .NLIST  ME
                        $TN=$TN+1
                        .ENDM   $$NEWTEST

                        .SBTTL  MACRO   SUBTST
                ;************** SUBTST *********************************
                ;
                ;THIS MACRO WILL FORMAT A SUBTEST HEADING WITH STARS
                ;A .SBTTL WILL BE FORCED & .NLISTED FOR THE TABLE OF CONTENTS.
                ;
                ;ARGUMENT:
                ;1) TXT --      THIS IS THE MESSAGE THAT WILL APPEAR IN THE TABLE OF CONTENTS & LISTING.
                ;
                ;EXAMPLE:       SUBTST  <<THIS IS A FUN SUBTST>>
                ;
                ;*************************************************************

                        .MACRO  SUBTST  ASCII
                        .NLIST  MC
                        $SUBTST <ASCII>
                        .LIST   MC
                        .ENDM   SUBTST

                        .SBTTL  MACRO   $SUBTST
                        .MACRO  $SUBTST ASC
                        .IRP    ASCI,<ASC>
                        .SBTTL  ASCI
                        .NLIST
                        .LIST   ME
                        .LIST
                ;*******************************************************************************
                ;*SUBTEST        ASCI
                        .ENDM
                ;*******************************************************************************
                        .NLIST  ME
                        .ENDM   $SUBTST
```

```
                        .SBTTL  MACRO   TYPOCT
                ;********************* TYPOCT *********************
                ;
                ;TYPOCT IS USED TO CHANGE A BINARY NUMBER
                ;       TO A 6 DIGIT OCTAL NUMBER AND TYPE IT
                ;
                ;ARGUMENTS:
                ;
                ;1)     NUM     THE NUMBER TO BE TYPED
                ;
                ;2)     REMARK  ALLOWS A COMMENT TO BE MADE
                ;
                ;ROUTINES REQUIRED
                ;
                ;1)     CONVERT BINARY TO OCTAL AND TYPE (.$TYPOCT)
                ;
                ;2)     TYPE AN ASCIZ STRING (.$TYPE)
                ;
                ;EXAMPLES:
                ;
                ;1)     TYPOCT  HILMT,<TYPES THE CONTENTS OF HILMT>
                ;2)     TYPOCT  #5,<TYPES ' 000005'>
                ;
                ;*****************************************************************
                ;
                        .MACRO  TYPOCT  NUM,REMARK
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LS1$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        MOV     NUM,-(SP)       ;;SAVE NUM FOR TYPEOUT
                        .IIF NB <REMARK>,                       ;;REMARK
                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   TYPOCT
```

```
                        .SBTTL  MACRO   TYPOCS
                ;********************* TYPOCS *********************
                ;
                ;TYPOCS IS USED TO CHANGE A BINARY NUMBER TO AN OCTAL
                ;       NUMBER AND TYPE 1 TO 6 DIGITS
                ;       WITH OR WITHOUT LEADING ZEROS.
                ;
                ;ARGUMENTS:
                ;
                ;1)     NUM     NUMBER TO BE TYPED
                ;
                ;2)     REMARK  ALLOWS A COMMENT TO BE MADE
                ;
                ;3)     N       NUMBER OF DIGITS (1 TO 6) TO BE TYPED
                ;
                ;4)     Z       BLANK=SUPPRESS LEADING ZEROS (TYPES SPACES)
                ;               NON-BLANK=TYPE LEADING ZEROS
                ;
                ;ROUTINES REQUIRED
                ;
                ;1)     CONVERT BINARY TO OCTAL AND TYPE (.$TYPOCT)
                ;
                ;2)     TYPE AN ASCIZ STRING (.$TYPE)
                ;
                ;EXAMPLES:
                ;1)     TYPOCS  #12345,<TYPES '5'>,1
                ;2)     TYPOCS  #004,<TYPES '04'>,2,X
                ;3)     TYPOCS  #004,<TYPES ' 4'>,2
                ;
                ;*****************************************************************

                .MACRO  TYPOCS  NUM,REMARK,N,Z
                .NLIST
                .DSABL  CRF
                .IIF DF LST$$ .LIST  ME
                .ENABL  CRF
                .LIST
                MOV     NUM,-(SP)       ;;SAVE NUM FOR TYPEOUT
                .IIF NB <REMARK>,                               ;;REMARK
                TYPOS                   ;;GO TYPE--OCTAL ASCII
                .IF NB N
                .BYTE   N               ;;TYPE N DIGIT(S)
                .IFF
                .BYTE   6               ;;TYPE 6 DIGITS
                .ENDC
                .IF NB Z
                .BYTE   1               ;;TYPE LEADING ZEROS
                .IFF
                .BYTE   0               ;;SUPPRESS LEADING ZEROS
                .ENDC
                .DSABL  CRF
                .IIF DF LST$$ .NLIST  ME
                .ENABL  CRF
                .ENDM   TYPOCS
```

```
                        .SBTTL  MACRO   TYPDEC
                ;********************* TYPDEC *********************
                ;
                ;TYPDEC IS USE TO CHANGE A BINARY NUMBER TO A SIGNED
                ;       DECIMAL NUMBER AND TYPE IT REPLACING LEADING ZERO
                ;       WITH SPACES.
                ;NOTE:  IF THE NUMBER IS NEGATIVE A
                ;       MINUS SIGN WILL BE TYPED.
                ;
                ;ARGUMENTS:
                ;
                ;1)     NUM     NUMBER TO BE TYPED
                ;
                ;2)     REMARK ALLOWS A COMMENT TO BE MADE
                ;
                ;ROUTINES REQUIRED
                ;
                ;1)     CONVERT BINARY TO DECIMAL AND TYPE (.$TYPDEC)
                ;
                ;2)     TYPE AN ASCIZ STRING (.$TYPE)
                ;
                ;EXAMPLES
                ;
                ;1)     TYPDEC  SIZE,<TYPE THE CONTENTS OF SIZE>
                ;2)     TYPDEC  #-10.,<TYPE A MINUS TEN>
                ;
                ;*****************************************************************

                        .MACRO  TYPDEC  NUM,REMARK
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        MOV     NUM,-(SP)       ;;SAVE NUM FOR TYPEOUT
                        .IIF NB,<REMARK>,                       ;;REMARK
                        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   TYPDEC
```

```
                        .SBTTL  MACRO   BMOV
                ;****************** BMOV ****************************
                ;
                ; THIS MACRO MOVES A BLOCK OF DATA.
                ;
                ;ARGUEMENTS:
                ;
                ;1)     FROMHERE        THE FIRST ADDRESS OF THE SOURCE BLOCK.
                ;
                ;2)     TOHERE          THE FIRST ADDRESS OF THE DESTINATION BLOCK.
                ;                       IF BLANK THE 1ST ADDRESS OF THE USER INSTRUCTION
                ;                       PAR'S IS USED (FASTCITY).
                ;
                ;3)     SIZE            THE SIZE OF THE SOURCE BLOCK.
                ;                       IF BLANK A 16 WORD TRANSFER IS ASSUMED.
                ;                       'WHY DEFAULT TO 16 WORDS?'' YOU ASK!
                ;                       'BECAUSE THAT'S HOW MANY WORDS TO THE USER PAR
                ;                       REGISTERS & THAT'S WHERE I INTEND TO MOVE LOTS
                ;                       OF STUFF.'' I REPLY!
                ;
                ;*****************************************************

                        .MACRO  BMOV    FROMHERE,TOHERE,SIZE
                          .IF B TOHERE
                            .NLIST
                            .DSABL      CRF
                            .IIF DF LST$$ .LIST  ME
                            .ENABL      CRF
                            .LIST
                            JSR R5,BLOCK1
                            FROMHERE
                            .DSABL      CRF
                            .IIF DF LST$$ .NLIST  ME
                            .ENABL      CRF
                            .MEXIT
                        .ENDC
                        .IF B SIZE
                            .NLIST
                            .DSABL      CRF
                            .IIF DF LST$$ .LIST  ME
                            .ENABL      CRF
                            .LIST
                            JSR R5,BLOCK2
                            TOHERE
                            FROMHERE
                            .DSABL      CRF
                            .IIF DF LST$$ .NLIST  ME
                            .ENABL      CRF
                            .MEXIT
                        .IFF
                            .NLIST
                            .DSABL      CRF
                            .IIF DF LST$$ .LIST  ME
                            .ENABL      CRF
                            .LIST
                            JSR R5,BLOCK3
                            SIZE
```

```
    855                                         TOHERE
    856                                         FROMHERE
    857                                         .DSABL      CRF
    858                                         .IIF DF LST$$ .NLIST  ME
    859                                         .ENABL      CRF
    860                                       .ENDC
    861                                     .ENDM   BMOV
```

```
                        .SBTTL  MACRO   MAP
                ;***************** MAP ****************************
                ;
                ; THIS MACRO MAPS A MEMORY BANK (16K) INTO THE
                ; TEST PATTERN AREA (SUPERVISOR VIRTUAL (60000-157777)).
                ;
                ;ARGUEMENTS:
                ;
                ;1)     BANK    THE BANK OF 16K WORDS TO BE MAPPED.
                ;               THERE ARE 120 BANKS OF 16K WORDS
                ;
                ;EXAMPLES
                ;
                ;       MAP     LOC             ;LOCATION 'LOC'' CONTAINS THE # OF THE BANK TO MAP
                ;
                ;       MAP     #28.            ;BANK 34 (OCTAL) WILL BE MAPPED
                ;
                ;***************************************************

                .MACRO  MAP     BANK
                PUSH    R3
                .NLIST
                .DSABL  CRF
                .IIF DF LST$$ .LIST  ME
                .ENABL  CRF
                .LIST
                .IF B BANK
                MOV     #120.,R3
                .IFF
                MOV     BANK,R3
                .ENDC
                CALL    MAPPER
                .DSABL  CRF
                .IIF DF LST$$ .NLIST  ME
                .ENABL  CRF
                POP     R3
                .ENDM   MAP
```

```
                        .SBTTL  MACRO   SUPERVISOR
                ;***************** SUPERVISOR **********************
                ;
                ; THIS MACRO SWITCHES TO SUPERVISOR MODE.
                ;
                ;ARGUEMENTS: NONE.
                ;
                ;*****************************************************

                        .MACRO  SUPERVISOR
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        BIS     #BIT14,PSW                      ;GO TO SUPERVISOR MODE
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   SUPERVISOR

                        .SBTTL  MACRO   USER
                ;******************* USER ***************************
                ;
                ; THIS MACRO SWITCHES TO USER MODE.
                ;
                ;ARGUEMENTS: NONE.
                ;
                ;*****************************************************

                        .MACRO  USER
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        BIS     #BIT15!BIT14,PSW                ;GO TO USER MODE
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   USER
```

```
                        .SBTTL  MACRO   TESTAREA
                ;***************** TESTAREA ************************
                ;
                ; THIS MACRO SWITCHES TO THE SPECIFIED TEST MODE.
                ;
                ;ARGUEMENTS: NONE.
                ;
                ;*****************************************************

                        .MACRO  TESTAREA
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        BIS     TESTMODE,PSW                    ;GO TO SYSTEM TEST MODE
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   TESTAREA
```

```
                        .SBTTL  MACRO   SET4 & RES4
                ;****************** SET4 & RES4 *********************
                ;
                ; THESE MACROS SET & RESTORE VECTOR 4(TIMEOUT TRAP)
                ;
                ; IN IT'S RESTORED MODE TRAPS ARE REPORTED AS SUCH.
                ;
                ;ARGUEMENTS:    LOC     ;THE LOCATION TO VECTOR TO (ONLY USED IN 'SET4' NOT 'RES4')
                ;
                ;I USE THE SET4 AND RES4 MACROS AROUND CODE THAT I EXPECT TO TRAP TO 4
                ;LIKE LOOKING FOR ALL POSSIBLE CSR'S AND ETC.  WHENEVER CODE IS NOT
                ;SURROUNDED BY SET4 AND RES4 THEN ANY TRAPS TO 4 WILL CAUSE AN ERROR
                ;PRINTOUT THAT SAYS 'UNEXPECTED TRAP TO 4' AND ALL THE ASSOCIATED REGISTER JUNK
                ;*****************************************************

                        .MACRO  SET4    ARG
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        MOV     ARG,4
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   SET4

                        .MACRO  RES4
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        MOV     #TIMEOUT,4
                        TST     NO22BIT         ;IS THIS AN 11/44?
                        BNE     101$            ;BRANCH IF NOT
                        CLR     CPUERR          ;CLEAR OUT THE CPU ERROR REGISTER BITS
                101$:
                                                ;THAT A EXPECTED TRAP COULD HAVE SET
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   RES4
```

```
                        .SBTTL  MACRO   DLEFT
                ;****************** DLEFT ****************************
                ;
                ; THIS MACRO DOES A DOUBLE WORD LEFT SHIFT
                ;
                ;ARGUEMENTS:    LOC     ;THE LOCATION TO BE SHIFTED LEFT (CARRY TO LOC+2)
                ;
                ;*****************************************************

                        .MACRO  DLEFT   ARG
                        .NLIST
                        .DSABL  CRF
                        .IIF DF LST$$ .LIST  ME
                        .ENABL  CRF
                        .LIST
                        ROL     ARG
                        ROL     ARG+2
                        .DSABL  CRF
                        .IIF DF LST$$ .NLIST  ME
                        .ENABL  CRF
                        .ENDM   DLEFT
                        .NLIST  MD              ;DON'T NEED TO SEE THEM ANY MORE
```

```
1037                                    .SBTTL  TRAP CATCHER
1038            000000                  .=0
1039 000000     000000  000000          .WORD   0,0
1040            000177                  .REPT   177             ;.WORD  .+2,HALT
1044
1045                                    .SBTTL  ACT11 HOOKS
1046                            ;*THE HOOKS REQUIRED BY ACT11 ARE DEFINED AND SETUP BELOW:
1047                            ;*      DEFINITIONS:
1048                            ;*              1)LOC.46        'END-OF-PASS' HOOK
1049                            ;*                      =ADDRESS OF END OF PASS ROUTINE
1050                            ;*                       MODIFIED BY ACT11.
1051                            ;*              2)LOC.52        PROGRAM NEEDS HOOK
1052                            ;*                      BIT 15=1 PROGRAM SHOULD BE POWER
1053                            ;*                               FAILED WHILE RUNNING
1054                            ;*                        =0 NO POWER FAIL
1055                            ;*                      BIT 14=1 PROGRAM MEMORY SIZE DEPENDENT
1056                            ;*                        =0 NOT MEMORY SIZE DEPENDENT
1057                            ;*                      BIT 13=1 PROGRAM REQUIRES MANUAL INTERVENTION
1058                            ;*                        =0  MANUAL INTERVENTION NOT REQUIRED
1059                            ;*                      BITS 12-0  MUST BE ZERO'S
1060            000046                  .=46
1061 000046     013326                  $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
1062            000052                  .=52
1063 000052     000020                  .WORD   BIT4            ;;2)SET LOC.52 TO INDICATE MEMORY SIZE DEPENDANT
1064                                    .SBTTL  APT11 HOOKS
1065            000024                  .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
1066 000024     000200                  200     ;;FOR APT START UP
1067            000042                  .=42
1068 000042     002000                  STACK                   ;SO RT11 CAN START WITH RUN COMMAND
1069            000044                  .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
1070 000044     061356                  $APTHDR ;;POINT TO APT HEADER BLOCK
1071            000200                  .=200
1072 000200     000437          START3: BR      START1          ;'NORMAL' START
1073 000202     000442                  BR      START2          ;RESTART (SAVE ERROR ACCOUNTING)
1077            000300                  .=300
1078 000300     005037  002566  START1: CLR     RESTART
1079 000304     000137  003630          JMP     START
1080 000310                     START2: SET     RESTART
1081 000316     000137  003630          JMP     START
1086            002000                  .=STACK
```

```
                                        .SBTTL  VARIABLES       INITIALIZED TO ZERO
                                ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                ;*USED IN THE PROGRAM.
002000                          $CMTAG:                         ;;START OF COMMON TAGS
002000  000000                  SELONLY:0                       ;SELECT ONLY BANKS MARKED BY FIELD SERVICE MODE FLAG
002002  000000                  DIAGFLAG:0                      ;SET FOR SHIFTING DIAGONAL TEST
002004  000000                  KAMIKAZE:0                      ;SET FOR KAMIKAZE MODE TESTING
002006  000000                  SKIPKAMI:0                      ;USED TO SKIP RESTORING KAMIKAZE MODE WHEN MODIFIED
                                ;NEXT TWO BYTES ARE DISPLAYED IN THE DISPLAY REGISTER
002010      000                 $PATMAR:.BYTE   0               ;PATTERN NUMBER
002011      000                 $BANK:  .BYTE   0               ;BANK & SIGN
002012      000                 $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
002013      000                 $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
002014  000000                  LASTERROR:.WORD 0               ;NUMBER OF ERRORS ON LAST PASS
002016  000000                  ERRPC:  .WORD   0               ;CONTAINS PC OF ERROR FOR TYPEOUT
002020  000000                  BADPC:  .WORD   0               ;CONTAINS PC OF ERROR
002022  000000                  ERRSP:  .WORD   0               ;CONTAINS SP OF ERROR FOR TYPEOUT
002024  000000                  BADSP:  .WORD   0               ;CONTAINS SP OF ERROR
002026  000000                  ERRPSW: .WORD   0               ;CONTAINS PSW OF ERROR FOR TYPEOUT
002030  000000                  BADPSW: .WORD   0               ;CONTAINS PSW OF ERROR
002032  000000                  ADDRESS:.WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
002034  000000                  PADDRESS:.WORD  0               ;ADDRESS OF PARITY ERROR
002036  000000  000000          PHYADD: .WORD   0,0             ;22 BIT PHYSICAL ADDRESS
002042  000000                  GOOD:   .WORD   0               ;;CONTAINS 'GOOD' DATA
002044  000000                  GOOD2:  .WORD   0               ;;CONTAINS 'GOOD2' DATA
002046  000000                  GOOD3:  .WORD   0               ;;CONTAINS 'GOOD3' DATA
002050  000000                  BAD:    .WORD   0               ;;CONTAINS 'BAD' DATA
002052  000000                  BAD2:   .WORD   0               ;;CONTAINS 'BAD2' DATA
002054  000000                  BAD3:   .WORD   0               ;;CONTAINS 'BAD3' DATA
002056  000000                  BADXOR: .WORD   0               ;XOR OF GOOD & BAD = BAD BITS!
002060  000000                  $AUTO:  .WORD   0               ;;AUTOMATIC MODE INDICATOR FOR APT,ACT, & XXDP
002062  000000                  FATAL$: .WORD   0               ;FATAL ERROR INDICATOR
002064  000000                  SKPERR: .WORD   0               ;USED TO SKIP ERROR MESSAGE IN ''$ERRGEN''
002066  000000                  NEMCNT: 0                       ;NON-EXISTANT MEMORY COUNTER (HOLES)
002070  000000                  PARCNT: 0                       ;PARITY ERROR COUNTER
002072  000000                  PATERR: 0                       ;PATTERN ERROR COUNTER
002074  000000                  NOPAR:  0                       ;NO PARITY ERROR MODE INDICATOR
002076  000000                  NONEM:  0                       ;NO NON-EXISTANT MEMORY (HOLES) MODE INDICATOR
002100  000000                  BANK:   0                       ;MEMORY BANK UNDER TEST
002102  000000                  BANKINDEX:0                     ;USED TO INDEX INTO CONFIG TABLE
002104  000000                  CPUBIT: 0                       ;CONTAINS 1 BIT TO IDENTIFY CPU TO CONFIGURATION TABLE
002106  000000                  MUT:    0                       ;MEMORY UNDER TEST FLAG
002110  000000                  PATTERN:0                       ;PATTERN NUMBER UNDER TEST
002112  000000                  KPFLAG: .WORD   0               ;BANK IS PROTECTED REGION OF ECC
002114  000000                  ACFLAG: .WORD   0               ;BANK CAN BE ACCESSED BY THIS CPU
002116  000000                  MKFLAG: .WORD   0               ;IF SET INDICATES MS11-M OR MF11S-K UNDER TEST
002120  000000                  PFLAG:  .WORD   0               ;BANK IS IN PROGRAM SPACE
002122  000000                  RRFLAG: .WORD   0               ;BANK IS WHERE PROGRAM RELOCATION IS REQUIRED TO TEST
002124  000000                  RLFLAG: .WORD   0               ;PROGRAM IS RELOCATED FLAG
002126  000000                  BMFLAG: .WORD   0               ;''BANK IS IDENTIFIED AS BAD MEMORY'' FLAG
002130  000000                  EUFLAG: .WORD   0               ;''BANK HAS EUB MEMORY'' FLAG
002132  000000                  TMFLAG: .WORD   0               ;''TYPE OF MEMORY TO TEST'' FLAG; 0 = PARITY, 1 = ECC
002134  000000                  INTFLAG:.WORD   0               ;''BANK IS INTERLEAVED'' FLAG
002136  000000                  INT64K: .WORD   0               ;''BANK IS 64K INTERLEAVED'' FLAG
002140  000000                  ABORTFLAG:.WORD 0               ;''ABORT OCCURED'' FLAG
002142  000000                  CTLKVEC:.WORD   0               ;HOLDS OLD KERNAL STACK POINTER IN CASE OF CNTL/K
002144  000000                  CSR:    .WORD   0               ;DATA TO OR FROM CSR
```

```
1146 002146  000000                  CSRNO:  0                       ;CSR ADDRESS NUMBER (4 LSB'S)
1147 002150  000000                  OLDCSR: .WORD   0               ;OLD CSR NUMBER(USED IN INH PTR TEST)
1148                                         ;THESE LOCATIONS STORE GPR'S DURING SUPERVISOR TESTS
1149 002152  000000                  SUPDR0: 0
1150 002154  000000                  SUPDR1: 0
1151 002156  000000                  SUPDR2: 0
1152 002160  000000                  SUPDR3: 0
1153 002162  000000                  SUPDR4: 0
1154 002164  000000                  SUPDR5: 0
1155 002166  000000                  SUPDR6: 0
1156 002170  000000                  DUMMY:  0                       ;DUMMY LOCATION FOR ADDRESS PASSING
1157                                 ;THESE LOCATIONS STORE GPR'S & PSW DURING DETAILED ERROR PRINTOUTS
1158 002172  000000                  DETR0:  0
1159 002174  000000                  DETR1:  0
1160 002176  000000                  DETR2:  0
1161 002200  000000                  DETR3:  0
1162 002202  000000                  DETR4:  0
1163 002204  000000                  DETR5:  0
1164 002206  000000                  DETSP:  0
1165 002210  000000                  DETPSW: 0
1166 002212  000000                  DETFLAG:0                       ;DETAILED REPORT FLAG
1167 002214  000000                  CONTFLAG:0                      ;CSR'S HAVE BEEN TESTED FLAG
1168 002216  000000                  TOTCSRS:.WORD   0               ;1 BIT PER EXISTING CSR, EG-
1169                                                                 ;CSR 0 REPRESENTED BY BIT 15, ETC.
1170 002220  000000                  CSRFIRST:.WORD  0               ;FIRST ADDRESS UNDER CONTROL OF THIS CSR
1171 002222  000000                  CSRLAST: .WORD  0               ;
1172 002224  000000                  CSRFBANK:.WORD  0               ;
1173 002226  000000                  CSRLBANK:.WORD  0               ;
1174 002230  000000                  CSRINT:  .WORD  0               ;
1175 002232  000000                  SPLTCSR: .WORD  0               ;
1176 002234  000000  000000          DATBUF: .WORD   0,0             ;TWO WORD DATA BUFFER
1177 002240  000000  000000          TSTDAT: .WORD   0,0             ;TWO WORD TEST DATA
1178 002244  000000  000000          SBEMSK: .WORD   0,0             ;TWO WORD SINGLE BIT ERROR MASK
1179 002250  000000  000000          DBEMSK: .WORD   0,0             ;TWO WORD DOUBLE BIT ERROR MASK
1180 002254  000000                  SUPDOADD:.WORD  0               ;ADDRESS OF SUBROUTINE TO EXECUTE IN SUPERVISOR MODE
1181 002256     000                  PASFLG: .BYTE   0               ;LOCAL LOOP PASS CONTROL
1182 002257     000                  UPPFLG: .BYTE   0               ;LOCAL LOOP PASS CONTROL
1183 002260  000000                  REALPAT:.WORD   0               ;REAL PATTERN UNDER TEST
1184 002262  000000                  OLDCACHE:.WORD  0               ;BACKED UP VALUE OF CACHE CONTROL REGISTER
1185 002264  000000                  PARTHERE:.WORD  0               ;PARITY TRAPS SOMETIMES GO TO ADDRESS STORED HERE
1186 002266  000000                  FSSTACK:.WORD   0               ;STACK SAVED HERE IF IN FIELD SERVICE MODE
1187 002270  000000                  NEWBANK:.WORD   0               ;USED FOR RELOCATION TO A NEW BANK
1188 002272  000000                  SOURCE: .WORD   0               ;SOURCE OF DATA WORDS FOR CHECKBIT GENERATION SUBROUTINE
1189 002274  000000                  CHECK:  .WORD   0               ;CHECKBITS TO BE LOADED INTO CSR
1190 002276  000000                  PCBUMP: .WORD   0               ;VALUE TO BUMP THE PC BY TO RECOVER AFTER A PARITY TRAP
1191 002300  000000                  CSRINC: .WORD   0               ;VALUE TO INCREMENT ADDRESS BY TO REMAIN IN THE SAME CSR
1192 002302  000000                  CSRLOOP:.WORD   0               ;LOOP CONTROL FOR CSR TESTING
1193 002304  000000                  SUCCESS:.WORD   0               ;FLAG SET BY SUCCESSFULL TASK OR SUBROUTINE
1194 002306  000000                  ZEROS:  .WORD   0               ;FOR AID IN 'MOV'' INSTRUCTIONS
1195 002310  000000                  TIME:   .WORD   0               ;SECONDS THAT BATTERIES SHOULD LAST
1196 002312  000000                  SKIPMK: .WORD   0               ;FLAG TO SKIP MKCONTROL SUBROUTINE
1197 002314  000000                  NULLFLAG:.WORD  0               ;SET WHEN RUNNING NULL PATTERNS
1198 002316  000000                  QVFLAG: 0                       ;FLAGS QUICK VERIFY PASS UNDER APT, ACT, OR XXDP CHAIN MODE
1199 002320  000000                  ACTFLAG:0                       ;FLAGS ACT AUTOMATIC MODE PROGRAMMING RULES
1200 002322  000000                  APTFLAG:0                       ;FLAGS APT AUTOMATIC MODE PROGRAMMING RULES
1201 002324  000000                  XXDPCHAIN:0                     ;FLAGS XXDP CHAIN MODE PROGRAMMING RULES
1202                                 ;NOTE: THESE TWO BYTES MUST STAY TOGETHER
```

```
1203 002326  000                  $NULL:  .BYTE   0              ;;CONTAINS NULL CHARACTER FOR FILLS
1204 002327  000                  $FILLS: .BYTE   0              ;CONTAINS # OF FILL CHARACTERS
1205 002330  000                  $TPFLG: .BYTE   0              ;;'TERMINAL NOT AVAILABLE'' FLAG
1206                                       .EVEN
1207 002332  000000               $ESCAPE:0                      ;;ESCAPE ON ERROR ADDRESS
1208 002334  000000               EVEN:   0                      ;USED FOR ALTERNATE DATA PATTERNS
1209 002336  000000               STRIPES:0                      ;COUNTS DIAGONAL STRIPES
1210 002340  000000               COUNT:  0                      ;BACKED UP COPY OF STRIPES
1211 002342  000000               NOTAB:  0                      ;NO TABLE BEING PRINTED - NOW
1212 002344  000000               BSIZE:  0                      ;SIZE OF 11/45 MOS MEMORY IN K WORDS
1213 002346  000000               KSIZE:  0                      ;SIZE OF MF11S-K MEMORY IN K WORDS
1214 002350  000000               LSIZE:  0                      ;SIZE OF MS11-L MEMORY IN K WORDS
1215 002352  000000               MSIZE:  0                      ;SIZE OF MS11-M MEMORY IN K WORDS
1216 002354  000000               PSIZE:  0                      ;SIZE OF UNIBUS PARITY MEMORY IN K WORDS
1217 002356  000000               TOOMANY:0                      ;FLAGS WHEN TOO MANY ERRORS HAVE BEEN PRINTED FOR A BANK
1218 002360  000000               READONLY:0                     ;FLAG TO PATTERNS TO READ ONLY
1219 002362  000000  000000       TESTADD:0,0                    ;THE ADDRESS TO RUN CSR TESTS ON
1220 002366  000000               UNITOP: 0                      ;HIGHEST ACCESSABLE BANK OF MEMORY THRU UNIBUS MAP
1221 002370  000000               STOPOK: 0                      ;FLAG TO ALLOW STOPPING WITH SWITCH REGISTER
1222 002372  000000               APTPAR: .WORD   0              ;AMOUNT OF PARITY MEMORY ACCORDING TO API
1223 002374  000000               APTECC: .WORD   0              ;AMOUNT OF ECC MEMORY ACCORDING TO APT
1224 002376  000000               NOFSMODE:0                     ;FLAG TO DISABLE FIELD SERVICE MODE
1225 002400  000000               NOERROR:0                      ;'THIS IS NOT AN ERROR'' FLAG
1226 002402  000000               LOADBANK:0                     ;BANK LOADERS ARE RELOCATED TO
1227 002404  000000               TEMP:   0                      ;USED FOR JUNK
1228 002406  000000               QUICK:  0                      ;QUICK STOP FLAG FOR APT POWER FAIL
1229 002410  000000               NOSCOPE:0                      ;'NO SCOPE LOOP ALLOWED'' FLAG
1230 002412  000000               FSINFLAG:0                     ;'FIELD SERVICE - NO INTERNAL INTERLEAVE'' FLAG
1231 002414  000000               APTSIZE:0                      ;APT SIZING INFO FLAG
1232 002416  000000               FS7FLAG:0                      ;TRUE WHEN IN FIELD SERVICE COMMAND 7
1237 002420  000000               CONFGERROR:0                   ;CONFIGURATION ERROR FLAG
1238 002422  000000               I:      0                      ;USED FOR GENERAL PURPOSE INDEXING
1239 002424  000000               NO22BIT:0                      ;NO 22-BIT MODE FLAG
1240 002426  000000               NOSUPER:0                      ;NO SUPERVISOR MODE FLAG
1241 002430  000000               ERRADD: .WORD   0              ;HOLDS THE CSR'S ERROR ADDRESS
1242 002432  000000  000000 000000 CSRINFO:0,0,0,0,0,0,0,0      ;USED TO STORE INFORMATION ABOUT THE 16
     002440  000000  000000 000000
     002446  000000  000000
1243 002452  000000  000000 000000        0,0,0,0,0,0,0,0      ;POSSIBLE CSR'S
     002460  000000  000000 000000
     002466  000000  000000
1244 002472  000000               LINK1:  0                      ;USED TO HOLD LINKS TO PATTERNS WHICH
1245 002474  000000               LINK2:  0                      ;CAN EXECUTE IN THE PAR/PDR'S OR NOT
1246 002476  000000               CSRHOLD:0                      ;USED TO STORE CSR VALUES FOR CSR TESTS
1247 002500  000000               KFLAG:  0                      ;USED TO FLAG MF11S-K MEMORY TO TESTS
1248 002502  000000  000000       PGMCSR: .WORD   0,0            ;POINTS TO PROGRAM CSR
1249 002506  000000               INHECC: .WORD   0              ;FLAGS INHIBIT ECC TESTS ON RELOCATION
1250 002510  000000               INHBANK:.WORD   0              ;
1251 002512  000000               FULLREL:.WORD   0              ;
1289 002514                       $CMTGE: ;*END OF COMMON TAGS
```

```
                                        .SBTTL  VARIABLES       INITIALIZED TO NON ZERO
002514  000001  000000          CACHKN: 1,0                     ;CACHE CONSTANT (MOVED TO CONTRL TO TURN ON CACHE)
002520  001415                  CACHKF: 1415                    ;CACHE CONSTANT (MOVED TO CONTRL TO TURN OFF CACHE)
002522  040000                  TESTMODE:40000                  ;USED TO SELECT THE PROPER TEST MODE FOR A PATTERN RUN
002524  000012                  ERRMAX: 10.                     ;MAX # OF ERRORS PER BANK WITH SW11
002526  000167                  LASTBANK:167                    ;HIGHEST BANK OF MEMORY
002530  170000                  LASTBLOCK:170000                ;HIGHEST BANK OF MEMORY+1 (IN PAR FORMAT)
002532  000031                  SOBK:   25.                     ;SOB CONSTANT
002534  002000                  KSTACK: STACK                   ;STACK BEGINNING
002536  000001                  LOADHOME:1                      ;HOME BANK OF LOADERS
002540  177777                  WORST:  177777                  ;SET IF TESTING BANKS IN WORST FIRST MODE(1ST PASS)
002542  176543                  SEEDHI: 176543                  ;WORKING SEED HI (USED FOR RANDOM NUMBER GENERATOR)
002544  123456                  SEEDLO: 123456                  ;WORKING SEED LO (USED FOR RANDOM NUMBER GENERATOR)
002546  176543                  MSEEDH: 176543                  ;MASTER SEED HI (USED FOR RANDOM NUMBER GENERATOR)
002550  123456                  MSEEDL: 123456                  ;MASTER SEED LO (USED FOR RANDOM NUMBER GENERATOR)
002552  177777                  HEADER: 177777                  ;USED TO PRINT HEADINGS ONLY ONCE
002554  177777                  ONES:   177777                  ;FOR AID IN 'MOV'' INSTRUCTIONS
002556  000003                  FLIPLOC:3                       ;COUNTER FOR FLIPING DATA ON WORST CASE NOISE TEST
002560  052525                  SOFTPAT:52525                   ;PATTERN FOR SOFT ERROR BACKGROUND TESTS
002562  000000                  $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
002564  000000                  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
002566  000000                  RESTART:0                       ;RESTART (START ADD 202) FLAG
002570  000000                  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS

                                ;********************** NOTE THESE TWO LOCATIONS MUST STAY TOGETHER ***********
002572  000377                  BAKPAT: .WORD   377             ;BACKGROUND PATTERN                            *
002574  177400                  SWAPAT: .WORD   177400          ;SWAPPED BAKPAT                                *
                                ;*********************************************************************************

002576  177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
002600  177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
002602  177560                  $TKS:   177560                  ;;TTY KBD STATUS
002604  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
002606  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
002610  177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
002612     012                  $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED''
002613     207  377  377        $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
002616     000
002617     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
002620     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
002621     012  000             $LF:    .ASCIZ  <12>            ;;LINE FEED
                                        .EVEN
```

```
                                        .SBTTL  CONFIGURATION TABLE
                                ;CONFIG:FIRST 16K CONFIGURATION WORDS (2 EACH)
                                ;       2ND     16K CONFIGURATION WORDS (2 EACH)
                                ;       .........................
                                ;       200TH   16K CONFIGURATION WORDS (2 EACH)
                                ;
                                ;CONFIGURATION WORDS:
                                ;                       LOW:    BIT 0     ERRORS PRESENT
                                ;                               BIT 1     MEMORY SUCESSFULLY ACCESSED
                                ;                               BIT 2-4   RESERVED
                                ;                               BIT 5     SKIP ECC LOGIC TESTS FLAG (1=SKIP)
                                ;                               BIT 6     PROTECTED REGION OF ECC MEMORY
                                ;                               BIT 7     PROTECTED (PROGRAM SPACE)
                                ;                               BIT 8-11  CSR CODE
                                ;                               BIT 12-15 INTERLEAVED CSR CODE
                                ;                       HIGH:   BIT 0-7   NUMBER OF ERRORS
                                ;                               BIT 8-10  MEMORY TYPE
                                ;                               BIT 11    INTERLEAVED BOARD TYPE (0=128K, 1 64K)
                                ;                               BIT 12    INTERLEAVE ENABLED
                                ;                               BIT 13    'BACKGROUND PATTERN VALID'' FLAG
                                ;                               BIT 14    BANK SELECTED FOR TEST BY FIELD SERVICE MODE
                                ;                               BIT 15    LOADERS HOME BANK
1360 002624  000201             CONFIG: .REPT   201
1363 003630                     CONFIEND:
```

```
1365                                            .SBTTL  ********** MAIN **********
1366 003630                                START:  SUBTST  <<INITIALIZE VARIABLES TO ZERO>>
                                                ;*************************************************************************
                                                ;*SUBTEST       INITIALIZE VARIABLES TO ZERO
                                                ;*************************************************************************
1370 003630  000005                             RESET
1371 003632  013706  002534                     MOV     KSTACK,SP       ;;SETUP THE STACK POINTER
1377 003636  012700  002000                     MOV     #$CMTAG,R0      ;;FIRST LOCATION TO BE CLEARED
1378 003642  005020                        1$:  CLR     (R0)+           ;;CLEAR MEMORY LOCATION
1379 003644  022700  002514                     CMP     #$CMTGE,R0      ;;DONE?
1380 003650  001374                             BNE     1$              ;LOOP BACK IF NO
1381 003652                                     SUBTST  <<CLEAR NON-PROGRAM SPACE>>
                                                ;*************************************************************************
                                                ;*SUBTEST       CLEAR NON-PROGRAM SPACE
                                                ;*************************************************************************
1382                                            ;THIS ATTEMPTS TO GET RID OF ANY PARITY ERRORS BY WRITING INTO
1383                                            ;EVERY LOCATION THAT IS NOT LOADED INTO BY THE PROGRAM OR ALLOCATED
1384                                            ;TO THE XXDP LOADERS
1385 003652  012737  000001  002074             MOV     #1,NOPAR                ;PARITY ACTION = COUNT & IGNORE
1386 003660  012700  001000                     MOV     #1000,R0
1387 003664  012720  000000                2$:  MOV     #0,(R0)+                ;CLEAR THE STACK AREA
1388 003670  020027  002000                     CMP     R0,#STACK
1389 003674  103773                             BLO     2$
1390 003676  012700  077662                     MOV     #END+2,R0
1391 003702  012720  000000                3$:  MOV     #0,(R0)+                ;CLEAR FROM THE END OF PROGRAM TO XXDP LOADERS
1392 003706  020027  152364                     CMP     R0,#152364
1393 003712  103773                             BLO     3$
1394 003714  005037  002074                     CLR     NOPAR                   ;RESTORE DEFAULT PARITY ACTION
1395
```

```
1404 003720                                     SUBTST  <<TYPE OF SYSTEM SIZER>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       TYPE OF SYSTEM SIZER
                                        ;*****************************************************************************
1405 003720                             4$:     SET4    #4$
1406 003726  005737  177746                    TST     CONTRL                  ;SEE IF CACHE REGISTER RESPONDS
1407 003732                                     SET4    #9$                     ;YES - DO WE HAVE 11/44 TYPE CACHE
1408 003740  005737  177750                    TST     MAINT                   ;OR 11/60 TYPE CACHE?
1409 003744  000411                            BR      5$                      ;BRANCH IF 11/44 TYPE CACHE
1410 003746  012737  000014  002520  9$:      MOV     #14,CACHKF              ;TURN OFF CONSTANT FOR 11/60 CACHE
1411 003754  000405                            BR      5$
1412 003756  005037  002514              4$:     CLR     CACHKN                  ;NO CACHE ON SYSTEM
1413 003762  012737  002306  066404            MOV     #ZEROS,DT14             ;DO NOT PRINT CONTRL ERROR MESSAGES
1414 003770                             5$:     SET4    #6$
1415 003776  005737  172516                    TST     MMR3                    ;DO WE HAVE AN MMR3?
1416 004002  005037  172516                    CLR     MMR3                    ;YES WE DO
1417 004006  052737  000020  172516            BIS     #BIT4,MMR3              ;SEE IF THERE IS 22-BIT MODE
1418 004014  032737  000020  172516            BIT     #BIT4,MMR3
1419 004022  001023                            BNE     8$                      ;BRANCH IF 22-BIT RELOCATION
1420 004024  000411                            BR      7$                      ;BRANCH IF MMR3 BUT NO 22-BIT RELOC.
1421                                    ;* 11/34 TYPE MACHINES ENTER HERE
1422 004026  012737  140000  002522  6$:      MOV     #140000,TESTMODE        ;MAKE TEST MODE USER
1423 004034  005237  002426                    INC     NOSUPER                 ;NO SUPERVISOR MODE
1424 004040  005037  066254                    CLR     DT5+10
1425 004044  005037  066414                    CLR     DT14+10
1426                                    ;* 11/45 TYPE MACHINES ENTER HERE
1427 004050  005237  002424              7$:     INC     NO22BIT                 ;NO 22 BIT MODE
1428 004054  012737  000007  002526            MOV     #7,LASTBANK             ;124K MEMORY MAX. MEMORY SIZE
1429 004062  005037  066256                    CLR     DT5+12                  ;DO NOT TRY TO PRINT ERROR REGISTER
1430 004066  005037  066416                    CLR     DT14+12                 ;ERROR MESSAGES, BECAUSE THERE IS
1431                                                                            ;IS NO ERROR REGISTER!
1432 004072                             8$:     RES4
```

```
                                        SUBTST  <<INITIALIZE VARIABLES TO NON ZERO>>
                                ;*****************************************************************************
                                ;*SUBTEST        INITIALIZE VARIABLES TO NON ZERO
                                ;*****************************************************************************
                                        SET     WORST
004120  012737  000003  002556          MOV     #3,FLIPLOC
004126                                  SET     HEADER
004134  012737  176543  002546          MOV     #176543,MSEEDH
004142  012737  123456  002550          MOV     #123456,MSEEDL
004150  013737  002546  002542          MOV     MSEEDH,SEEDHI   ;PRIME THE RANDOM NUMBER GENERATOR
004156  013737  002550  002544          MOV     MSEEDL,SEEDLO   ;BOTH HIGH AND LOW WORDS
004164  012737  000377  002572          MOV     #377,BAKPAT
004172  012737  177400  002574          MOV     #177400,SWAPAT
004200                                  SUBTST  <<INITIALIZE VECTORS>>
                                ;*****************************************************************************
                                ;*SUBTEST        INITIALIZE VECTORS
                                ;*****************************************************************************
004200  012737  054154  000020          MOV     #$SCOPE,IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
004206  012737  000340  000022          MOV     #340,IOTVEC+2   ;;LEVEL 7
004214  012737  054446  000030          MOV     #$ERROR,EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
004222  012737  000340  000032          MOV     #340,EMTVEC+2   ;;LEVEL 7
004230  012737  061372  000034          MOV     #$TRAP,TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
004236  012737  000340  000036          MOV     #340,TRAPVEC+2;LEVEL 7
004244  012737  050450  000024          MOV     #$PWRDN,PWRVEC ;;POWER FAILURE VECTOR
004252  012737  000340  000026          MOV     #340,PWRVEC+2   ;;LEVEL 7
004260  012737  036420  000114          MOV     #PARITY,PARVEC;GET READY FOR PARITY ERRORS
004266  012737  000340  000116          MOV     #340,PARVEC+2
004274  012737  036614  000010          MOV     #PDP1105,RESVEC;RESERVED INSTRUCTION TRAP
004302  012737  000340  000012          MOV     #340,RESVEC+2
004310  012737  036570  000004          MOV     #TIMEOUT,ERRVEC;SETUP TIMEOUT ERRORS
004316  012737  000340  000006          MOV     #340,ERRVEC+2   ;SET PRIORITY OF ERROR TRAPS
004324  012737  036602  000250          MOV     #MMTRAP,MMVEC   ;VECTOR FOR MEMORY MANAGEMENT
004332  012737  000340  000252          MOV     #340,MMVEC+2
004340  104423                          CACHON                  ;TURN CACHE ON
```

```
                                        SUBTST  <<INITIALIZE PATTERNS>>
                                ;*******************************************************************************
                                ;*SUBTEST        INITIALIZE PATTERNS
                                ;*******************************************************************************
                                        ;THE APT E-TABLE DETERMINES WHICH PATTERNS ARE GOING TO BE RUN.
                                        ;EACH BIT SET REPRESENTS A PATTERN TABLE ENTRY THAT IS TO BE LEFT
                                        ;ALONE (TO BE RUN).  EACH BIT CLEARED REPRESENTS A PATTERN TABLE ENTRY
                                        ;THAT IS TO BE OVERLAYED WITH THE ADDRESS OF A NULL PATTERN.
1478 004342  012700  061342            MOV     #$DDW0,R0
1479 004346  012001                    MOV     (R0)+,R1
1480 004350  012703  016264            MOV     #MKCSRT,R3
1481 004354  012702  000020            MOV     #16.,R2
1482 004360  004737  004460            CALL    PATPLUG
1483 004364  012001                    MOV     (R0)+,R1
1484 004366  012702  000010            MOV     #8.,R2
1485 004372  004737  004460            CALL    PATPLUG
1486 004376  012001                    MOV     (R0)+,R1
1487 004400  012703  016514            MOV     #MKPAT,R3
1488 004404  012702  000020            MOV     #16.,R2
1489 004410  004737  004460            CALL    PATPLUG
1490 004414  012001                    MOV     (R0)+,R1
1491 004416  012702  000010            MOV     #8.,R2
1492 004422  004737  004460            CALL    PATPLUG
1493 004426  012001                    MOV     (R0)+,R1
1494 004430  012703  016700            MOV     #MJPAT,R3
1495 004434  012702  000020            MOV     #16.,R2
1496 004440  004737  004460            CALL    PATPLUG
1497 004444  012001                    MOV     (R0)+,R1
1498 004446  012702  000010            MOV     #8.,R2
1499 004452  004737  004460            CALL    PATPLUG
1500 004456  000417                    BR      SUBAAA
1501
1502 004460                     PATPLUG:SUBTST  <<SUBR  PLUG IN NULL PATTERNS>>
                                ;*******************************************************************************
                                ;*SUBTEST        SUBR    PLUG IN NULL PATTERNS
                                ;*******************************************************************************
1503 004460                             FOR I := #1 TO R2
1504 004466  006001                       ROR   R1
1505 004470                               ON.NCERROR            ;IF CARRY CLEAR
1506 004472  012713  025074                 MOV  #MT0999,(R3)
1507 004476                               END ;OF ON.ERROR
1508 004476  062703  000002               ADD #2,R3
1509 004502                             END ;OF FOR
1510 004514  000207                     RETURN
```

```
1513 004516                                SUBAAA: SUBTST  <<CLEAR THE CONFIGURATION TABLE>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       CLEAR THE CONFIGURATION TABLE
                                           ;*****************************************************************************
1514                                               ;THIS ZEROS (UNLESS WE STARTED AT ADDRESS 202) THE CONFIG TABLE
1515                                               ;WHICH IS FULLY DISCRIBED AT LOCATION 'CONFIG'.
1516                                               .ENABLE LSB
1517 004516                                       IF RESTART IS FALSE
1518 004524  012700  002624                         MOV   #CONFIG,R0
1519 004530  005020                        1$:      CLR   (R0)+
1520 004532  022700  003630                         CMP   #CONFIEND,R0
1521 004536  001374                                 BNE   1$
1522 004540                                       END ;OF IF RESTART
1523                                               .DSABL  LSB
1524 004540  012737  000002  002104               MOV     #BIT1,CPUBIT            ;SET ID BIT
1525 004546                                       SUBTST  <<SIZE FOR A HARDWARE SWITCH REGISTER>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       SIZE FOR A HARDWARE SWITCH REGISTER
                                           ;*****************************************************************************
1526                                               ;;IF NOT FOUND OR IT IS
1527                                               ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1528                                               .ENABL  LSB
1529 004546                                       SET4    #3$                     ;TRAPS TO 4 GOTO 3$
1530 004554  012737  177570  002576               MOV     #DSWR,SWR               ;;SETUP FOR A HARDWARE SWITCH REGISTER
1531 004562  012737  177570  002600               MOV     #DDISP,DISPLAY          ;;AND A HARDWARE DISPLAY REGISTER
1532 004570                                       IF #-1 EQ @SWR                  ;IF NO TRAP FROM REFERENCE TO @SWR AND @SWR - #-1
1533 004600  000403                                 BR    2$                      ;;BRANCH IF NO TIMEOUT
1534 004602  012716  004610                3$:      MOV   #2$,(SP)                ;;SET UP FOR TRAP RETURN
1535 004606  000002                                 RTI
1536 004610                                2$:      RES4                          ;RESET TRAPS TO 4 TO DEFAULT
1537 004630  012737  000176  002576                 MOV   #SWREG,SWR              ;;POINT TO SOFTWARE SWR
1538 004636  012737  000174  002600                 MOV   #DISPREG,DISPLAY
1539 004644                                       END ;OF IF #-1
1540                                              .DSABL  LSB
```

```
1543 004644                          SUBAAB: SUBTST  <<SETUP ACT, APT, & XXDP>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       SETUP ACT, APT, & XXDP
                                     ;*****************************************************************************
1544                                         ;THIS SETS UP A BUNCH OF FLAGS TO TELL THE PROGRAM EVERYTHING
1545                                         ;IT CARES TO KNOW ABOUT APT, ACT, & XXDP.
1546 004644  005037  061264                 CLR     $PASS           ;CLEAR PASS COUNT
1547 004650                                  IFB #BIT5 SET.IN $ENVM
1548 004660                                    SET   $TPFLG          ;INDICATE NO TERMINAL
1549 004666                                  END ;OF IFB #BIT5
1550 004666                                  IFB #BIT7 SET.IN $ENVM
1551 004676                                    SET   APTSIZE
1552 004704                                  END ;OF IFB #BIT7
1553 004704                                  IFB $ENV EQ #1
1554 004714                                    SET   APTFLAG,QVFLAG,$AUTO,QUICK
1555 004744  012737  043650  000024            MOV   #APTDOWN,PWRVEC
1556 004752  012737  061300  002576            MOV   #$SWREG,SWR     ;USE APT SWR
1557 004760                                  ELSE
1558 004762                                    IF 42 NE #STACK AND 42 NE #0
1559 005000                                      SET QVFLAG,$AUTO
1560 005014                                      IF 42 EQ #$ENDAD
1561 005024                                        SET       ACTFLAG
1562 005032                                      ELSE
1563 005034                                        SET       XXDPCHAIN
1564 005042                                      END ;OF IF 42
1565 005042                                    END ;OF IF 42
1566 005042                                  END ;OF IFB $ENV
```

```
1568 005042                                    SUBTST  <<PROTECT PROGRAM & LOADERS>>
                                       ;*****************************************************************************
                                       ;*SUBTEST       PROTECT PROGRAM & LOADERS
                                       ;*****************************************************************************
1569 005042  052737  000200  002624             BIS     #BIT7,CONFIG            ;PROTECT PROGRAM SPACE (BANK 0)
1570 005050  052737  000200  002630             BIS     #BIT7,CONFIG+4          ;PROTECT LOADER SPACE (BANK 1)
1571 005056                                     IF #$ENDAD NE 42                ;NOT ACT-11?
1572 005066                                       TYPE  MSG000                  ;TYPE PROGRAM TITLE
1573 005072                                     END ;OF IF #$ENDAD
1574
1575 005072                                    SUBTST  <<CHECK SYSTEM FOR CACHE>>
                                       ;*****************************************************************************
                                       ;*SUBTEST       CHECK SYSTEM FOR CACHE
                                       ;*****************************************************************************
1576                                           ;* THIS FIGURES OUT IF THERE IS A CACHE ON THE SYSTEM,
1577                                           ;* WHAT TYPE OF SYSTEM IT IS, AND WHETHER IT IS ENABLED
1578                                           ;* OR DISABLED.
1579 005072                                     SET4    #3$
1580 005100  005737  177746                    TST     CONTRL                  ;IS THERE A CONTROL REGISTER?
1581 005104                                     SET4    #2$
1582 005112  005737  177750                    TST     MAINT                   ;IS THERE A MAINTENANCE REGISTER?
1583 005116                                     SET4    #1$
1584 005124  005737  177754                    TST     DATARG                  ;IS THERE A DATA REGISTER?
1585 005130                                     TYPE    MSG117                  ; 11/44
1586 005134  000405                             BR      4$
1587 005136                            1$:      TYPE    MSG116                  ; 11/34
1588 005142  000402                             BR      4$
1589 005144                            2$:      TYPE    MSG118                  ; 11/60
1590 005150  052737  000014  177746     4$:      BIS     #BIT2!BIT3,CONTRL       ;SET CACHE DISABLE BITS
1591 005156  042737  000014  177746              BIC     #BIT2!BIT3,CONTRL       ;CLEAR CACHE DISABLE BITS
1592 005164  032737  000004  177746              BIT     #BIT2,CONTRL            ;IS THE BIT SET?
1593 005172  001004                             BNE     7$                      ;BRANCH IF THE BIT IS SET
1594 005174  032737  000010  177746              BIT     #BIT3,CONTRL            ;IS THE BIT SET?
1595 005202  001413                             BEQ     6$                      ;BRANCH IF THE BIT IS SET
1596 005204                            7$:      TYPE    MSG121                  ; CACHE BYPASSED
1597 005210  104424                             CACHOFF
1598 005212  013737  002514  002516              MOV     CACHKN,CACHKN+2         ;SAVE INFO ABOUT CACHE
1599 005220  005037  002514                     CLR     CACHKN                  ;CACHE CANNOT BE USED - IT'S BYPASSED
1600 005224  000404                             BR      8$
1601 005226                            3$:      TYPE    MSG119                  ; NO
1602 005232                            6$:      TYPE    MSG120                  ; CACHE AVAILABLE
1603
```

```
                                        SUBTST  <<SETUP USER & SUPERVISOR STACK>>
                                ;*******************************************************************************
                                ;*SUBTEST       SETUP USER & SUPERVISOR STACK
                                ;*******************************************************************************
1651 005236  104421             8$:     DEENERGIZE              ;TURN OFF MEMORY MANAGEMENT
1652 005240  005737  002426             TST     NOSUPER         ;IS THERE A SUPERVISOR MODE?
1653 005244  001011                     BNE     5$              ;NO-SKIP SUPERVISOR SETUP.
1654
1655                                    ;SET PREVIOUS MODE TO SUPERVISOR
1656 005246  042737  030000  177776     BIC     #BIT13!BIT12,PSW
1657 005254  052737  010000  177776     BIS     #BIT12,PSW
1658
1659 005262                             PUSH    #SUPSTK
1660 005266  006606                     MTPI    SSP
1661
1662                                    ;SET PREVIOUS MODE TO USER
1663 005270  052737  030000  177776 5$: BIS     #BIT13.BIT12,PSW
1664
1665 005276                             PUSH    #USESTK
1666 005302  006606                     MTPI    USP
1667
1668 005304                             SUBTST  <<GET SOFTWARE SWITCH REGISTER IF NECESSARY>>
                                ;*******************************************************************************
                                ;*SUBTEST       GET SOFTWARE SWITCH REGISTER IF NECESSARY
                                ;*******************************************************************************
1672 005304                             IF $AUTO IS FALSE                       ;IF NOT(APT OR ACT)
1673 005312                               IF SWR EQ #SWREG              ;IF SOFTWARE SWITCH REG SELECTED
1674 005322  104407                         GTSWR                               ;;GET SOFT-SWR SETTINGS
1675 005324                               END ;OF IF SWR
1676 005324                             END ;OF IF $AUTO
1680
1681 005324                             SUBTST  <<GET MEMORY MANAGEMENT READY>>
                                ;*******************************************************************************
                                ;*SUBTEST       GET MEMORY MANAGEMENT READY
                                ;*******************************************************************************
1685 005324  104422                     KMAP                    ;MAP KERNEL SPACE 1 TO 1
1689 005326                             MAP                     ;MAP SUPERVISOR SPACE (TEST AREA) 1 TO 1
1690 005342  104420                     ENERGIZE                ;TURN ON MEMORY MANAGEMENT
```

```
1692
```

```
1695 005344                             NEWTST  <<BIT TEST OF ALL CSR'S>>
                                ;*******************************************************************************
                                ;*TEST 1        BIT TEST OF ALL CSR'S
                                ;*******************************************************************************
     005344  000004             TST1:   SCOPE
1696                            ;* THE FIRST PART OF THE CONFIGURATION ANALYSIS DOES THE FOLLOWING:
1697                            ;*      1) FINDS WHICH CSR'S RESPOND, AND PUTS THEM INTO THE CSR INFORMATION
1698                            ;*              TABLE, AND STORES ANOTHER BIT FOR 'TOTCSRS'.
1699                            ;*      2) TESTS THE CSR BITS COMMON TO ALL CSR'S.
1700                            ;*      3) FIGURES OUT IF THERE IS AN EUB BIT, AN ECC BIT, AND THE EXISTANCE
1701                            ;*              OF THE ERROR ADDRESS BITS, AND MARKS THIS IN THE CSR
1702                            ;*              INFORMATION TABLE.
1703                            ;*      4) TESTS THE BITS PARTICULAR TO THAT TYPE OF CSR.
1704                            ;*      5) IF ANY BITS TEST BAD IN THE CSR UNDER TEST, THE CSR OK BIT IN THE
1705                            ;*              CSR INFORMATION TABLE IS CLEARED.
1706                            ;* THE INFORMATION BITS ONE THROUGH THREE FORM A CODE WHICH GIVES THE TYPE
1707                            ;* OF CSR:
1708                            ;*                      ERR. ADDR.   PARITY  NOT EUB
1709                            ;*       TYPE             BIT2        BIT1    BIT0      CODE TOTALS
1710                            ;* UNIBUS PARITY          1           1       1           7
1711                            ;*     MS11-L             1           1       0           6
1712                            ;*     MF11S-K            1           0       1           5
1713                            ;*     MS11-M             1           0       0           4
1714                            ;* 11/45 BIPOLAR          0           1       1           3
1715                            ;*
1716                            ;* THIS MEMORY CODE WILL BE USED IN THE SECOND PART OF THIS ANALYSIS
1717                            ;*
1718 005346  005005                        CLR     R5              ;R5 IS THE TOTAL CSR NUMBER
1719 005350  005000                        CLR     R0              ;R0 IS A TABLE INDEX
1720 005352  012703  172100                MOV     #CSRADD,R3      ;R3 HAS THE CSR ADDRESS
1721 005356  012737  000001  002074        MOV     #1,NOPAR        ;IGNORE PARITY ERRORS
1722 005364                                SET4    #2$
1723 005372  005713                1$:     TST     (R3)    ;DOES THE CSR RESPOND?
1724 005374  052705  000001                BIS     #1,R5
1725 005400  005004                        CLR     R4              ;CLEAR THE LAST CSR INDICATOR
1726 005402  052760  000007  002432        BIS     #7,CSRINFO(R0)  ;SET ALL THE MEMORY INFO BITS
1727 005410  052760  000030  002432        BIS     #BIT4!BIT3,CSRINFO(R0)  ;YES-MARK IT IN CSR INFORMATION TABLE
1728 005416  004537  005752                JSR     R5,TEST         ;TEST BIT 0 AND 15
1729 005422  100001                        .WORD   BIT15!BIT0
1730 005424  012713  040000                MOV     #BIT14,(R3)     ;IS THERE A BIT 14 RESPONDING
1731 005430  032713  040000                BIT     #BIT14,(R3)     ;(IT'S THE EUB BIT)
1732 005434  001403                        BEQ     3$              ;BRANCH IF NO EUB BIT
1733 005436  042760  000001  002432        BIC     #BIT0,CSRINFO(R0) ;CLEAR EUB INFO IN THE CSR TABLE
1734 005444  005013                3$:     CLR     (R3)            ;CLEAR THE CSR UNDER TEST
1735 005446  012713  020000                MOV     #BIT13,(R3)     ;DOES BIT 13 RESPOND
1736 005452  032713  020000                BIT     #BIT13,(R3)     ;(TO TEST FOR ECC CSR)
1737 005456  001417                        BEQ     4$              ;BRANCH IF NOT ECC CSR
1738 005460  042760  000002  002432        BIC     #BIT1,CSRINFO(R0) ;CLEAR PARITY INFO IN THE CSR TABLE
1739 005466  012713  020000                MOV     #BIT13,(R3)     ;SET THE INHIBIT MODE POINTER TO 1ST 16K
1740 005472  004537  005752                JSR     R5,TEST         ;TEST BIT 3
1741 005476  000010                        .WORD   BIT3
1742 005500  012713  020000                MOV     #BIT13,(R3)
1743 005504  004537  005752                JSR     R5,TEST         ;TEST BIT 1 AND 4
1744 005510  000022                        .WORD   BIT4!BIT1
1745 005512  012713  020000                MOV     #BIT13,(R3)
```

```
1747 005516 004537 005752        4$:    JSR     R5,TEST          ;TEST BIT 2
1748 005522 000004                      .WORD   BIT2
1749 005524 005013                      CLR     (R3)
1750 005526 052713 007740               BIS     #7740,(R3)       ;ARE THERE ERROR ADDRESS BITS?
1751 005532 032713 007740               BIT     #7740,(R3)
1752 005536 001404                      BEQ     5$               ;BRANCH IF NO ERROR ADDR. BITS.
1753 005540 004537 005752               JSR     R5,TEST          ;TEST BITS 5->11
1754 005544 007740                      .WORD   7740
1755 005546 000403                      BR      6$               ;SKIP OVER THE INFORMATION REPORTING
1756 005550 042760 000004 002432 5$:    BIC     #BIT2,CSRINFO(R0) ;REPORT THAT THERE ARE NO ERROR ADDRESS BITS.
1757 005556 032760 000002 002432 6$:    BIT     #BIT1,CSRINFO(R0) ;IS THIS CSR AN ECC CSR?
1758 005564 001014                      BNE     7$               ;BRANCH IF NOT
1759 005566 032760 000001 002432        BIT     #BIT0,CSRINFO(R0) ;IS THE EUB BIT SET?
1760 005574 001410                      BEQ     7$               ;BRANCH IF IT IS
1761                                    ;WE MUST NOW TEST FOR MS11-M ON THE UNIBUS
1762 005576 012713 007760               MOV     #7760,(R3)       ;PUT PATTERN & SBE BIT INTO BITS 4->11
1763 005602 022713 007760               CMP     #7760,(R3)       ;ARE THEY STILL THERE?
1764 005606 001403                      BEQ     7$               ;YES - BRANCH FOR MF11S-K CSR
1765 005610 042760 000001 002432        BIC     #BIT0,CSRINFO(R0) ;NO - SET EUB BIT FOR MS11-M
1766 005616 005013                7$:   CLR     (R3)             ;CLEAR CSR
1767 005620 022760 000040 002432        CMP     #40,CSRINFO(R0)  ;IS THIS A LEGAL CONFIGURATION?
1768 005626 100004                      BPL     10$              ;BRANCH IF IT'S LEGAL
1769 005630 016037 002432 002050        MOV     CSRINFO(R0),BAD
1770 005636 104021                      ERROR   +21              ;ILLEGAL TYPE ERROR
1771 005640 032760 000004 002432 10$:   BIT     #BIT2,CSRINFO(R0) ;DOES THIS CSR HAVE ERROR BITS
1772 005646 001016                      BNE     2$               ;BRANCH IF TRUE
1773 005650 032760 000002 002432        BIT     #BIT1,CSRINFO(R0) ;ARE THE OTHER 2 BITS SET?
1774 005656 001404                      BEQ     11$              ;BRANCH IF NOT
1775 005660 032760 000001 002432        BIT     #BIT0,CSRINFO(R0)
1776 005666 001006                      BNE     2$
1777 005670 016037 002432 002050 11$:   MOV     CSRINFO(R0),BAD
1778 005676 104021                      ERROR   +21              ;ILLEGAL TYPE ERROR
1779 005700 005060 002432               CLR     CSRINFO(R0)      ;CLEAR THE CSR INFO-IT WILL NOT EXIST IN THE PROGRAM
1780 005704 062700 000002        2$:    ADD     #2,R0            ;INCREMENT TO NEXT CSR TABLE
1781 005710 062703 000002                ADD     #2,R3            ;INCREMENT TO NEXT CSR
1782 005714 006305                       ASL     R5
1783 005716 103001                       BCC     8$               ;IS THERE A CSR 0?
1784 005720 005204                       INC     R4               ;YES - SET CSR PRESENT FLAG
1785 005722 022700 000040        8$:     CMP     #40,R0           ;ARE WE DONE?
1786 005726 001221                       BNE     1$               ;BRANCH IF MORE TO DO
1787 005730 000241                       CLC
1788 005732 006005                       ROR     R5               ;RESYNC R5
1789 005734 005704                       TST     R4               ;WAS THERE A CSR 0?
1790 005736 001402                       BEQ     9$               ;BRANCH IF NOT
1791 005740 052705 100000                BIS     #BIT15,R5        ;YES - SET IN THE CSR TABLE
1792 005744 010537 002216        9$:     MOV     R5,TOTCSRS       ;STORE R5 IN TOTCSRS
1793 005750 000436                       BR      ANA2             ;JUMP OVER SUBROUTINE
1794                                     ;THIS SUBROUTINE TESTS THE CSR BITS
1795 005752 012501               TEST:   MOV     (R5)+,R1         ;GET THE BIT TO TEST
1796 005754 050113                       BIS     R1,(R3)          ;SET THAT IN THE CSR UNDER TEST
1797 005756 030113                       BIT     R1,(R3)          ;IS THE BIT STILL THERE?
1798 005760 001013                       BNE     1$               ;BRANCH IF STILL THERE
1799 005762 011337 002144                MOV     (R3),CSR         ;READ CSR
1800 005766 010137 002042                MOV     R1,GOOD
1801 005772 032713 100020                BIT     #BIT15!BIT4,(R3);IS IT BECAUSE OF A SBE OR DBE?
1802 005776 001004                       BNE     1$               ;BRANCH IF IT IS
1803 006000 104035                       ERROR   +35              ;BIT SET ERROR
```

```
1804 006002  042760  000010  002432          BIC     #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT
1805 006010  040113                  1$:     BIC     R1,(R3)           ;CLEAR THE SELECTED BIT
1806 006012  030113                          BIT     R1,(R3)           ;IS IT CLEARED?
1807 006014  001413                          BEQ     2$                ;BRANCH IF IT IS CLEARED
1808 006016  011337  002144                  MOV     (R3),CSR          ;READ CSR
1809 006022  010137  002042                  MOV     R1,GOOD
1810 006026  032713  100020                  BIT     #BIT15.BIT4,(R3);IS IT BECAUSE OF A SBE OR DBE?
1811 006032  001004                          BNE     2$                ;BRANCH IF TRUE
1812 006034  104010                          ERROR   +10               ;BIT CLEAR ERROR
1813 006036  042760  000010  002432          BIC     #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT
1814 006044  000205                  2$:     RTS     R5
1815
```

```
 1817 006046                    ANA2:   SUBTST  <<MATCH ALL CSR'S WITH MEMORY>>
                                ;*****************************************************************************
                                ;*SUBTEST       MATCH ALL CSR'S WITH MEMORY
                                ;*****************************************************************************
 1818                           ;* THE SECOND PART OF THE ANALYSIS MATCHES UP THE CSR'S WITH THE MEMORY, AND
 1819                           ;* INSTALLS ALL THE INFORMATION FOUND IN THE CONFIGURATION TABLE.  FOR ECC,
 1820                           ;* THIS IS DONE BY TAKING EACH CSR FOUND IN THE PREVIOUS SECTION SEQUENTIALLY
 1821                           ;* AND CHECKING THROUGH ALL OF MEMORY, ONE BANK AT AT TIME, TO SEE WHICH BANKS
 1822                           ;* RESPOND TO THE CSR IN QUESTION. THE FIRST DOUBLE WORD PAIR IN EACH BANK ARE
 1823                           ;* WRITTEN WITH DATA AND DIAGNOSTIC CHECK MODE SET IN THE CSR IN ORDER TO AC-
 1824                           ;* COMPLISH THIS. ALL POSSIBLE CONFIGURATIONS OF DOUBLE WORD PAIRS (NON-INTER-
 1825                           ;* LEAVED, 64K INTERLEAVED, OR 128K INTERLEAVED) ARE CHECKED FOR EACH BANK
 1826                           ;* THROUGH USE OF TESTADD AND KERNEL INSTRUCTION PAGE ADDRESS REGISTERS 4 AND
 1827                           ;* 5. IF WE GET THE PROPER CHECKBITS BACK, WE HAVE A MATCH. IF NOT, THE ROUT-
 1828                           ;* INE CHECKS FOR SINGLE OR DOUBLE BIT ERRORS.
 1829                           ;*                 IF ONE OR THE OTHER IS FOUND, THE ERROR ADDRESS IS CHECKED
 1830                           ;* TO SEE IF IT IS THAT BANK.  IF IT IS, WE HAVE A MATCH.  AT THE END OF EACH
 1931                           ;* BANK PASS, FOR EACH CSR PASS, THE PROGRAM COMES UP WITH A NUMBER, STORED IN
 1832                           ;* "I", WHICH DENOTES THE FOLLOWING:
 1833                           ;*
 1834                           ;*      I       MEMORY DESCRIPTION
 1835                           ;*      -       ------------------
 1836                           ;*      0       NON-EXISTANT MEMORY
 1837                           ;*      1       NON-INTERLEAVED MEMORY
 1838                           ;*      2       64K INTERLEAVED, A1 NOT ASSERTED MEMORY
 1839                           ;*      3       128K INTERLEAVED, A1 NOT ASSERTED MEMORY
 1840                           ;*      4       64K INTERLEAVED, A1 ASSERTED MEMORY
 1841                           ;*      5       128K INTERLEAVED, A1 ASSERTED MEMORY
 1842                           ;*
 1843                           ;*  NOTE - I=2 THROUGH I=5 CAN ONLY OCCUR WITH MS11-M MEMORY.
 1844                           ;*
 1845                           ;* NOTE THAT PARITY MEMORY WRITES WRONG PARITY TO THE DOUBLE WORDS, THEN LOOKS
 1846                           ;* FOR THE PARITY ERROR BIT TO BE SET.  IF THE BIT IS SET, WE HAVE A MATCH.
 1847                           ;*
 1848 006046  005037  002274            CLR     CHECK
 1849 006052  000241                    CLC
 1850 006054  013703  002216            MOV     TOTCSRS,R3
 1851 006060  006303            70$:    ASL     R3
 1852 006062  103403                    BCS     71$
 1853 006064  005237  002274            INC     CHECK
 1854 006070  000773                    BR      70$
 1855 006072  006337  002274    71$:    ASL     CHECK
 1856 006076                            SET4    #6$                     ;NE MEMORY TRAPS GO TO 4
 1857 006104  005037  002502            CLR     PGMCSR                  ;SET PROGRAM CSR POINTER TO ZERO
 1858 006110  012701  002362            MOV     #TESTADD,R1             ;SET UP THE VIRTUAL ADDR. POINTER
 1859 006114  013703  002216            MOV     TOTCSRS,R3              ;MOVE CSR MAP INTO R3
 1860 006120  005000                    CLR     R0                      ;CLEAR THE CSR POINTER
 1861 006122  005005                    CLR     R5                      ;CLEAR THE PROGRAM CSR STATUS POINTER
 1862 006124  005737  002424            TST     NO22BIT                 ;IS THIS AN 11/44?
 1863 006130  001403                    BEQ     7$                      ;BRANCH IF IT IS
 1864 006132  005037  002530            CLR     LASTBLOCK               ;ADJUST LASTBLOCK INDICATOR FOR 124K MACHINE
 1865 006136  000413                    BR      1$                      ;BRANCH OVER NEXT PIECE OF CODE
 1866 006140  022737  000167  002526  7$:       CMP     #167,LASTBANK   ;IS THERE UNIBUS MEMORY ABOVE 17000000?
 1867 006146  001407                    BEQ     1$                      ;BRANCH IF NOT
 1868 006150  013702  002526            MOV     LASTBANK,R2             ;SET UP A NEW LAST BLOCK INDICATOR
 1869 006154  005202                    INC     R2
 1870 006156  072227  000011            ASH     #9.,R2
```

```
1871 006162 010237 002530            MOV     R2,LASTBLOCK
1872 006166 012702 000004      1$:   MOV     #4,R2              ;R2 IS INDEX FOR CONFIG TABLE
1873 006172 012737 001000 172350     MOV     #1000,KIPAR4       ;SET KIPAR4 FOR BANK 1
1874 006200 012737 001000 172352     MOV     #1000,KIPAR5       ;SET KIPAR5 FOR BANK 1
1875 006206 006303             2$:   ASL     R3                 ;DOES THIS CSR EXIST?
1876 006210 103420                   BCS     3$                 ;BRANCH IF IT DOES EXIST
1877 006212 062700 000002            ADD     #2,R0              ;INCREMENT THE CSR POINTER
1878 006216 010037 002146            MOV     R0,CSRNO           ;STORE IT IN CSRNO ALSO
1879 006222 005703                   TST     R3                 ;ARE THERE ANY MORE CSR'S TO DO?
1880 006224 001370                   BNE     2$                 ;BRANCH IF ALL CSRS NOT DONE
1881 006226 012737 001000 172350     MOV     #1000,KIPAR4       ;RESTORE KIPAR4
1882 006234 012737 001200 172352     MOV     #1200,KIPAR5       ;RESTORE KIPAR5
1883 006242 013706 002534            MOV     KSTACK,SP          ;RESTORE STACK
1884 006246 000137 007550            JMP     SUBAAS             ;JUMP TO SUBAAS IF ALL CSR'S ARE DONE
1885 006252 010037 002146      3$:   MOV     R0,CSRNO           ;MAKE SURE CSRNO IS UPDATED
1886 006256 022705 000003            CMP     #3,R5              ;PROGRAM CSR FOUND?
1887 006262 001413                   BEQ     13$                ;BRANCH IF TRUE
1888 006264 022705 000001            CMP     #1,R5              ;INTERLEAVED PROGRAM CSR HALF FOUND?
1889 006270 001403                   BEQ     14$                ;BRANCH IF TRUE
1890 006272 110037 002502            MOVB    R0,PGMCSR          ;SET UP PROGRAM CSR POINTER
1891 006276 000405                   BR      13$
1892 006300 110037 002503      14$:  MOVB    R0,PGMCSR+1        ;SET UP INTERLEAVED PROGRAM CSR POINTER
1893 006304 052737 100000 002502     BIS     #BIT15,PGMCSR      ;SET INTERLEAVED PROGRAM CSR FLAG
1894 006312 104424             13$:  CACHOFF                    ;TURN THE CACHE OFF
1895 006314 000240                   NOP
1896 006316 012737 100000 002362 45$:  MOV     #100000,TESTADD    ;SET UP VIRTUAL ADDRESS TO KIPAR4
1897 006324 012737 120002 002364     MOV     #120002,TESTADD+2  ;SET UP VIRTUAL ADDRESS TO KIPAR5
1898 006332 032762 000040 002624     BIT     #BIT5,CONFIG(R2)   ;IS THIS A BANK TO SKIP?
1899 006340 001402                   BEQ     43$                ;NO - BRANCH AROUND NEXT INSTRUCTION
1900 006342 000137 007474            JMP     6$                 ;YES - GO TO END OF BANK
1901 006346 005037 002422      43$:  CLR     I                  ;CLEAR THE MEMORY CONFIGURATION COUNTER
1902 006352 005771 000000      4$:   TST     @(R1)              ;TEST TO SEE THAT THERE IS MEMORY PRESENT
1903 006356 005237 002422            INC     I
1904 006362                          PUSH    @(R1),@2(R1)       ;SAVE THE LOCATIONS UNDER TEST
1905 006372 032760 000002 002432     BIT     #BIT1,CSRINFO(R0)  ;IS THIS PARITY MEMORY?
1906 006400 001414                   BEQ     34$                ;NO - BRANCH
1907 006402 052760 000004 172100     BIS     #BIT2,CSRADD(R0)   ;SET WRITE WRONG PARITY
1908 006410 012771 123456 000000     MOV     #123456,@(R1)      ;SET THE FIRST LOCATION UNDER TEST
1909 006416 012771 123456 000002     MOV     #123456,@2(R1)     ;SET THE SECOND LUT
1910 006424 005060 172100            CLR     CSRADD(R0)         ;CLEAR THE CSR
1911 006430 000411                   BR      41$                ;TEST LOCATIONS
1912 006432 012771 123456 000000 34$:  MOV     #123456,@(R1)      ;SET THE FIRST LOCATION UNDER TEST
1913 006440 012771 123456 000002     MOV     #123456,@2(R1)     ;SET THE SECOND LUT
1914 006446 104503                   CLR1CSR                    ;RESET CSR
1915 006450 104475                   CB1CSR                     ;SET DIAG. CHECK MODE IN CSR UNDER TEST
1916 006452 000240             40$:  NOP
1917 006454 005771 000000      41$:  TST     @(R1)              ;READ THE FIRST LUT TO WRITE CKBITS. INTO CSR
1918 006460 104426                   READCSR                    ;READ THE CSR UNDER TEST
1919 006462 000240                   NOP
1920 006464 013704 002144            MOV     CSR,R4             ;GET THE CHECKBITS FROM THE CSR
1921 006470 000240                   NOP
1922 006472 010437 002404            MOV     R4,TEMP            ;SAVE IN TEMP FOR LATER
1923 006476 104503                   CLR1CSR                    ;RESET CSR
1924 006500                          POP     @2(R1),@(R1)       ;RESTORE LOCATIONS UNDER TEST
1925 006510 032760 000002 002432     BIT     #BIT1,CSRINFO(R0)  ;IS THIS PARITY MEMORY?
1926 006516 001404                   BEQ     42$                ;NO - BRANCH
1927 006520 005704                   TST     R4                 ;DID WE GET A PARITY ERROR?
```

```
1928 006522  100414                        BMI     25$                     ;YES - FILL IN CONFIG TABLE
1929 006524  000137  007474                JMP     6$                      ;NO - JUMP TO END OF BANK
1930 006530  072427  177773        42$:    ASH     #-5,R4                  ;MANIPULATE THE CSR BITS
1931 006534  042704  177600              BIC     #^C177,R4               ;INTO A USABLE FORM.
1932 006540  000240                      NOP
1933 006542  022704  000157              CMP     #157,R4                 ;DO THE CHECKBITS COMPARE TO WHAT WAS WRITTEN?
1934 006546  001402                      BEQ     25$                     ;BRANCH IF THERE IS A MATCH
1935 006550  000137  007154              JMP     22$                     ;ELSE BRANCH IF NOT THE SAME
1936                                      ;*
1937                                      ;* WE COME HERE IF THERE IS A MATCH
1938                                      ;*
1939 006554  010004        25$:          MOV     R0,R4                   ;GET THE CSR NUMBER
1940 006556  006204                      ASR     R4                      ;SET IT UP FOR USE IN THE
1941 006560  000304                      SWAB    R4                      ;CONFIGURATION TABLE.
1942 006562  042704  170377              BIC     #170377,R4              ;CLEAR OFF EXTRANEOUS BITS
1943 006566  032737  000004  002422      BIT     #BIT2,I                 ;INTERLEAVED A1 ASSERTED MEMORY FOUND?
1944 006574  001402                      BEQ     15$                     ;BRANCH IF NOT
1945 006576  072427  000004              ASH     #4,R4                   ;PUT CSR NUMBER IN INTERLEAVED CSR SLOT
1946 006602  050462  002624        15$:  BIS     R4,CONFIG(R2)           ;PUT CSR NUMBER IN CONFIG. TABLE
1947 006606  016004  002432              MOV     CSRINFO(R0),R4          ;GET MEMORY TYPE
1948 006612  042704  177770              BIC     #^C7,R4                 ;CLEAR OFF THE EXTRANEOUS BITS
1949 006616  000304                      SWAB    R4                      ;MOVE INTO PROPER POSITION
1950 006620  050462  002626              BIS     R4,CONFIG+2(R2)         ;SET IT INTO THE CONFIG TABLE
1951 006624  022737  000001  002422      CMP     #1,I                    ;WAS THIS NON-INTERLEAVED MEMORY?
1952 006632  001431                      BEQ     24$                     ;BRANCH IF IT WAS
1953 006634  052762  010000  002626      BIS     #BIT12,CONFIG+2(R2)     ;SET THE INTERLEAVED BIT
1954 006642  010204                      MOV     R2,R4                   ;SAVE THE CURRENT BANK INDEX
1955 006644  032737  000001  002422      BIT     #BIT0,I                 ;WAS THIS 128K INTERLEAVED?
1956 006652  001006                      BNE     5$                      ;BRANCH IF TRUE
1957 006654  052762  004000  002626      BIS     #BIT11,CONFIG+2(R2)     ;SET 64K INTERLEAVED FLAG IN CONFIG
1958 006662  062704  000020              ADD     #20,R4                  ;SET NEW BANK POINTER TO 4 BANKS AHEAD
1959 006666  000402                      BR      16$                     ;JUMP OVER NEXT INSTRUCTION
1960 006670  062704  000040        5$:   ADD     #40,R4                  ;SET NEW BANK POINTER 8 BANKS AHEAD
1961 006674  052764  000040  002624  16$:  BIS     #BIT5,CONFIG(R4)        ;SET SKIP ECC LOGIC TESTS FLAG
1962 006702  056264  002624  002624        BIS     CONFIG(R2),CONFIG(R4)   ;SET OTHER INFO INTO THAT BANK
1963 006710  056264  002626  002626        BIS     CONFIG+2(R2),CONFIG+2(R4)
1964                                      ;*
1965                                      ;* THIS SECTION IS EXECUTED ONLY WHEN THE BANK 1
1966                                      ;*
1967 006716  022737  001000  172350  24$:  CMP     #1000,KIPAR4            ;IS THIS BANK 1?
1968 006724  001402                      BEQ     30$                     ;BRANCH IF TRUE
1969 006726  000137  007334              JMP     33$                     ;ELSE JUMP TO END OF THIS BANK
1970 006732  032737  100020  002404  30$:  BIT     #BIT15!BIT4,TEMP        ;WAS THERE A SBE OR DBE?
1971 006740  001417                      BEQ     44$                     ;BRANCH IF NOT
1972 006742  013704  002404              MOV     TEMP,R4                 ;GET CSR CONTENTS
1973 006746  072427  177767              ASH     #-9.,R4                 ;MAKE ERROR ADDRESS INTO BANK #
1974 006752  022704  000001              CMP     #1,R4                   ;ERROR IN BANKS 0 OR 1?
1975 006756  003010                      BGT     44$                     ;BRANCH IF NOT
1976 006760  052762  000001  002624      BIS     #BIT0,CONFIG(R2)        ;SET ERROR FLAG IN CONFIG TABLE
1977 006766  105262  002626              INCB    CONFIG+2(R2)            ;ADD ONE TO BANK ERROR COUNT
1978 006772                              SET     CONFGERROR              ;PRINT CONFIG TABLE
1979 007000  022737  000001  002422  44$:  CMP     #1,I                    ;WAS THIS NON-INTERLEAVED MEMORY?
1980 007006  001003                      BNE     9$                      ;BRANCH IF NOT
1981 007010  012705  000003              MOV     #3,R5                   ;SET PGM CSR FLAG TO DONE
1982 007014  000420                      BR      10$                     ;BRANCH OVER NEXT STUFF
1983 007016  032737  000004  002422  9$:   BIT     #BIT2,I                 ;A1 ASSERTED INTERLEAVED MEMORY?
1984 007024  001010                      BNE     11$                     ;BRANCH IF TRUE
```

```
1985 007026  110037  002503                    MOVB    R0,PGMCSR+1             ;SET PGM CSR IN HI BYTE
1986 007032  052737  100000  002502            BIS     #BIT15,PGMCSR           ;SET INTERLEAVED PGM CSR FLAG
1987 007040  052705  000002                    BIS     #2,R5                   ;SET PGM CSR FLAG TO HI BYTE DONE
1988 007044  000404                            BR      10$                     ;BRANCH OVER NEXT STUFF
1989 007046  110037  002502            11$:    MOVB    R0,PGMCSR               ;SET PGM CSR TO LO BYTE
1990 007052  052705  000001                    BIS     #1,R5                   ;SET INTERLEAVE PGM CSR FLAG
1991 007056  053737  002630  002624    10$:    BIS     CONFIG+4,CONFIG         ;SET UP INFORMATION IN BANK ZERO
1992 007064  053737  002632  002626            BIS     CONFIG+6,CONFIG+2
1993 007072  000240                            NOP
1994 007074  022737  000001  002422            CMP     #1,I                    ;WAS THIS NON-INTERLEAVED MEMORY
1995 007102  001002                            BNE     46$                     ;NO - BRANCH OVER NEXT STMT.
1996 007104  000137  007474                    JMP     6$                      ;YES - JUMP TO END OF THIS BANK
1997 007110  012704  000020            46$:    MOV     #20,R4                  ;SET UP COUNTER FOR 64K INTERLEAVED
1998 007114  032737  000001  002422            BIT     #BIT0,I                 ;WAS IT 128K INTERLEAVED?
1999 007122  001402                            BEQ     26$                     ;BRANCH IF NOT
2000 007124  062704  000020                    ADD     #20,R4                  ;SET UP COUNTER FOR 128K INTERLEAVED
2001 007130  053764  002624  002624    26$:    BIS     CONFIG,CONFIG(R4)       ;SET OTHER BANK WITH SAME INFORMATION
2002 007136  053764  002626  002626            BIS     CONFIG+2,CONFIG+2(R4)   ;AS IN BANK 0
2003 007144  052764  000040  002624            BIS     #BIT5,CONFIG(R4)        ;SET SKIP ECC LOGIC TESTS FLAG
2004 007152  000470                            BR      33$                     ;BRANCH
2005                                           ;*
2006                                           ;* IF CHECKBITS DID NOT MATCH, WE COME HERE
2007                                           ;*
2008 007154  032737  100020  002144    22$:    BIT     #BIT15!BIT4,CSR         ;SBE OR DBE FLAGS SET?
2009 007162  001001                            BNE     8$                      ;BRANCH IF TRUE
2010 007164  000463                            BR      33$                     ;CHECK TO SEE IF IT IS MS11-M
2011 007166  013704  002146            8$:     MOV     CSRNO,R4                ;GET CSRNO
2012 007172  042764  000006  172100            BIC     #6,CSRADD(R4)           ;TURN OFF DIAG CHECK & ECC DISABLE
2013 007200                                    PUSH    R0,R1                   ;SAVE R0 & R1
2014 007204  016401  172100                    MOV     CSRADD(R4),R1           ;GET CSR INFORMATION
2015 007210  072127  177773                    ASH     #-5,R1                  ;SET UP ERROR ADDRESS
2016 007214  042701  177600                    BIC     #^C177,R1
2017 007220  005737  002424                    TST     NO22BIT                 ;IS THIS AN 11/44
2018 007224  001015                            BNE     27$                     ;BRANCH IF NOT
2019 007226  052764  040000  172100            BIS     #BIT14,CSRADD(R4)       ;GET EXTENDED ERROR ADDRESS BITS
2020 007234  016400  172100                    MOV     CSRADD(R4),R0           ;READ FROM CSR
2021 007240  042764  040000  172100            BIC     #BIT14,CSRADD(R4)       ;TURN OFF FUB BIT
2022 007246  042700  177037                    BIC     #^C740,R0               ;SET UP EXTENDED BITS
2023 007252  006300                            ASL     R0
2024 007254  006300                            ASL     R0
2025 007256  060001                            ADD     R0,R1                   ;SET UP TOTAL ERROR ADDRESS
2026 007260  010104                    27$:    MOV     R1,R4                   ;SAVE IN R4
2027 007262                                    POP     R1,R0                   ;RESTORE R0 & R1
2028 007266  072427  000005                    ASH     #5,R4                   ;SET ERROR ADDRESS UP IN PAR NOTATION
2029 007272  020437  172350                    CMP     R4,KIPAR4               ;DOES IT EQUAL KIPAR4?
2030 007276  001001                            BNE     28$                     ;BRANCH IF FALSE
2031 007300  000403                            BR      35$                     ;YES - MARK INFO IN CONFIG TABLE
2032 007302  020437  172352            28$:    CMP     R4,KIPAR5               ;DOES IT EQUAL KIPAR5?
2033 007306  001012                            BNE     33$                     ;BRANCH IF FALSE
2034 007310  052762  000001  002624    35$:    BIS     #BIT0,CONFIG(R2)        ;SET BANK ERROR FLAG
2035 007316  105262  002626                    INCB    CONFIG+2(R2)            ;INCREMENT BANK ERROR COUNTER
2036 007322                                    SET     CONFGERROR              ;PRINT CONFIG TABLE
2037 007330  000137  006554                    JMP     25$                     ;YES - MARK INFO IN CONFIG TABLE
2038                                           ;*
2039                                           ;* THIS SECTION SETS UP ALL THE POSSIBLE CONFIGURATIONS OF
2040                                           ;* MS11-M MEMORY.
2041                                           ;*
```

```
2042 007334  032760  000003  002432  33$:    BIT     #BIT0.BIT1,CSRINFO(R0)  ;IS THIS MS11-M MEMORY?
2043 007342  001054                          BNE     6$                      ;NO - GO TO END OF BANK
2044 007344  032760  000004  002432          BIT     #BIT2,CSRINFO(R0)
2045 007352  001450                          BEQ     6$
2046 007354  022737  000001  002422          CMP     #1,I                    ;IS THIS 1ST TIME THROUGH?
2047 007362  103410                          BLO     18$                     ;BRANCH IF NOT
2048 007364  162737  000002  002364          SUB     #2,TESTADD+2            ;TRY AS 64K INTERLEAVED
2049 007372  062737  004000  172352          ADD     #4000,KIPAR5            ;A1 NON-ASSERTED MEMORY
2050 007400  000137  006352                  JMP     4$                      ;TRY TO MATCH AGAIN
2051 007404  022737  000004  002422  18$:    CMP     #4,I                    ;4TH TIME THROUGH?
2052 007412  001404                          BEQ     20$                     ;YES - BRANCH
2053 007414  022737  000002  002422          CMP     #2,I                    ;2ND TIME THROUGH
2054 007422  103405                          BLO     12$                     ;NO - BRANCH
2055 007424  062737  004000  172352  20$:    ADD     #4000,KIPAR5            ;TRY AS 128K INTERLEAVED
2056 007432  000137  006352                  JMP     4$                      ;TRY TO MATCH AGAIN
2057 007436  022737  000003  002422  12$:    CMP     #3,I                    ;THIRD TIME THROUGH?
2058 007444  103413                          BLO     6$                      ;NO - BRANCH
2059 007446  062737  000002  002362          ADD     #2,TESTADD              ;TRY TESTING THE BANK
2060 007454  062737  000002  002364          ADD     #2,TESTADD+2            ;AS A1 ASSERTED
2061 007462  162737  004000  172352          SUB     #4000,KIPAR5            ;64K INTERLEAVED MEMORY
2062 007470  000137  006352                  JMP     4$                      ;TRY TO MATCH AGAIN
2063                                         ;*
2064                                         ;*END OF BANK ROUTINE
2065                                         ;*
2066 007474  104503                  6$:     CLR1CSR                         ;CLEAR THE CSR UNDER TEST
2067 007476  062702  000004                  ADD     #4,R2                   ;UPDATE CONFIGURATION POINTER
2068 007502  062737  001000  172350          ADD     #1000,KIPAR4            ;UPDATE KIPAR4 TO NEXT BANK
2069 007510  013737  172350  172352          MOV     KIPAR4,KIPAR5           ;AND UPDATE KIPAR5
2070 007516  000240                          NOP
2071 007520  023737  002530  172350          CMP     LASTBLOCK,KIPAR4        ;HAVE WE DONE THE WHOLE MEMORY SPACE?
2072 007526  002002                          BGE     19$                     ;BRANCH IF DONE
2073 007530  000137  006316                  JMP     45$                     ;JUMP IF NOT DONE
2074 007534  062700  000002          19$:    ADD     #2,R0                   ;INCREMENT CSR POINTER
2075 007540  000240                          NOP
2076 007542  104423                          CACHON                          ;TURN ON THE CACHE
2077 007544  000137  006166                  JMP     1$                      ;JUMP TO TRY NEXT CSR
```

```
                                SUBAAS: CACHON          ;MAKE SURE THE CACHE IS ON
                                        ECCINIT         ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
                                        NEWTST  <<TEST BANK 0 ACCESSES>>
                                ;*************************************************************************************
                                ;*TEST 2         TEST BANK 0 ACCESSES
                                ;*************************************************************************************
    007554  000004              TST2:   SCOPE
                                        ;THIS DOES A 'TST'' INSTRUCTION ON EVERY LOCATION IN BANK #0 TO SEE
                                        ;IF IT GETS ANY PARITY TRAPS.
                                        ;SINCE EVERY LOCATION IS EITHER LOADED OR WRITTEN INTO BY THE PROGRAM
                                        ;PRIOR TO THIS POINT - THEN A PARITY ERROR IMPLIES THAT THERE IS A
                                        ;HARDWARE FAILURE IN THE MEMORY.
                                        ;THESE ERRORS ARE COUNTED AND A FATAL ACTION IS TAKEN
007556  005037  002070                  CLR     PARCNT                  ;CLEAR PARITY ERROR COUNTER
007562  012737  000001  002074          MOV     #1,NOPAR                ;SET THE NO PARITY ERROR FLAG
007570  005037  002066                  CLR     NEMCNT                  ;CLEAR NON-EXISTANT MEMORY ERROR COUNTER
007574  012737  000001  002076          MOV     #1,NONEM                ;SET THE NON-EXISTANT MEMORY ERROR MODE TO COUNT
007602                                  SET4    #NONEXIST               ;TRAPS TO 4 GOTO NONEXIST
007610  005000                          CLR     R0
007612  012701  040000                  MOV     #SIZE,R1
007616  104424                          CACHOFF                         ;TURN CACHE OFF
007620  005720                  1$:     TST     (R0)+                   ;SEE IF I CAN DO A READ ACCESS WITHOUT A PARITY TRAP
007622  077102                          SOB     R1,1$
007624  104423                          CACHON                          ;TURN CACHE ON
                                        ;SEE IF ANY FAILURES
007626  005737  002070                  TST     PARCNT                  ;ANY PARITY ERRORS?
007632  001403                          BEQ     2$                      ;NO - SKIP
007634                                  FATAL   3
007642  005737  002066          2$:     TST     NEMCNT                  ;ANY NON-EXISTANT MEMORY (HOLES)?
007646  001406                          BEQ     3$                      ;SKIP IF EQUAL
007650  162737  000002  002032          SUB     #2,ADDRESS              ;UPDATE 1ST ADDRESS FAILURE FROM AUTO INCREMENT #
007656                                  FATAL   4
007664  053737  002104  002624  3$:     BIS     CPUBIT,CONFIG           ;SET CORRECT ACCESSED BIT ON BANK 0
007672                                  RES4                            ;RESET TRAPS TO 4 TO DEFAULT

007712                                  SUBTST  <<ENABLE ECC FOR CORRECT TRAPS>>
                                ;*************************************************************************************
                                ;*SUBTEST        ENABLE ECC FOR CORRECT TRAPS
                                ;*************************************************************************************
007712                                  IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE
007730  104506                            ENASBE                        ;TRAP ON SINGLE BIT ERRORS
007732                                  ELSE
007734  104472                            ECCINIT                       ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
007736                                  END ;OF IF #SW0
```

```
 2118 007736                                    NEWTST  <<TEST BANKS 1-200 (OCTAL) FOR ZEROS & ONES>>
                                        ;*****************************************************************************
                                        ;*TEST 3        TEST BANKS 1-200 (OCTAL) FOR ZEROS & ONES
                                        ;*****************************************************************************
      007736  000004                    TST3:   SCOPE
 2119                                           ;EACH BANK IS TESTED FOR EXISTANCE AND IF IT EXISTS
 2120                                           ;THEN IS IS TESTED FOR ZEROS & ONES.
 2121                                           ;EXCEPT -
 2122                                           ;       PROTECTED BANKS (WHERE THE PROGRAM IS) ARE ONLY TESTED BY
 2123                                           ;       "TST" INSTRUCTIONS LIKE BANK #0
 2124                                           ;
 2125                                           ;ANY BAD BANKS ARE LOGGED IN THE CONFIGURATION TABLE.
 2126 007740  005037  002100                    ;THIS ROUTINE IS ONLY DOING A SMART SIZE - NOT ACTUAL TESTING.
                                                CLR     BANK
 2127 007744  012737  000001  002074            MOV     #1,NOPAR                ;SET NO PARITY ERROR FLAG
 2128 007752  012737  000002  002076            MOV     #2,NONEM                ;SET NON-EXISTANT MEMORY MODE TO EXIT TEST LOOP
 2129 007760                                    SET4    #NONEXIST       ;TRAPS TO 4 GOTO NONEXIST
 2130 007766  005737  002424                    TST     NO22BIT         ;IS THIS AN 11/44?
 2131 007772  001407                            BEQ     1$              ;BRANCH IF TRUE
 2132 007774  012737  010564  002472            MOV     #MTST3+4,LINK1  ;SET UP LINKS
 2133 010002  012737  010566  002474            MOV     #MTST3+6,LINK2
 2134 010010  000411                            BR      TAG9$
 2135 010012                            1$:     BMOV    MTST3                   ;PUT IN FAST MEMORY
 2136 010020  012737  177644  002472            MOV     #UIPAR2,LINK1   ;SET UP LINKS
 2137 010026  012737  177643  002474            MOV     #UIPAR3,LINK2
 2138 010034  005237  002100            TAG9$:  INC     BANK
 2139 010040  023737  002526  002100            CMP     LASTBANK,BANK           ;DONE?
 2140 010046  103456                            BLO     TAG2$                   ;YES - SKIP TO NEXT TEST
 2141 010050  013701  002100                    MOV     BANK,R1
 2142 010054  006301                            ASL     R1
 2143 010056  006301                            ASL     R1              ;BANK * 4
 2144 010060  010137  002102                    MOV     R1,BANKINDEX
 2145 010064  005037  002072                    CLR     PATERR                  ;CLEAR PATTERN ERROR COUNTER
 2146 010070  005037  002070                    CLR     PARCNT                  ;CLEAR PARITY ERROR COUNTER
 2147 010074  005037  002066                    CLR     NEMCNT                  ;CLEAR NON-EXISTANT MEMORY COUNTER (HOLES)
 2148 010100                                    MAP     BANK                    ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
 2149 010114  105761  002624                    TSTB    CONFIG(R1)              ;IS THIS BANK PROTECTED?
 2150 010120  100552                            BMI     TSTBANK                 ;YES - GO TEST BANK SPECIAL
 2151 010122  012777  000207  172342    WARN1:  MOV     #207,@LINK1             ;PUT "RETURN" INSTRUCTION AFTER WRITE ROUTINE
 2152 010130  012700  040000                    MOV     #FIRST,R0
 2153 010134  010004                            MOV     R0,R4
 2154 010136  012701  040000                    MOV     #SIZE,R1
 2155 010142  010103                            MOV     R1,R3
 2156 010144  005002                            CLR     R2                      ;DATA IS ZEROS
 2157 010146  104424                            CACHOFF                 ;TURN CACHE OFF
 2158 010150                                    TESTAREA                        ;ENTER SUPERVISOR MODE
 2159 010156  005737  002424                    TST     NO22BIT         ;IS THIS AN 11/44?
 2160 010162  001403                            BEQ     1$              ;BRANCH IF TRUE
 2161 010164  004737  010560                    CALL    MTST3
 2162 010170  000402                            BR      2$
 2163 010172  004737  177640            1$:     CALL    FASTCITY                ;CALL TO THE USER INSTRUCTION PAR'S
 2164 010176  104417                    2$:     KERNEL                  ;ENTER KERNEL MODE
 2165 010200  104423                            CACHON                  ;TURN CACHE ON
 2166 010202  000415                            BR      TAG3$                   ;SKIP NEXT INSTRUCTION
 2167 010204  005037  002100            TAG2$:  CLR     BANK
 2168 010210                                    RES4                    ;RESET TRAPS TO 4 TO DEFAULT
 2169 010230  005037  002074                    CLR     NOPAR                   ;INDICATE DEFAULT PARITY ACTION
 2170 010234  000563                            BR      SUBAAI
```

```
2171 010236 005737 002066          TAG3$: TST   NEMCNT                  ;ANY TRAPS?
2172 010242 001401                        BEQ   1$                       ;NO - SKIP
2173 010244 000673                        BR    TAG9$                    ;NOW - TRY NEXT BANK
2174 010246 104424                  1$:   CACHOFF                 ;TURN CACHE OFF
2175 010250                               TESTAREA                       ;ENTER SUPERVISOR MODE
2176 010256 004777 172212                 CALL  @LINK2                   ;FINISH PATTERN
2177 010262 104417                        KERNEL          ;ENTER KERNEL MODE
2178 010264 104423                        CACHON                         ;TURN CACHE ON
2179 010266 005737 002072                 TST   PATERR                   ;ANY PATTERN ERRORS
2180 010272 001037                        BNE   2$                       ;YES - SKIP
2181 010274 005737 002070                 TST   PARCNT                   ;ANY PARITY ERRORS
2182 010300 001034                        BNE   2$                       ;YES - SKIP
2183 010302 005737 002066                 TST   NEMCNT                   ;ANY NON EXISTANT MEMORY
2184 010306 001031                        BNE   2$                       ;YES - SKIP
2185 010310 012700 060000                 MOV   #FIRST,R0
2186 010314 010004                        MOV   R0,R4
2187 010316 012701 040000                 MOV   #SIZE,R1
2188 010322 010103                        MOV   R1,R3
2189 010324 013702 002554                 MOV   ONES,R2                  ;DATA IS ONES
2190 010330 012777 000240 172134          MOV   #000240,@LINK1           ;PUT 'NOP' INSTRUCTION BACK IN SUBROUTINE
2191 010336 104424                        CACHOFF               ;TURN CACHE OFF
2192 010340                               TESTAREA                       ;ENTER TEST MODE
2193 010346 005737 002424                 TST   NO22BIT                  ;IS THIS AN 11/44?
2194 010352 001403                        BEQ   5$                       ;BRANCH IF IT IS
2195 010354 004737 010560                 CALL  MTST3                    ;DO IN MEMORY IF NOT
2196 010360 000402                        BR    6$                       ;JUMP OVER NEXT INSTRUCTION
2197 010362 004737 177640           5$:   CALL  FASTCITY                 ;CALL TO THE USER INSTRUCTION PAR'S
2198 010366 104417                  6$:   KERNEL          ;ENTER KERNEL MODE
2199 010370 104423                        CACHON                         ;TURN CACHE ON
2200 010372 013700 002102           2$:   MOV   BANKINDEX,R0
2201 010376 005737 002072                 TST   PATERR                   ;ANY PATTERN ERRORS?
2202 010402 001006                        BNE   3$                       ;YES - SKIP
2203 010404 005737 002070                 TST   PARCNT                   ;ANY PARITY ERRORS?
2204 010410 001003                        BNE   3$                       ;YES - SKIP
2205 010412 005737 002066                 TST   NEMCNT                   ;ANY HOLES?
2206 010416 001406                        BEQ   4$                       ;NONE - SKIP
2207 010420 052760 000001 002624    3$:   BIS   #BIT0,CONFIG(R0)         ;SET ERROR BIT IN THIS BANK
2208 010426                               SET   CONFGERROR               ;FORCE PRINTING OF CONFIGURATION TABLE
2209 010434 053760 002104 002624    4$:   BIS   CPUBIT,CONFIG(R0)        ;SET ACCESSED BIT
2210 010442 000137 010034                 JMP   TAG9$
2211
2212                                      ;TEST A PROTECTED BANK
2213 010446                         TSTBANK:PUSH  R1
2214 010450 012737 000001 002076          MOV   #1,NONEM                 ;SET NON-EXISTANT MEMORY TO COUNT
2215 010456 012700 060000                 MOV   #FIRST,R0
2216 010462 012701 020000                 MOV   #20000,R1
2217 010466 104424                        CACHOFF                        ;TURN CACHE OFF
2218 010470                               TESTAREA                       ;ENTER TEST MODE
2219 010476 005720                  4$:   TST   (R0)+
2220 010500 077102                        SOB   R1,4$
2221 010502 104417                        KERNEL                         ;ENTER KERNEL MODE
2222 010504 104423                        CACHON                         ;TURN CACHE ON
2223 010506 012737 000002 002076          MOV   #2,NONEM                 ;RESET NON-EXISTANT MEMORY TO EXIT TEST LOOP
2224 010514                               POP   R1
2225 010516                               IF PARCNT NE #0
2226 010524 052761 000001 002624            BIS   #BIT0,CONFIG(R1)       ;ERROR BANK
2227 010532                                 SET   CONFGERROR
```

```
2228 010540                                        END ;OF IF PARCNT
2229 010540                                        IF NEMCNT EQ #0
2230 010546  053761  002104  002624                  BIS   CPUBIT,CONFIG(R1)      ;ACCESSED BANK
2231 010554                                        END ;OF IF NEMCNT
2232 010554  000137  010034                        JMP   TAG9$
2233 010560  010220                    MTST3:      MOV   R2,(R0)+               ;V177640
2234 010562  077102                                SOB   R1,MTST3               ;V177642
2235 010564  000240                                NOP                          ;V177644
2236 010566  012401                    2$:         MOV   (R4)+,R1               ;V177646
2237 010570  020102                                CMP   R1,R2                  ;V177650
2238 010572  001402                                BEQ   3$                     ;V177652
2239 010574  005237  002072                        INC   PATERR                 ;V177654
2240 010600  077306                    3$:         SOB   R3,2$                  ;V177660
2241 010602  000207                                RETURN                       ;V177662
```

```
                                SUBAAI: SUBTST  <<FIND SHADOW INHIBIT MODE POINTERS>>
                                ;*************************************************************************
                                ;*SUBTEST       FIND SHADOW INHIBIT MODE POINTERS
                                ;*************************************************************************
                                ;* THIS SECTION LOOKS FOR INTERLEAVED MS11-M MEMORIES AND FIGURES OUT
                                ;* WHERE THE SHADOW INHIBIT MODE POINTERS ARE LOCATED.  THESE AREAS
                                ;* ARE THEN MARKED AS PROGRAM SPACE.
010604  005037  002100          SHADL1: CLR     BANK                    ;RESET BANK TO ZERO
010610  004737  042700                  CALL    EXBANK                  ;SET BANK PARAMETERS
010614  013700  002102                  MOV     BANKINDEX,R0
010620                                  IF ACFLAG IS TRUE AND INTFLAG IS TRUE
010634                                    IF INT64K IS TRUE
010642  062700  000020                      ADD #20,R0                  ;POINT TO BANKINDEX + 4
010646  062737  000010  002100              ADD #10,BANK                ;POINT TO BANK + 8
010654                                    ELSE
010656  062702  000040                      ADD #40,R2                  ;POINT TO BANKINDEX + 8
010662  062737  000020  002100              ADD #20,BANK                ;POINT TO BANK + 16
010670                                    END; OF IF INT64K
010670  052760  000200  002624            BIS   #BIT7,CONFIG(R0)        ;MAKE NEW BANK PROGRAM SPACE
010676                                  ELSE
010700  005237  002100                    INC   BANK                    ;GO TO NEXT BANK
010704                                  END; OF IF ACFLAG
010704  023737  002526  002100          CMP     LASTBANK,BANK           ;HAVE WE DONE ALL THE BANKS?
010712  002336                          BGE     SHADL1                  ;BRANCH IF NOT
```

```
 2266 010714                                    NEWTST  <<ECC INHIBIT MODE POINTER TEST>>
                                        ;*****************************************************************************
                                        ;*TEST 4        ECC INHIBIT MODE POINTER TEST
                                        ;*****************************************************************************
      010714  000004                    TST4:   SCOPE
 2267                                           ;THE MS11-M OR MF11S-K INHIBIT ECC DISABLE AND DIAGNOSTIC CHECK MODE
 2268                                           ;ON THE BOTTOM  FIRST OR SECOND 16K WORDS CONTROLLED BY A CSR.  THIS
 2269                                           ;IS CONSIDERED  TO BE A  PROTECTED BANK  BY THE  PROGRAM.  IT MAY BE
 2270                                           ;QUITE  COMPLEX TO DETERMINE ON A  GIVEN SYSTEM  CONFIGURATION WHICH
 2271                                           ;BANKS CAN BE PROTECTED;
 2272                                           ;SO
 2273                                           ;THIS ROUTINE ATTEMPS TO  CREATE A DOUBLE BIT ERROR IN ADDRESS 0 & 2
 2274                                           ;OF  EVERY ECC BANK.  ECC HARDWARE WILL PREVENT THIS  FROM HAPPENING
 2275                                           ;IN  PROTECTED BANKS  WHICH SHOULD ALWAYS  INCLUDE BANK ZERO - WHERE
 2276                                           ;THE PROGRAM IS.
 2277                                           ;
 2278                                           ;
 2279                                           ;WARNING:!!.!!.!!!!
 2280                                           ;  IN CASE OF HARDWARE FAILURE IT IS COMMON  THAT A DOUBLE BIT ERROR
 2281                                           ;  WILL BE  CREATED ON  THE KERNEL  STACK & ''CRASH'' THE  DIAGNOSTIC
 2282                                           ;DURING  THIS ROUTINE.  YOUR ONLY CLUE IS THAT YOU CAN GET AS FAR AS
 2283                                           ;THIS ROUTINE BUT NOT PAST IT!
 2284 010716  104424                            CACHOFF                         ;TURN CACHE OFF
 2285 010720  012737  177777  002150            MOV     #-1,OLDCSR
 2286 010726                                    FOR BANK := #0 TO LASTBANK
 2287 010732  012701  060000                      MOV   #FIRST,R1               ;SET UP VIRT ADDR POINTER
 2288 010736  004737  042700                      CALL  EXBANK
 2289 010742  013700  002102                      MOV   BANKINDEX,R0
 2290 010746                                      IF ACFLAG IS TRUE
 2291 010754                                        IF MKFLAG IS TRUE
 2292 010762                                        IF SKIPMK IS FALSE
 2293 010770                                          IF INTFLAG IS TRUE
 2294 010776  012703  040000                            MOV     #40000,R3      ;SET INDEX COUNTER
 2295 011002  012737  000001  002232                    MOV     #1,SPLTCSR     ;MAP AS INTERLEAVED BANK
 2296 011010                                          ELSE
 2297 011012  012703  000002                            MOV     #2,R3          ;SET INDEX COUNTER
 2298 011016                                          END; OF IF INTFLAG
 2299 011016  116002  002625                          MOVB    CONFIG+1(R0),R2
 2300 011022  006302                                  ASL     R2
 2301 011024  042702  177741                          BIC     #^C36,R2
 2302 011030  010237  002146                          MOV     R2,CSRNO
 2303 011034                                          IF CSRNO NE OLDCSR
 2304 011044  013737  002146  002150                    MOV     CSRNO,OLDCSR
 2305 011052                                            IF PFLAG IS FALSE
 2306 011060  052760  000100  002624                      BIS  #BIT6,CONFIG(R0)
 2307 011066                                            END; OF IF PFLAG
 2308 011066  004737  011222                            CALL    IMPTEST
```

```
2310 011072                                          IF INTFLAG IS TRUE
2311 011100  116002  002625                            MOVB  CONFIG+1(R0),R2
2312 011104  072227  177775                            ASH   #-3,R2
2313 011110  042702  177741                            BIC   #^C36,R2
2314 011114  010237  002146                            MOV   R2,CSRNO
2315 011120  062701  000002                            ADD   #2,R1            ;FIX POINTER FOR A1 ASSERTED HALF
2316 011124  004737  011222                            CALL  IMPTEST
2317 011130  005037  002232                            CLR   SPLTCSR
2318 011134                                          END; OF IF INTFLAG
2319 011134                                        END; OF IF CSRNO
2320 011134                                      END; OF IF SKIPMK
2321 011134                                      END; OF IF MKFLAG
2322 011134                                    END; OF IF ACFLAG
2323 011134                                  END; OF FOR BANK
2324 011150                                  MAP                       ;MAP TEST SPACE TO BANK 0
2325 011164  005037  002100                  CLR     BANK
2326 011170                                  IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE
2327 011206  104506                            ENASBE                  ;TRAP ON SINGLE BIT ERRORS
2328 011210                                  ELSE
2329 011212  104472                            ECCINIT                 ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
2330 011214                                  END; OF IF #SW0
2331 011214  104423                          CACHON                    ;TURN THE CACHE BACK ON
2332 011216  000137  011464                  JMP     SUBAAR            ;JUMP OVER THE SUBROUTINE
```

```
2334 011222  005004                    IMPTEST:CLR     R4
2335 011224                                    MAP BANK                        ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
2336 011240  005005                            CLR     R5
2337 011242  012737  020000  002144            MOV     #BIT13,CSR
2338 011250                                    TESTAREA                        ;ENTER TEST MODE
2339 011256                                    PUSH    (R1)                    ;SAVE TEST LOCATION
2340 011260  060301                            ADD     R3,R1                 ;INDEX TO NEXT LOCATION
2341 011262                                    PUSH    (R1)                  ;SAVE TEST LOCATION
2342 011264  104505                            CHK1DIS                       ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
2343 011266  010411                            MOV     R4,(R1)               ;WRITE CHECKBITS (ALL ZEROS)
2344 011270  160301                            SUB     R3,R1
2345 011272  010411                            MOV     R4,(R1)
2346 011274  104503                            CLR1CSR                         ;CLEAR CSR
2347 011276  005711                            TST     (R1)                    ;READ CHECKBITS INTO REAL CSR
2348 011300  104501                            WAS1DBE                         ;WAS THERE A DOUBLE BIT ERROR
2349
2350                                           ;THIS MAKES SURE THAT SBE'S DON'T LOOK LIKE PROTECTED AREAS
2351
2352 011302                                    ON.NOERROR ;1
2353 011304  012737  020000  002144              MOV   #BIT13,CSR
2354 011312  104505                              CHK1DIS                       ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
2355 011314  013711  002554                      MOV   ONES,(R1)
2356 011320  060301                              ADD   R3,R1
2357 011322  013711  002554                      MOV   ONES,(R1)
2358 011326  160301                              SUB   R3,R1
2359 011330  104503                              CLR1CSR                       ;CLEAR CSR
2360 011332  005711                              TST   (R1)
2361 011334  104501                              WAS1DBE                       ;WAS THERE A DOUBLE BIT ERROR
2362 011336                                      ON.NOERROR ;2
2363 011340  012737  027400  002144                MOV #27400,CSR
2364 011346  104505                                CHK1DIS                     ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
2365 011350  010411                                MOV R4,(R1)
2366 011352  060301                                ADD R3,R1                 ;ADD INDEX TO GET TO SECOND WORD
2367 011354  010411                                MOV R4,(R1)
2368 011356  160301                                SUB R3,R1                 ;SUBTRACT INDEX TO FIRST WORD
2369 011360  104503                                CLR1CSR                         ;CLEAR CSR
2370 011362  005711                                TST (R1)
2371 011364  104501                                WAS1DBE                         ;WAS THERE A DOUBLE BIT ERROR
2372 011366                                        ON.NOERROR ;3
2373 011370  012737  074000  002144                  MOV       #74000,CSR
2374 011376  104505                                  CHK1DIS                   ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
2375 011400  010411                                  MOV       R4,(R1)
2376 011402  060301                                  ADD       R3,R1           ;INDEX TO SECOND WORD
2377 011404  010411                                  MOV       R4,(R1)
2378 011406  104503                                  CLR1CSR                   ;CLEAR CSR
2379 011410  160301                                  SUB       R3,R1           ;GO BACK TO FIRST WORD
2380 011412  005711                                  TST       (R1)
2381 011414  104501                                  WAS1DBE                   ;WAS THERE A DOUBLE BIT ERROR
2382 011416                                        END ;OF ON.NOERROR ;3
2383 011416                                    END ;OF ON.NOERROR ;2
2384 011416                                END ;OF ON.NOERROR ;1
2385 011416                                ON.ERROR
2386 011420  005205                              INC   R5                        ;IDENTIFY AS BAD BANK
2387 011422                                END ;OF ON.ERROR
2388 011422  104471                        ECC1DIS                             ;DISABLE ERROR CORRECTION
2389 011424  010411                        MOV     R4,(R1)                   ;CLEAR OUT DOUBLE BIT ERROR.
2390 011426  060301                        ADD     R3,R1                     ;INDEX TO SECOND WORD
```

```
2391 011430  010411                        MOV     R4,(R1)             ;CLEAR OUT DOUBLE BIT ERROR!
2392 011432  104503                        CLR1CSR
2393 011434  005705                        TST     R5
2394 011436  001405                        BEQ     1$
2395 011440  050560  002624                BIS     R5,CONFIG(R0)
2396 011444  105260  002626                INCB    CONFIG+2(R0)
2397 011450  104036                       ERROR   +36
2398 011452                          1$:   POP     (R1)                ;RESTORE TEST LOCATION (2ND WORD)
2399 011454  160301                        SUB     R3,R1               ;GO BACK TO FIRST WORD
2400 011456                                POP     (R1)                ;RESTORE TEST LOCATION (1ST WORD)
2401 011460  104417                        KERNEL
2402 011462  000207                        RETURN
```

```
                                                                    M 2
   2746
   2747 011464                          SUBAAR: SET     STOPOK          ;PROGRAM CAN NOW BE HALTED
```

```
                                                SUBTST  <<LEGAL CONFIGURATION CHECK>>
                                        ;*****************************************************************************
                                        ;*SUBTEST        LEGAL CONFIGURATION CHECK
                                        ;*****************************************************************************
2751 011474  012700  000020             MOV     #16.,R0
2752 011476  012701  002432             MOV     #CSRINFO,R1
2753 011502  005021                     1$:     CLR     (R1)+
2754 011504  077002                             SOB     R0,1$
2758 011506                                       FOR BANK := #0 TO LASTBANK
2759 011512  00473,  042700                         CALL        EXBANK
2760 011516  013700  002102                       MOV BANKINDEX,R0
2782
2783 011522                                       IF ACFLAG IS TRUE
2784 011530  116003  002625                         MOVB        CONFIG+1(R0),R3
2785 011534  042703  177760                         BIC         #^C17,R3
2786 011540  006303                               ASL         R3
2787 011542  005263  002432                       INC         CSRINFO(R3)
2788 011546                                       IF MKFLAG IS TRUE
2789                                              ;MAKE SURE THAT EACH BANK HAS NO MORE THAN 2 CSRS
2790 011554                                         BEGIN   LEGALCSR
2791 011554                                           IF INTFLAG IS TRUE
2792 011562  116003  002625                             MOVB        CONFIG+1(R0),R3
2793 011566  010304                                   MOV R3,R4
2794 011570  042703  177760                           BIC #^C17,R3
2795 011574  072427  177774                           ASH #-4,R4
2796 011600  042704  177760                           BIC #^C17,R4
2797 011604                                           IF R3 EQ R4
2798 011610  042760  014000  002626                     BIC         #BIT11!BIT12,CONFIG+2(R0)
2799 011616  042760  170000  002624                     BIC         #170000,CONFIG(R0)
2800 011624                                             LEAVE LEGALCSR
2801 011626                                           END; OF IF R3
2802 011626                                           IF KFLAG IS FALSE
2803 011634                                             LEAVE     LEGALCSR
2804 011636                                           END; OF IF KFLAG
2805 011636  006303                                   ASL R3
2806 011640  006304                                   ASL R4
2807 011642  005263  002432                           INC CSRINFO(R3)
2808 011646  005264  002432                           INC CSRINFO(R4)
2809 011652                                         ELSE
2810 011654                                           LEAVE       LEGALCSR
2811 011656                                         END; OF IF INTFLAG
2812 011656                                       TYPE    MSG124                    ;# OF CSR'S IS WRONG
2813 011662                                       TYPOCS  BANK,<TYPE BANK #>,3
2814 011672                                       SET     CONFGERROR
2815 011700                                       END     LEGALCSR
2816 011700                                     END ;OF IF MKFLAG
2817 011700                                   END ;OF IF ACFLAG
2818 011700                                 END; OF FOR BANK
2819 011714                               PUSH    R5,R0                             ;SAVE CONTENTS OF R5, R0
2820 011720  005000                       CLR     R0                                ;CLEAR REGISTERS
2821 011722  005001                       CLR     R1
2822 011724  005005                       CLR     R5
2823 011726  005037  002274               CLR     CHECK                             ;CLEAR ERROR INDICATOR
2824 011732  022761  000010  002432  2$:  CMP     #10,CSRINFO(R1)                   ;IS CURRENT CSR <= 10
2825 011740  002043                       BGE     5$                                ;BRANCH IF SO
2826 011742  022761  000020  002432       CMP     #20,CSRINFO(R1)                   ;IS CURRENT CSR < 20
2827 011750  002003                       BGE     3$                                ;BRANCH IF SO
```

```
2828 011752  004737  012102                    CALL    ILLCSR                          ;CALL ERROR ROUTINE
2829 011756  000434                            BR      5$                              ;TRY NEXT CSR
2830 011760  016005  002624            3$:     MOV     CONFIG(R0),R5                   ;MOVE LOW WORD TO R5
2831 011764  032705  000002                    BIT     #BIT1,R5                        ;DOES MEMORY EXIST HERE?
2832 011770  001415                            BEQ     4$                              ;BRANCH IF NOT
2833 011772  042705  170377                    BIC     #^C7400,R5                      ;ISOLATE CSR NUMBER IN
2834 011776  072527  177771                    ASH     #-7,R5                          ;REGISTER 5
2835 012002  020501                            CMP     R5,R1                           ;IS IT THE CURRENT CSR?
2836 012004  001007                            BNE     4$                              ;TRY NEXT WORD OF CONFIG IF NOT
2837 012006  032760  010000  002626            BIT     #BIT12,CONFIG+2(R0)             ;IS IT INTERLEAVED?
2838 012014  001003                            BNE     4$                              ;BRANCH IF SO
2839 012016  012737  000001  002274            MOV     #1,CHECK                        ;SET ERROR INDICATOR
2840 012024  062700  000004            4$:     ADD     #4,R0                           ;UPDATE CONFIG COUNTER
2841 012030  022700  000340                    CMP     #340,R0                         ;CONFIG TABLE ALL DONE?
2842 012034  001351                            BNE     3$                              ;BRANCH IF NOT
2843 012036  005737  002274                    TST     CHECK                           ;ERRORS FOUND?
2844 012042  001402                            BEQ     5$                              ;TRY NEXT CSR IF NOT
2845 012044  004737  012102                    CALL    ILLCSR                          ;CALL ERROR ROUTINE
2846 012050  005000            5$:     CLR     R0                              ;REINITIALIZE CONFIG COUNTER
2847 012052  005037  002274                    CLR     CHECK                           ;CLEAR ERROR INDICATOR
2848 012056  062701  000002                    ADD     #2,R1                           ;UPDATE CSR COUNTER
2849 012062  022701  000040                    CMP     #40,R1                          ;ALL CSR'S DONE?
2850 012066  001321                            BNE     2$                              ;BRANCH IF NOT
2851 012070                                    POP     R0,R5                           ;RESTORE REGISTERS
2852 012074  005037  002274                    CLR     CHECK                           ;RESET CHECK
2856 012100  000421                            BR      SUBAAP          ;BRANCH TO NEXT SUBTEST
2857 012102  010102            ILLCSR: MOV     R1,R2           ;R2 HAS CSR NUMBER
2858 012104  006202                            ASR     R2              ;MAKE ACCEPTABLE FOR PRINTING
2859 012106  022702  000012                    CMP     #10.,R2
2860 012112  100002                            BPL     1$
2861 012114  062702  000007                    ADD     #7,R2
2862 012120  062702  000060            1$:     ADD     #60,R2
2863 012124  110237  077470                    MOVB    R2,MSGA122      ;PUT NUMBER INTO ERROR MESSAGE
2864 012130                                    TYPE    MSG122
2865 012134                                    SET     CONFGERROR
2866 012142  000207                            RETURN
```

```
2869 012144                          SUBAAP: SUBTST  <<PRINT CONFIGURATION DETAILS>>                    ;
                                     ;*****************************************************************************
                                     ;*SUBTEST       PRINT CONFIGURATION DETAILS
                                     ;*****************************************************************************
2870 012144                                  CLEAR   BSIZE,KSIZE,LSIZE,MSIZE,PSIZE
2871 012170  013702  002526                  MOV     LASTBANK,R2
2872 012174  006302                          ASL     R2
2873 012176  006302                          ASL     R2
2874 012200                                  FOR R1 := #0 TO R2 BY #4
2875 012202                                   IF CPUBIT SET.IN CONFIG(R1)
2876 012212                                     IF #BIT10 SET.IN CONFIG+2(R1)
2877 012222                                       IF #BIT8 SET.IN CONFIG+2(R1)
2878 012232                                         IF #BIT9 SET.IN CONFIG+2(R1)
2879 012242                                           LET PSIZE := PSIZE + #1
2880 012246                                         ELSE
2881 012250                                           LET KSIZE := KSIZE + #1
2882 012254                                         END;IF BIT9
2883 012254                                       ELSE
2884 012256                                         IF #BIT9 SET.IN CONFIG+2(R1)
2885 012266                                           LET LSIZE := LSIZE + #1
2886 012272                                         ELSE
2887 012274                                           LET MSIZE := MSIZE + #1
2888 012300                                         END; IF BIT9
2889 012300                                       END;IF BIT8
2890 012300                                     ELSE
2891 012302                                       IF #BIT9 SET.IN CONFIG+2(R1)
2892 012312                                         IF #BIT8 SET.IN CONFIG+2(R1)
2893 012322                                           LET BSIZE := BSIZE + #1
2894 012326                                         END; OF IF #BIT8
2895 012326                                       END; OF IF #BIT9
2896 012326                                     END;IF BIT10
2897 012326                                   END; OF IF CPUBIT
2898 012326                                  END ;OF FOR ALL BANKS IN TABLE
2899
2900 012336  005037  002422                  CLR     I
2901 012342                                  FOR R1 := #0 TO #10 BY #2
2902 012344  006361  002344                    ASL   BSIZE(R1)
2903 012350  006361  002344                    ASL   BSIZE(R1)
2904 012354  006361  002344                    ASL   BSIZE(R1)
2905 012360  006361  002344                    ASL   BSIZE(R1)                ;BSIZE(R1) := BSIZE(R1) * 16.
2906 012364  066137  002344  002422            ADD   BSIZE(R1),I              ;I <- I + BSIZE(R1)
2907 012372                                  END; FOR R1
2908 012404                                  FOR R1 := #0 TO #200 BY #4
2909 012406                                     IF CPUBIT SET.IN CONFIG(R1)
2910 012416                                       LET UNITOP := UNITOP + #1
2911 012422                                     END; OF IF CPUBIT
2912 012422                                  END; OF FOR R1
2913 012434  006337  002366                  ASL     UNITOP
2914 012440  006337  002366                  ASL     UNITOP
2915 012444  006337  002366                  ASL     UNITOP
2916 012450  006337  002366                  ASL     UNITOP                 ;UNITOP := UNITOP * 16.
2917 012454                                  IF I LT UNITOP THEN LET I := UNITOP
2918 012472                                  TYPE    $CRLF
2919 012476  005737  002344                  TST     BSIZE
2920 012502  001405                          BEQ     1$
2921 012504                                  TYPDEC  BSIZE
2922 012512                                  TYPE    MSG071
```

```
 2923 012516  005737  002346          1$:     TST     KSIZE
 2924 012522  001405                          BEQ     2$
 2925 012524                                   TYPDEC  KSIZE
 2926 012532                                   TYPE    MSG072
 2927 012536  005737  002350          2$:     TST     LSIZE
 2928 012542  001405                          BEQ     3$
 2929 012544                                   TYPDEC  LSIZE
 2930 012552                                   TYPE    MSG112
 2931 012556  005737  002352          3$:     TST     MSIZE
 2932 012562  001405                          BEQ     4$
 2933 012564                                   TYPDEC  MSIZE
 2934 012572                                   TYPE    MSG113
 2935 012576  005737  002354          4$:     TST     PSIZE
 2936 012602  001405                          BEQ     5$
 2937 012604                                   TYPDEC  PSIZE
 2938 012612                                   TYPE    MSG114
 2939 012616                          5$:     TYPDEC  I
 2940 012624                                   TYPE    MSG070
 2941 012630                                   IF #SW6 OFF.IN @SWR
 2942 012640  004737  035254                      CALL  PCONFIG
 2943 012644                                   END; OF IF #SW6
```

```
2946 012644                                    SUBTST  <<CHECK APT SIZING>>
                                       ;*****************************************************************************
                                       ;*SUBTEST       CHECK APT SIZING
                                       ;*************************************************** *****************************
2947 012644                                    IF APTFLAG IS TRUE AND APTSIZE IS TRUE
2948 012660  005037  002404                       CLR    TEMP
2949 012664  012700  061306                       MOV    #$MAMS1,R0
2950 012670                                       FOR R2 := #0 TO #4
2951 012672                                         IFB 1(R0) NE #0
2952 012700  111001                                   MOVB       (R0),R1
2953 012702  042701  177400                           BIC        #177400,R1
2954 012706                                           IF 2(R0) LT #0
2955 012714  000261                                     SEC
2956 012716                                           ELSE
2957 012720  000241                                     CLC
2958 012722                                           END ;OF IF 2(R0)
2959 012722  006101                                   ROL        R1
2960 012724  005201                                   INC        R1              ;TO COMPENSATE FOR 4 BANKS BEING (0-3)
2961 012726  006301                                   ASL        R1
2962 012730  006301                                   ASL        R1
2963 012732  006301                                   ASL        R1
2964 012734  006301                                   ASL        R1
2965 012736  163701  002404                           SUB        TEMP,R1
2966 012742  010137  002404                           MOV        R1,TEMP
2967 012746                                           IFB 1(R0) EQ #3
2968 012756  060137  002372                             ADD      R1,APTPAR
2969 012762                                           END ;OF IFB 1(R0)
2970 012762                                           IFB 1(R0) EQ #4
2971 012772  060137  002374                             ADD      R1,APTECC
2972 012776                                           END ;OF IFB 1(R0)
2973 012776  062700  000004                           ADD        #4,R0
2974 013002                                         END ;OF IFB 1(R0)
2975 013002                                       END ;OF FOR R2
2976 013012                                       IF APTPAR NE LSIZE OR APTECC NE MSIZE
2977 013032  104046                                 ERROR        +46
2978 013034                                       END ;OF IF APTPAR
2990 013034                                    END ;OF IF APTFLAG
```

```
                                LOOP:   NEWTST  <<DIAGNOSTIC MODE DISPATCH ROUTINE>>
                                ;*******************************************************************************
                                ;*TEST 5         DIAGNOSTIC MODE DISPATCH ROUTINE
                                ;*******************************************************************************
     013034  000004             TST5:   SCOPE
2993 013036  005037  002214             CLR     CONTFLAG
2994 013042  017700  167530             MOV     @SWR,R0         ;GET SWITCHES
2995 013046  042700  177761             BIC     #^C16,R0        ;MASK TO ONLY MODE BITS
2996 013052  004770  013062             CALL    @DISPTBL(R0)    ;DISPATCH TO ROUTINE THROUGH NEXT TABLE
2997 013056  000137  013102             JMP     MEMDONE         ;GO TO NEXT TEST
2998 013062  013546             DISPTBL:BAFPAF  ;MODE 0;BANKS FORWARD, PATTERNS FORWARD
2999 013064  013654                     BAFPAR  ;MODE 1;BANKS FORWARD, PATTERNS REVERSE
3000 013066  013762                     BAWPAF  ;MODE 2;BANKS WORST FIRST, PATTERNS FORWARD
3001 013070  014112                     BAWPAR  ;MODE 3;BANKS WORST FIRST, PATTERNS REVERSE
3002 013072  014242                     PAFBAF  ;MODE 4;PATTERNS FORWARD, BANKS FORWARD
3003 013074  014372                     PAFBAW  ;MODE 5;PATTERNS FORWARD, BANKS WORST FIRST
3004 013076  014544                     PARBAF  ;MODE 6;PATTERNS REVERSE, BANKS FORWARD
3005 013100  014674                     PARBAW  ;MODE 7;PATTERNS REVERSE, BANKS WORST FIRST
3006
3007 013102  004737  013446     MEMDONE:CALL    DOBACK                          ;CHECK BACKGROUND PATTERN
3008 013106                             IF SELONLY IS TRUE
3009 013114  005137  002560               COM   SOFTPAT                         ;COMPLIMENT BACKGROUND PATTERN
3010 013120                             END ;OF IF SELONLY
3011
3012 013120                             NEWTST<<UNIQUE BANK TEST>>
                                ;*******************************************************************************
                                ;*TEST 6         UNIQUE BANK TEST
                                ;*******************************************************************************
     013120  000004             TST6:   SCOPE
3013                            ;MAKE SURE THAT EACH BANK CAN HAVE UNIQUE DATA
3014                            ;WRITE AND READ THE BANK NUMBER IN EACH BANK (EXCEPT WHERE THE PROGRAM IS)
3015 013122                             IF SELONLY IS FALSE
3016 013130                               SET   HEADER,MUT
3017 013144  004737  022740               CALL  MT0027
3018 013150                               SET   HEADER
3019 013156  005037  002106               CLR   MUT
3020 013162                             END ;OF IF SELONLY
3021 013162  004737  013446             CALL    DOBACK          ;RESTORE BACKROUND PATTERN
3025
3026 013166                     FLUSH:  SUBTST  <<FLUSH OUT DBE'S>>
                                ;*******************************************************************************
                                ;*SUBTEST        FLUSH OUT DBE'S
                                ;*******************************************************************************
3027 013166  004737  023412             CALL    MT0030
```

```
                                        .SBTTL  END OF PASS ROUTINE
                                ;*******************************************************************************
                                ;*INCREMENT THE PASS NUMBER ('$PASS)
                                ;*INDICATE END-OF-PROGRAM AFTER EACH PASSES THRU THE PROGRAM
                                ;*TYPE 'END PASS #XXXXX'' (WHERE XXXXX IS A DECIMAL NUMBER)
                                ;*IF THERES A MONITOR GO TO IT
                                ;*IF THERE ISN'T JUMP TO LOOP
013172  005037  002412          $EOP:   CLR     FSINFLAG
013176  012700  002626                  MOV     #CONFIG+2,R0    ;MOVE 2ND WORD OF CONFIG TO R0
013202  042710  020000          1$:     BIC     #BIT13,(R0)     ;CLEAR BACKGROUND VALID BIT
013206  062700  000004                  ADD     #4,R0           ;INCREMENT TO NEXT BANK
013212  020027  003620                  CMP     R0,#3620        ;DONE?
013216  003771                          BLE     1$              ;NO - BRANCH
013220  013737  002570  002014          MOV     $ERTTL,LASTERROR
013226  005237  061264                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
013232  042737  100000  061264          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
013240                                  TYPE    MSG077          ;;TYPE 'END PASS #''
013244                                  IF #SW11 SET.IN @SWR OR QVFLAG IS TRUE
013262                                    TYPE  MSG035          ;QV
013266  005037  002316                    CLR   QVFLAG
013272                                  END ;OF IF SW11
013272                                  TYPDEC  $PASS
013300  013700  000042                  MOV     42,R0           ;;GET MONITOR ADDRESS
013304  001456                          BEQ     $DOAGAIN        ;;BRANCH IF NO MONITOR
013306  022700  002000          $ZAP42: CMP     #STACK,R0       ;ARE WE UNDER RT11
013312  001453                          BEQ     $DOAGAIN        ;YES - BRANCH
                                        ;WE ARE UNDER (HEAVEN HELP US) XXDP!
013314                                  PUSH    R0
013316  004737  043566                  CALL    SHUTUP
013322                                  POP     R0
013324  000005                          RESET                   ;;CLEAR THE WORLD
013326  004710                  $ENDAD: CALL    (R0)            ;;GO TO MONITOR
013330  000240                          NOP                     ;;SAVE ROOM
013332  000240                          NOP                     ;;FOR
013334  000240                          NOP                     ;;ACT11
013336                          $DOAGN: ;UNDO SHUTUP STUFF
                                        ;       RESTORE STACK
                                        ;       ENERGIZE UNIBUS MAP & 22 BIT ADDRESSING
                                        ;       ENERGIZE MEMORY MANAGEMENT
                                        ;       PUT LOADERS BACK HOME
013336  013706  002534                  MOV     KSTACK,SP
013342  005737  002424                  TST     NO22BIT
013346  001003                          BNE     1$
013350  052737  000060  172516          BIS     #BIT5.BIT4,MMR3
013356  104420                  1$:     ENERGIZE                ;TURN ON MEMORY MANAGEMENT
013360  013700  002536                  MOV     LOADHOME,R0             ;DESTINATION BANK
013364  012701  000001                  MOV     #1,R1                   ;SOURCE BANK
013370  004737  042350                  CALL    BANKMOV
013374                                  IF APTFLAG IS TRUE
013402                                    IF $USWR EQ $PASS
013412  012701  000050          APTHANG:    MOV #50,R1
013416  077001                  2$:         SOB R0,2$
013420  062737  000001  061266            ADD #1,$DEVCT
013426  005537  061270                    ADC $UNIT
013432  077107                            SOB R1,2$
013434  005237  061264                    INC $PASS
013440  000764                            BR  APTHANG
```

```
 3087 013442                                    END ;OF IF $USWR
 3088 013442                                  END ;OF IF APTFLAG
 3089 013442  000137  013034          $DOAGAIN: JMP   LOOP              ;RETURN
```

```
 3092 013446                        DOBACK: SUBTST  <<WRITE BACKGROUND PATTERNS>>
                                    ;*****************************************************************************
                                    ;*SUBTEST       WRITE BACKGROUND PATTERNS
                                    ;*****************************************************************************
 3093 013446  005037  002110               CLR     PATTERN
 3094 013452                               FOR BANK := #0 TO LASTBANK
 3095 013456  004737  042700                 CALL  EXBANK
 3096 013462                                 IF ACFLAG IS TRUE AND RRFLAG IS FALSE
 3097 013476                                   SET         HEADER,MUT
 3098 013512  004737  016354                   CALL        MKTEST          ;CALL   MJTEST WOULD ALSO WORK
 3099 013516  005037  002106                   CLR         MUT
 3100 013522                                   SET         HEADER
 3101 013530                                 END ;OF IF ACFLAG
 3102 013530                               END ;OF FOR BANK
 3103 013544  000207                       RETURN
```

```
3106                                    .SBTTL  MTEST MODES
3107
3108 013546                        BAFPAF: SUBTST  <<BANKS FORWARD,PATTERNS FORWARD      **RECURSIVE**>>
                                    ;*********************************************************************************
                                    ;*SUBTEST       BANKS FORWARD,PATTERNS FORWARD      **RECURSIVE**
                                    ;*********************************************************************************
3109 013546  005037  002100                CLR     BANK                ;SET BANK TO 0
3110                                        ;START OF BANK LOOP
3111 013552  004737  042700         1$:    CALL    EXBANK              ;EXAMINE BANK
3112 013556  005737  002114                TST     ACFLAG              ;CAN WE ACCESS THIS BANK?
3113 013562  001412                        BEQ     4$                  ;NO - GO TO BANK LOOP TERMINATION
3114 013564  005737  002122                TST     RRFLAG              ;RELOCATION REQUIRED?
3115 013570  001007                        BNE     4$                  ;YES - GO TO BANK LOOP TERMINATION
3116 013572  005037  002110                CLR     PATTERN             ;SET PATTERN TO 0
3117                                        ;START OF PATTERN LOOP
3118 013576  004737  015046         2$:    CALL    MTEST               ;GO TEST CORRECT MEMORY
3119                                        ;TERMINATION OF PATTERN LOOP
3120 013602  004737  043366                CALL    INCPAT              ;GO SEE IF THIS IS THE LAST PATTERN
3121 013606  001373                        BNE     2$                  ;NO - LOOP ON THIS PATTERN
3122                                        ;TERMINATION OF BANK LOOP
3123 013610  005037  002214         4$:    CLR     CONTFLAG
3124 013614  004737  043412                CALL    INCBNK              ;NEXT HIGHER BANK
3125 013620  002354                        BGE     1$                  ;IF NOT DONE - LOOP ON THIS BANK
3126                                        ;END OF LOOPS
3127 013622  005737  002124                TST     RLFLAG              ;HAVE WE BEEN RELOCATED?
3128 013626  001401                        BEQ     5$                  ;NO - SKIP
3129 013630  000207                        RETURN                      ;YES - RETURN
3130 013632  004737  041154         5$:    CALL    RELOCATE            ;MOVE & MAP PROGRAM
3131 013636                                ON.ERROR THEN $RETURN
3132                                        ;**NOTE** RECURSIVE CALL
3133 013642  004737  013546                CALL    BAFPAF              ;CALL SELF
3134 013646  004737  042042                CALL    UNRELOCATE          ;UNMOVE & UNMAP PROGRAM
3135 013652  000207                        RETURN
```

```
3138 013654                          BAFPAR: SUBTST  <<BANKS FORWARD,PATTERNS REVERSE      **RECURSIVE**>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       BANKS FORWARD,PATTERNS REVERSE      **RECURSIVE**
                                     ;*****************************************************************************
3139 013654  005037  002100                     CLR     BANK            ;SET BANK TO 0
3140                                             ;START OF BANK LOOP
3141 013660  004737  042700          1$:        CALL    EXBANK          ;EXAMINE BANK
3142 013664  005737  002114                     TST     ACFLAG          ;CAN WE ACCESS THIS BANK?
3143 013670  001412                             BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3144 013672  005737  002122                     TST     RRFLAG          ;RELOCATION REQUIRED?
3145 013676  001007                             BNE     4$              ;YES - GO TO BANK LOOP TERMINATION
3146 013700  004737  043402                     CALL    SETPAT          ;SET HIGH PATTERN FOR CORRECT MEMORY
3147                                             ;START OF PATTERN LOOP
3148 013704  004737  015046          2$:        CALL    MTEST           ;GO TEST CORRECT MEMORY
3149                                             ;TERMINATION OF PATTERN LOOP
3150 013710  005337  002110                     DEC     PATTERN         ;IS THIS THE LAST PATTERN?
3151 013714  100373                             BPL     2$              ;NO - LOOP ON THIS PATTERN
3152                                             ;TERMINATION OF BANK LOOP
3153 013716  005037  002214          4$:        CLR     CONTFLAG
3154 013722  004737  043412                     CALL    INCBNK          ;NEXT HIGHER BANK
3155 013726  002354                             BGE     1$              ;IF NOT DONE - LOOP ON THIS BANK
3156                                             ;END OF LOOPS
3157 013730  005737  002124                     TST     RLFLAG          ;HAVE WE BE N RELOCATED?
3158 013734  001401                             BEQ     5$              ;NO - SKIP
3159 013736  000207                             RETURN                  ;YES - RETURN
3160 013740  004737  041154          5$:        CALL    RELOCATE        ;MOVE & MAP PROGRAM
3161 013744                                      ON.ERROR THEN $RETURN
3162                                             ;**NOTE** RECURSIVE CALL
3163 013750  004737  013654                     CALL    BAFPAR          ;CALL SELF
3164 013754  004737  042042                     CALL    UNRELOCATE      ;UNMOVE & UNMAP PROGRAM
3165 013760  000207                             RETURN
```

```
3168 013762                         BAWPAF: SUBTST  <<BANKS WORST FIRST,PATTERNS FORWARD  **RECURSIVE**>>
                                    ;*****************************************************************************
                                    ;*SUBTEST       BANKS WORST FIRST,PATTERNS FORWARD  **RECURSIVE**
                                    ;*****************************************************************************
3169 013762  005037  002100                CLR     BANK            ;SET BANK TO 0
3170                                       ;START OF BANK LOOP
3171 013766  004737  042700         1$:    CALL    EXBANK          ;EXAMINE BANK
3172 013772  005737  002114                TST     ACFLAG          ;CAN WE ACCESS THIS BANK?
3173 013776  001415                        BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3174 014000  005737  002126                TST     BMFLAG          ;IS THIS BAD MEMORY (WORST FIRST)?
3175 014004  001412                        BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3176 014006  005737  002122                TST     RRFLAG          ;RELOCATION REQUIRED?
3177 014012  001007                        BNE     4$              ;YES - GO TO BANK LOOP TERMINATION
3178 014014  005037  002110                CLR     PATTERN         ;SET PATTERN TO 0
3179                                       ;START OF PATTERN LOOP
3180 014020  004737  015046         2$:    CALL    MTEST           ;GO TEST CORRECT MEMORY
3181                                       ;TERMINATION OF PATTERN LOOP
3182 014024  004737  043366                CALL    INCPAT          ;GO SEE IF THIS IS THE LAST PATTERN
3183 014030  001373                        BNE     2$              ;NO - LOOP ON THIS PATTERN
3184                                       ;TERMINATION OF BANK LOOP
3185 014032  005037  002214         4$:    CLR     CONTFLAG
3186 014036  004737  043412                CALL    INCBNK          ;NEXT HIGHER BANK
3187 014042  002351                        BGE     1$              ;IF NOT DONE - LOOP ON THIS BANK
3188                                       ;END OF LOOPS
3189 014044  005137  002540                COM     WORST           ;IS THIS AN EVEN NUMBERED PASS?
3190 014050  001003                        BNE     5$              ;YES - SKIP
3191                                       ;**NOTE** RECURSIVE CALL
3192 014052  004737  013762                CALL    BAWPAF          ;CALL SELF
3193 014056  000207                        RETURN
3194 014060  005737  002124         5$:    TST     RLFLAG          ;HAVE WE BEEN RELOCATED?
3195 014064  001401                        BEQ     6$              ;NO - SKIP
3196 014066  000207                        RETURN                  ;YES - RETURN
3197 014070  004737  041154         6$:    CALL    RELOCATE        ;MOVE & MAP PROGRAM
3198 014074                                ON.ERROR THEN $RETURN
3199                                       ;**NOTE** RECURSIVE CALL
3200 014100  004737  013762                CALL    BAWPAF          ;CALL SELF
3201 014104  004737  042042                CALL    UNRELOCATE      ;UNMOVE & UNMAP PROGRAM
3202 014110  000207                        RETURN
```

```
3205 014112                          BAWPAR: SUBTST  <<BANKS WORST FIRST,PATTERNS REVERSE  **RECURSIVE**>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       BANKS WORST FIRST,PATTERNS REVERSE  **RECURSIVE**
                                     ;*****************************************************************************
3206 014112  005037  002100                  CLR     BANK            ;SET BANK TO 0
3207                                         ;START OF BANK LOOP
3208 014116  004737  042700          1$:     CALL    EXBANK          ;EXAMINE BANK
3209 014122  005737  002114                  TST     ACFLAG          ;CAN WE ACCESS THIS BANK?
3210 014126  001415                          BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3211 014130  005737  002126                  TST     BMFLAG          ;IS THIS BAD MEMORY (WORST FIRST)
3212 014134  001412                          BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3213 014136  005737  002122                  TST     RRFLAG          ;RELOCATION REQUIRED?
3214 014142  001007                          BNE     4$              ;YES - GO TO BANK LOOP TERMINATION
3215 014144  004737  043402                  CALL    SETPAT          ;SET HIGH PATTERN FOR CORRECT MEMORY
3216                                         ;START OF PATTERN LOOP
3217 014150  004737  015046          2$:     CALL    MTEST           ;GO TEST CORRECT MEMORY
3218                                         ;TERMINATION OF PATTERN LOOP
3219 014154  005337  002110                  DEC     PATTERN         ;IS THIS THE LAST PATTERN?
3220 014160  100373                          BPL     2$              ;NO - LOOP ON THIS PATTERN
3221                                         ;TERMINATION OF BANK LOOP
3222 014162  005037  002214          4$:     CLR     CONTFLAG
3223 014166  004737  043412                  CALL    INCBNK          ;NEXT HIGHER BANK
3224 014172  002351                          BGE     1$              ;IF NOT DONE - LOOP ON THIS BANK
3225                                         ;END OF LOOPS
3226 014174  005137  002540                  COM     WORST           ;IS THIS AN EVEN NUMBERED PASS?
3227 014200  001003                          BNE     5$              ;YES - SKIP
3228                                         ;**NOTE** RECURSIVE CALL
3229 014202  004737  014112                  CALL    BAWPAR          ;CALL SELF
3230 014206  000207                          RETURN
3231 014210  005737  002124          5$:     TST     RLFLAG          ;HAVE WE BEEN RELOCATED?
3232 014214  001401                          BEQ     6$              ;NO - SKIP
3233 014216  000207                          RETURN                  ;YES - RETURN
3234 014220  004737  041154          6$:     CALL    RELOCATE        ;MOVE & MAP PROGRAM
3235 014224                                  ON.ERROR THEN $RETURN
3236                                         ;**NOTE** RECURSIVE CALL
3237 014230  004737  014112                  CALL    BAWPAR          ;CALL SELF
3238 014234  004737  042042                  CALL    UNRELOCATE      ;UNMOVE & UNMAP PROGRAM
3239 014240  000207                          RETURN
```

```
3242 014242                          PAFBAF: SUBTST  <<PATTERNS FORWARD,BANKS FORWARD      **RECURSIVE**>>
                                     ;*************************************************************************
                                     ;*SUBTEST       PATTERNS FORWARD,BANKS FORWARD      **RECURSIVE**
                                     ;*************************************************************************
3243 014242  005037  002110                 CLR     PATTERN          ;SET PATTERN TO 0
3244                                        ;START OF PATTERN LOOP
3245 014246  005037  002100          1$:    CLR     BANK             ;SET BANK TO 0
3246                                        ;START OF BANK LOOP
3247 014252  004737  042700          2$:    CALL    EXBANK           ;EXAMINE BANK
3248 014256  004737  043350                 CALL    BANKOK           ;CORRECT MEMORY FOR THIS BANK?
3249 014262  001010                         BNE     4$               ;NO - GO TO BANK LOOP TERMINATOR
3250 014264  005737  002114                 TST     ACFLAG           ;CAN WE ACCESS THIS BANK?
3251 014270  001405                         BEQ     4$               ;NO - GO TO BANK LOOP TERMINATION
3252 014272  005737  002122                 TST     RRFLAG           ;RELOCATION REQUIRED?
3253 014276  001002                         BNE     4$               ;YES - GO TO BANK LOOP TERMINATION
3254 014300  004737  015046                 CALL    MTEST            ;GO TEST CORRECT MEMORY
3255                                        ;TERMINATION OF BANK LOOP
3256 014304  005037  002214          4$:    CLR     CONTFLAG
3257 014310  004737  043412                 CALL    INCBNK           ;NEXT HIGHER BANK
3258 014314  002356                         BGE     2$               ;IF NOT DONE - LOOP ON THIS BANK
3259                                        ;TERMINATION OF PATTERN LOOP
3260 014316  004737  043366                 CALL    INCRPT           ;NEXT HIGHER PATTERN
3261 014322  001351                         BNE     1$               ;OK - LOOP; ELSE CONTINUE
3262                                        ;END OF LOOPS
3263 014324  005137  002132                 COM     TMFLAG           ;COMPLEMENT TYPE OF MEMORY
3264                                                                 ;IS THIS AN EVEN NUMBER PASS?
3265 014330  001403                         BEQ     5$               ;YES - SKIP
3266                                        ;**NOTE** RECURSIVE CALL
3267 014332  004737  014242                 CALL    PAFBAF           ;CALL SELF
3268 014336  000207                         RETURN
3269 014340  005737  002124          5$:    TST     RLFLAG           ;HAVE WE BEEN RELOCATED?
3270 014344  001401                         BEQ     6$               ;NO - SKIP
3271 014346  000207                         RETURN                   ;YES - RETURN
3272 014350  004737  041154          6$:    CALL    RELOCATE         ;MOVE & MAP PROGRAM
3273 014354                                 ON.ERROR THEN $RETURN
3274                                        ;**NOTE** RECURSIVE CALL
3275 014360  004737  014242                 CALL    PAFBAF           ;CALL SELF
3276 014364  004737  042042                 CALL    UNRELOCATE       ;UNMOVE & UNMAP PROGRAM
3277 014370  000207                         RETURN
```

```
3280 014372                     PAFBAW: SUBTST  <<PATTERNS FORWARD,BANKS WORST FIRST  **RECURSIVE**>>
                                ;*********************************************************************
                                ;*SUBTEST       PATTERNS FORWARD,BANKS WORST FIRST  **RECURSIVE**
                                ;*********************************************************************
3281 014372  005037  002110            CLR     PATTERN         ;SET PATTERN TO 0
3282                            ;START OF PATTERN LOOP
3283 014376  005037  002100    1$:     CLR     BANK            ;SET BANK TO 0
3284                            ;START OF BANK LOOP
3285 014402  004737  042700    2$:     CALL    EXBANK          ;EXAMINE BANK
3286 014406  004737  043350            CALL    BANKOK          ;CORRECT MEMORY FOR THIS BANK?
3287 014412  001013                    BNE     4$              ;NO - GO TO BANK LOOP TERMINATOR
3288 014414  005737  002114            TST     ACFLAG          ;CAN WE ACCESS THIS BANK?
3289 014420  001410                    BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3290 014422  005737  002126            TST     BMFLAG          ;IS THIS BAD MEMORY (WORST FIRST)
3291 014426  001405                    BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3292 014430  005737  002122            TST     RRFLAG          ;RELOCATION REQUIRED?
3293 014434  001002                    BNE     4$              ;YES - GO TO BANK LOOP TERMINATION
3294 014436  004737  015046            CALL    MTEST           ;GO TEST CORRECT MEMORY
3295                            ;TERMINATION OF BANK LOOP
3296 014442  005037  002214    4$:     CLR     CONTFLAG
3297 014446  004737  043412            CALL    INCBNK          ;NEXT HIGHER BANK
3298 014452  002353                    BGE     2$              ;IF NOT DONE - LOOP ON THIS BANK
3299                            ;TERMINATION OF PATTERN LOOP
3300 014454  004737  043366            CALL    INCRPT          ;NEXT HIGHER PATTERN
3301 014460  001346                    BNE     1$              ;OK - LOOP; ELSE CONTINUE
3302                            ;END OF LOOPS
3303 014462  005137  002132            COM     TMFLAG          ;COMPLEMENT TYPE OF MEMORY
3304                                                            ;IS THIS AN EVEN NUMBER PASS?
3305 014466  001403                    BEQ     5$              ;YES - SKIP
3306                            ;**NOTE** RECURSIVE CALL
3307 014470  004737  014372            CALL    PAFBAW          ;CALL SELF
3308 014474  000207                    RETURN
3309 014476  005137  002540    5$:     COM     WORST           ;4TH PASS?
3310 014502  001003                    BNE     6$              ;YES - SKIP
3311                            ;**NOTE** RECURSIVE CALL
3312 014504  004737  014372            CALL    PAFBAW          ;CALL SELF
3313 014510  000207                    RETURN
3314 014512  005737  002124    6$:     TST     RLFLAG          ;HAVE WE BEEN RELOCATED?
3315 014516  001401                    BEQ     7$              ;NO - SKIP
3316 014520  000207                    RETURN                  ;YES - RETURN
3317 014522  004737  041154    7$:     CALL    RELOCATE        ;MOVE & MAP PROGRAM
3318 014526                     ON.ERROR THEN $RETURN
3319                            ;**NOTE** RECURSIVE CALL
3320 014532  004737  014372            CALL    PAFBAW          ;CALL SELF
3321 014536  004737  042042            CALL    UNRELOCATE      ;UNMOVE & UNMAP PROGRAM
3322 014542  000207                    RETURN
```

```
3325 014544                          PARBAF: SUBTST  <<PATTERNS REVERSE,BANKS FORWARD      **RECURSIVE**>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       PATTERNS REVERSE,BANKS FORWARD      **RECURSIVE**
                                     ;*****************************************************************************
3326 014544  004737  043402                  CALL    HIPAT           ;SET HIGHTEST PATTERNS
3327                                         ;START OF PATTERN LOOP
3328 014550  005037  002100          1$:     CLR     BANK            ;SET BANK TO 0
3329                                         ;START OF BANK LOOP
3330 014554  004737  042700          2$:     CALL    EXBANK          ;EXAMINE BANK
3331 014560  004737  043350                  CALL    BANKOK          ;CORRECT MEMORY FOR THIS BANK?
3332 014564  001010                          BNE     4$              ;NO - GO TO BANK LOOP TERMINATOR
3333 014566  005737  002114                  TST     ACFLAG          ;CAN WE ACCESS THIS BANK?
3334 014572  001405                          BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3335 014574  005737  002122                  TST     RRFLAG          ;RELOCATION REQUIRED?
3336 014600  001002                          BNE     4$              ;YES - GO TO BANK LOOP TERMINATION
3337 014602  004737  015046                  CALL    MTEST           ;GO TEST CORRECT MEMORY
3338                                         ;TERMINATION OF BANK LOOP
3339 014606  005037  002214          4$:     CLR     CONTFLAG
3340 014612  004737  043412                  CALL    INCBNK          ;NEXT HIGHER BANK
3341 014616  002356                          BGE     2$              ;IF NOT DONE - LOOP ON THIS BANK
3342                                         ;TERMINATION OF PATTERN LOOP
3343 014620  005337  002110                  DEC     PATTERN         ;NEXT LOWER PATTERN
3344 014624  100351                          BPL     1$              ;OK - LOOP; ELSE CONTINUE
3345                                         ;END OF LOOPS
3346 014626  005137  002132                  COM     TMFLAG          ;COMPLEMENT TYPE OF MEMORY
3347                                                                 ;IS THIS AN EVEN NUMBER PASS?
3348 014632  001403                          BEQ     5$              ;YES - SKIP
3349                                         ;**NOTE** RECURSIVE CALL
3350 014634  004737  014544                  CALL    PARBAF          ;CALL SELF
3351 014640  000207                          RETURN
3352 014642  005737  002124          5$:     TST     RLFLAG          ;HAVE WE BEEN RELOCATED?
3353 014646  001401                          BEQ     6$              ;NO - SKIP
3354 014650  000207                          RETURN                  ;YES - RETURN
3355 014652  004737  041154          6$:     CALL    RELOCATE        ;MOVE & MAP PROGRAM
3356 014656                                  ON.ERROR THEN $RETURN
3357                                         ;**NOTE** RECURSIVE CALL
3358 014662  004737  014544                  CALL    PARBAF          ;CALL SELF
3359 014666  004737  042042                  CALL    UNRELOCATE      ;UNMOVE & UNMAP PROGRAM
3360 014672  000207                          RETURN
```

```
3363 014674                          PARBAW: SUBTST  <<PATTERNS REVERSE,BANKS WORST FIRST  **RECURSIVE**>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       PATTERNS REVERSE,BANKS WORST FIRST  **RECURSIVE**
                                     ;*****************************************************************************
3364 014674  004737  043402                  CALL    HIPAT           ;SET HIGHTEST PATTERN
3365                                         ;START OF PATTERN LOOP
3366 014700  005037  002100          1$:     CLR     BANK            ;SET BANK TO 0
3367                                         ;START OF BANK LOOP
3368 014704  004737  042700          2$:     CALL    EXBANK          ;EXAMINE BANK
3369 014710  004737  043350                  CALL    BANKOK          ;CORRECT MEMORY FOR THIS BANK?
3370 014714  001013                         BNE     4$              ;NO - GO TO BANK LOOP TERMINATOR
3371 014716  005737  002114                 TST     ACFLAG          ;CAN WE ACCESS THIS BANK?
3372 014722  001410                         BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3373 014724  005737  002126                 TST     BMFLAG          ;IS THIS BAD MEMORY (WORST FIRST)
3374 014730  001405                         BEQ     4$              ;NO - GO TO BANK LOOP TERMINATION
3375 014732  005737  002122                 TST     RRFLAG          ;RELOCATION REQUIRED?
3376 014736  001002                         BNE     4$              ;YES - GO TO BANK LOOP TERMINATION
3377 014740  004737  015046                 CALL    MTEST           ;GO TEST CORRECT MEMORY
3378                                         ;TERMINATION OF BANK LOOP
3379 014744  005037  002214          4$:     CLR CONTFLAG
3380 014750  004737  043412                 CALL    INCBNK          ;NEXT HIGHER BANK
3381 014754  002353                         BGE     2$              ;IF NOT DONE - LOOP ON THIS BANK
3382                                         ;TERMINATION OF PATTERN LOOP
3383 014756  005337  002110                 DEC     PATTERN         ;NEXT LOWER PATTERN
3384 014762  100346                         BPL     1$              ;OK - LOOP; ELSE CONTINUE
3385                                         ;END OF LOOPS
3386 014764  005137  002132                 COM     TMFLAG          ;COMPLEMENT TYPE OF MEMORY
3387                                                                 ;IS THIS AN EVEN NUMBER PASS?
3388 014770  001403                         BEQ     5$              ;YES - SKIP
3389                                         ;**NOTE** RECURSIVE CALL
3390 014772  004737  014674                 CALL    PARBAW          ;CALL SELF
3391 014776  000207                         RETURN
3392 015000  005137  002540          5$:     COM     WORST           ;4TH PASS?
3393 015004  001003                         BNE     6$              ;YES - SKIP
3394                                         ;**NOTE** RECURSIVE CALL
3395 015006  004737  014674                 CALL    PARBAW          ;CALL SELF
3396 015012  000207                         RETURN
3397 015014  005737  002124          6$:     TST     RLFLAG          ;HAVE WE BEEN RELOCATED?
3398 015020  001401                         BEQ     7$              ;NO - SKIP
3399 015022  000207                         RETURN                  ;YES - RETURN
3400 015024  004737  041154          7$:     CALL    RELOCATE        ;MOVE & MAP PROGRAM
3401 015030                                  ON.ERROR THEN $RETURN
3402                                         ;**NOTE** RECURSIVE CALL
3403 015034  004737  014674                 CALL    PARBAW          ;CALL SELF
3404 015040  004737  042042                 CALL    UNRELOCATE      ;UNMOVE & UNMAP PROGRAM
3405 015044  000207                         RETURN
```

```
3408 015046                          MTEST:  SUBTST  <<SUBR  SETUP MEMORY TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       SUBR    SETUP MEMORY TEST
                                     ;*****************************************************************************
3409 015046                                  SET     HEADER                  ;INITIALIZE HEADER MESSAGE TYPEOUT
3410 015054                                  SET     MUT                     ;INDICATE THERE IS A MEMORY UNDER TEST
3411 015062  005037  002256                  CLR     PASFLG
3412 015066  005737  002116                  TST     MKFLAG                  ;ECC?
3413 015072  001413                          BEQ     MT1                     ;NO - SKIP
3414 015074                                  BEGIN   HOLDLOOP
3415 015074                                    IF CONTFLAG IS TRUE THEN LEAVE HOLDLOOP
3416 015102                                    IF SKIPMK IS FALSE
3417 015110  004737  015142                      CALL        MKCONTROL
3418 015114                                    END; OF IF SKIPMK
3419 015114                                  END     HOLDLOOP
3420 015114  004737  016354                  CALL    MKTEST                  ;YES - DO ECC TESTS
3421 015120  000402                          BR      MT2
3422 015122  004737  016574          MT1:    CALL    MJTEST                  ;DO PARITY TESTS
3423 015126  005037  002106          MT2:    CLR     MUT                     ;NOW - NO MEMORY UNDER TEST
3424 015132                                  SET     HEADER                  ,ALLOW HEADERS NORMAL
3425 015140  000207                          RETURN
```

```
3428 015142                                  MKCONTROL:SUBTST        <<SUBR  TEST ECC CSR LOGIC DISPATCH>>
                                             ;*****************************************************************************
                                             ;*SUBTEST       SUBR    TEST ECC CSR LOGIC DISPATCH
                                             ;*****************************************************************************
3429                                                 ;THE NEXT TWO MODULES SOLVE THE PROBLEM OF
3430                                                 ;HOW TO RUN THE CSR TESTS ON EACH ECC MEMORY
3431                                                 ;
3432 015142                                          IF SELONLY IS TRUE THEN $RETURN
3433 015152                                          IF INHECC IS TRUE THEN $RETURN
3434 015162                                          PUSH    BANK,R0,R1,R2,R3
3435 015176  012737  060000  002224                 MOV     #FIRST,CSRFBANK         ;SET FIRST TEST ADDRFSS TO FIRST ADDR.
3436 015204  012737  157776  002226                 MOV     #LAST,CSRLBANK
3437 015212  005037  002230                         CLR     CSRINT
3438 015216  005037  002232                         CLR     SPLTCSR
3439 015222  005037  002302                         CLR     CSRLOOP                 ; AND ZERO THE LOOP COUNTER
3440 015226  013700  002102                         MOV     BANKINDEX,R0            ;GET THE BANK INDEX
3441 015232  016001  002624                         MOV     CONFIG(R0),R1           ;GET CSR NUMBER
3442 015236  000301                                 SWAB    R1
3443 015240  042701  177760                         BIC     #^C17,R1
3444 015244  006301                                 ASL     R1
3445 015246  010137  002476                         MOV     R1,CSRHOLD              ;STORE IN THE LOW BYTE
3446 015252  005737  002134                         TST     INTFLAG                 ;IS THIS BANK INTERLEAVED?
3447 015256  001421                                 BEQ     1$                      ;BRANCH IF NOT INTERLEAVED
3448 015260  005237  002232                         INC     SPLTCSR
3449 015264  012737  120000  002226                 MOV     #120000,CSRLBANK
3450 015272  005237  002302                         INC     CSRLOOP                 ;WE MUST LOOP TWICE FOR AN INTERLEAVED BANK
3451 015276  005237  002230                         INC     CSRINT
3452 015302  016001  002624                         MOV     CONFIG(R0),R1           ;GET THE INTERLEAVE CSR NUMBER
3453 015306  072127  177775                         ASH     #-3,R1
3454 015312  042701  160777                         BIC     #^C17000,R1
3455 015316  050137  002476                         BIS     R1,CSRHOLD              ;STORE IT IN CSRHOLD'S UPPER BYTE
3456 015322  005003                         1$:     CLR     R3
3457 015324  116337  002476  002146  MKLOOP: MOVB    CSRHOLD(R3),CSRNO
3458 015332  042737  177741  002146                 BIC     #^C36,CSRNO             ;CLEAR ANY UNNECESSARY BITS
3459 015340                                            FOR R2 := #0 TO CSRINT
3460 015342                                              FOR CSRFIRST := CSRFBANK TO CSRLBANK BY #4000
3461 015350                                              MAP BANK                    ;MAP TEST SPACE TO BANK
3462 015364  104511                                      INVALIDATE                  ;INVALIDATE BACKROUND PATTERN
3463 015366                                                BEGIN CSRSTUFF
3464 015366  005037  002304                               CLR       SUCCESS
3465 015372                                                 IF ACFLAG IS TRUE AND RRFLAG IS FALSE
3466 015406  013737  002220  002222                          MOV   CSRFIRST,CSRLAST
3467 015414  062737  004000  002222                          ADD   #4000,CSRLAST
3468 015422                                                    FOR TESTADD := CSRFIRST TO CSRLAST BY #4
3469 015430  013737  002362  002364                             MOV       TESTADD,TESTADD+2
3470 015436  005737  002232                                   TST       SPLTCSR
3471 015442  001404                                           BEQ       1$
3472 015444  062737  040000  002364                         ADD       #40000,TESTADD+2
3473 015452  000403                                         BR        2$
3474 015454  062737  00000_  002364                   1$:   ADD       #2,TESTADD+2
3475 015462  004737  015750                           2$:   CALL      SBETEST
3476 015466                                                  ON.NOERROR
3477 015470  104424                                           CACHOFF                 ;TURN CACHE OFF
3478 015472  005037  002074                                   CLR     NOPAR           ;INDICATE PARITY ACTION
3479 015476                                                    FOR I := #0 TO #27
3480 015502                                                      SET   HEADER
3481 015510  005037  002256                                     CLR   PASFLG
```

```
3482 015514                                                    LET R0 := I
3483 015520                                                    PUSH  R3                ;SAVE LOOP COUNTER
3484 015522  010637  002142                                    MOV   SP,CTLKVEC        ;SAVE VECTOR IN CSR OF ^K
3485 015526  162737  000002  002142                            SUB   #2,CTLKVEC
3486 015534  004737  016254                                    CALL CSRCASE
3487 015540                                                    POP   R3                ;RESTORE LOOP COUNTER
3488 015542                                                  END ;OF FOR I
3489 015556  104423                                          CACHON                    ;TURN CACHE ON
3490 015560                                                  SET     SUCCESS
3491 015566                                                  LEAVE CSRSTUFF
3492 015570                                                END ;OF ON.NOERROR
3493 015570                                              END ;OF FOR TESTADD
3494 015606                                            END ;OF IF
3495 015606                                          END CSRSTUFF
3496 015606                                          IF SUCCESS IS FALSE
3497 015614                                            TYPE    MSGA34
3498 015620                                            TYPOCS  BANK,<TYPES BANK NUMBER>,3
3499 015630                                            TYPE    MSGB34
3500 015634                                          END ;OF IF SUCCESS
3501 015634                                        END; OF FOR CSRFIRST
3502 015652  005237  002232                        INC  SPLTCSR
3503 015656                                      END; OF FOR R2
3504 015666  062737  000002  002224          ADD     #2,CSRFBANK
3505 015674  012737  000001  002232          MOV     #1,SPLTCSR
3506 015702  005203                          INC     R3
3507 015704  020337  002302                  CMP     R3,CSRLOOP
3508 015710  003002                          BGT     1$
3509 015712  000137  015324                  JMP     MKLOOP
3510 015716  104472                  1$:     ECCINIT         ,TRAP ON DOUBLE BIT ERRORS (NORMAL)
3511 015720                                  SET     CONTFLAG
3512 015726  005037  002232                  CLR     SPLTCSR
3513 015732                                  POP     R3,R2,R1,R0,BANK
3514 015746  000207                          RETURN
```

```
3517 015750                          SBETEST:SUBTST  <<CHECK FOR SBE FREE LOCATIONS>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       CHECK FOR SBE FREE LOCATIONS
                                     ;*****************************************************************************
3518                                         ;IN ORDER TO DETERMINE IF A LOCATION IS SBE FREE I DO THIS
3519                                         ;
3520                                         ;WRITE ZEROS WITH ECC DISABLE
3521                                         ;READ ZEROS BACK
3522                                         ;IF NOT ZEROS THEN RETURN ERROR
3523                                         ;
3524                                         ;WRITE ZEROS WITH ECC ENABLED BUT TRAPS DISABLED
3525                                         ;READ ZEROS BACK
3526                                         ;IF NOT ZEROS THEN RETURN ERROR
3527                                         ;
3528                                         ;TEST THE LOCATION FROM THE PAR'S (WITH NO PROGRAM FETCHES)
3529                                         ;IF THERE WERE ANY SBE'S OR DBE'S THEN RETURN ERROR
3530                                         ;
3531                                         ;COMPLIMENT ZEROS TO ONES WITH ECC DISABLE
3532                                         ;READ ONES BACK
3533                                         ;IF NOT ONES THEN RETURN ERROR
3534                                         ;
3535                                         ;WRITE 100,,100000,00000 (CHECKBITS COMPLIMENT OF BEFORE)
3536                                         ;  WITH ECC ENABLED BUT TRAPS DISABLED
3537                                         ;TEST THE LOCATION FROM THE PAR'S (WITH NO PROGRAM FETCHES)
3538                                         ;IF THERE WERE ANY SBE'S OR DBE'S THEN RETURN ERROR
3539                                         ;
3540                                         ;IF NONE OF THE ABOVE FORCES A RETURN ERROR THEN RETURN NO.ERROR
3541                                         .ENABL  LSB
3542 015750                          PUSH    R0,R1,R4                ;PUSH R0,R1,R4 ONTO STACK
3543 015756  013701  002362          MOV     TESTADD,R1
3544 015762  013704  002364          MOV     TESTADD+2,R4
3545 015766                          TESTAREA                        ;ENTER TEST MODE
3546 015774  104424                  CACHOFF                         ;TURN CACHE OFF
3547 015776  104471                  ECC1DIS                         ;DISABLE ECC ON 1 SELECTED CSR
3548 016000                          CLEAR   (R1),(R4)
3549 016004  005711                  TST     (R1)
3550 016006  001107                  BNE     SBENT
3551 016010  005714                  TST     (R4)
3552 016012  001105                  BNE     SBENT
3553
3554 016014  104503                  CLR1CSR                         ;CLEAR 1 SELECTED CSR
3555 016016                          CLEAR   (R1),(R4)
3556 016022  005711                  TST     (R1)
3557 016024  001100                  BNE     SBENT
3558 016026  005714                  TST     (R4)
3559 016030  001076                  BNE     SBENT
3560
3561 016032  104510                  TSTREAD                 ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
3562 016034                          IF #BIT15!BIT4 SET.IN CSR
3563 016044                            SET SKPERR                    ;DISABLE ERRGEN'S ERROR PRINTOUT
3564 016052  104512                    ERRGEN
3565 016054  013700  002430            MOV   ERRADD,R0
3566 016060  072027  177774            ASH   #-4,R0
3567 016064  042700  177600            BIC   #^C177,R0
3568 016070                            IF BANK EQ R0 THEN GOTO SBENT
3569 016076                          END: OF IF #BIT15
3570 016076  104471                  ECC1DIS                         ;DISABLE ECC ON 1 SELECTED CSR
```

```
3571 016100  005111                          COM     (R1)
3572 016102  005114                          COM     (R4)
3573 016104  023711  002554                  CMP     ONES,(R1)
3574 016110  001046                          BNE     SBENT
3575 016112  023714  002554                  CMP     ONES,(R4)
3576 016116  001043                          BNE     SBENT
3577
3578 016120  104503                          CLR1CSR                         ;CLEAR 1 SELECTED CSR
3579 016122  005011                          CLR     (R1)
3580 016124  012714  100000                  MOV     #BIT15,(R4)
3581 016130  005711                          TST     (R1)
3582 016132  001035                          BNE     SBENT
3583 016134  022714  100000                  CMP     #BIT15,(R4)
3584 016140  001032                          BNE     SBENT
3585
3586 016142  104510                          TSTREAD                 ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
3587 016144                                  IF #BIT15!BIT4 SET.IN CSR
3588 016154                                    SET SKPERR                    ;DISABLE ERRGEN'S ERROR PRINTOUT
3589 016162  104512                            ERRGEN
3590 016164  013700  002430                    MOV   ERRADD,R0
3591 016170  072027  177774                    ASH   #-4,R0
3592 016174  042700  177600                    BIC   #^C177,R0
3593 016200                                    IF BANK EQ R0 THEN GOTO SBENT
3594 016206                                  END; OF IF #BIT15
3595
3596 016206  104417                          KERNEL                          ;ENTER KERNEL MODE
3597 016210  104473                          ECC1INIT                        ;INITIALIZE 1 SELECTED CSR
3598 016212  104423                          CACHON                          ;TURN CACHE ON
3599 016214                                  POP     R4,R1,R0                ;POP R0,R1 & R4 FROM STACK
3600 016222                                  $RETURN NOERROR
3601
3602 016226  104503          SBENT:          CLR1CSR                         ;CLEAR 1 SELECTED CSR
3603 016230                                  CLEAR   (R1),(R4)
3604 016234  104417                          KERNEL                          ;ENTER KERNEL MODE
3605 016236  104473                          ECC1INIT                        ;INITIALIZE 1 SELECTED CSR
3606 016240  104423                          CACHON                          ;TURN CACHE ON
3607 016242                                  POP     R4,R1,R0                ;POP R0,R1 & R4 FROM STACK
3608 016250                                  $RETURN ERROR
3609                                         .DSABL  LSB
```

```
3612 016254                          CSRCASE:SUBTST  <<CSR PATTERN CASE STATEMENT>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       CSR PATTERN CASE STATEMENT
                                     ;*****************************************************************************
3613 016254                                  CASE R0
3614                                           ;WARNING IF YOU CHANGE THIS TABLE ALSO
3615                                           ;CHANGE '$DDW0' - '$DDW5' (THE PATTERN BIT MAP)
3616
3617                                           ;PAT  TIME              DISCRIPTION
3618 016264  020014                MKCSRT:     MT0006          ;<1 SEC          INITIAL DATA TEST
3619 016266  020112                            MT0010          ;<1 SEC          BYTE ADDRESSING TEST
3620 016270  022402                            MT0025          ;<1 SEC          INTERRUPT ENABLE TEST
3621 016272  020146                            MT0011          ;<2 SEC          CREATE SINGLE BIT ERROR TEST
3622 016274  020214                            MT0012          ;<1 SEC          WRITE BYTE CLEARS SBE TEST
3623 016276  020310                            MT0013          ; 1 SEC          CREATE DOUBLE BIT ERROR TEST
3624 016300  020364                            MT0014          ; 1 SEC          WRITE INHIBIT DURING DATIP WITH DBE
3625 016302  020442                            MT0015          ; 1 SEC          WRITE INHIBIT OF BYTE WITH DBE
3626 016304  020510                            MT0016          ;<1 SEC          WRITE INHIBIT OF WORD WITH DBE
3627 016306  025074                            MT0999          ; 0 SEC          NULL TEST
3628 016310  025074                            MT0999          ; 0 SEC          NULL TEST
3629 016312  025074                            MT0999          ; 0 SEC          NULL TEST
3630 016314  025074                            MT0999          ; 0 SEC          NULL TEST
3631 016316  025074                            MT0999          ; 0 SEC          NULL TEST
3632 016320  025074                            MT0999          ; 0 SEC          NULL TEST
3633 016322  025074                            MT0999          ; 0 SEC          NULL TEST
3634 016324  025074                            MT0999          ; 0 SEC          NULL TEST
3635 016326  025074                            MT0999          ; 0 SEC          NULL TEST
3636 016330  025074                            MT0999          ; 0 SEC          NULL TEST
3637 016332  025074                            MT0999          ; 0 SEC          NULL TEST
3638 016334  025074                            MT0999          ; 0 SEC          NULL TEST
3639 016336  025074                            MT0999          ; 0 SEC          NULL TEST
3640 016340  025074                            MT0999          ; 0 SEC          NULL TEST
3641 016342  025074                            MT0999          ; 0 SEC          NULL TEST
3642 016344                          END ;OF CASE R0
3643 016352  000207                RETURN
```

```
3646 016354                                MKTEST: SUBTST  <<SUBR  ECC TEST DISPATCH>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       SUBR    ECC TEST DISPATCH
                                           ;*****************************************************************************
3650 016354                                        IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE
3651 016372  104470                                  ECCDIS                        ;DISABLE ERROR CORRECTION
3652 016374                                        ELSE
3653 016376  104502                                  CLRCSR                        ;CLEAR ALL CSR'S
3654 016400                                        END ;OF IF
3655 016400  012737  000002  002074                MOV     #2,NOPAR              ;INDICATE PARITY ACTION
3656 016406  012737  000002  002276                MOV     #2,PCBUMP             ;TRAPS ADD 2 TO PC
3657 016414  013700  002110                        MOV     PATTERN,R0            ;GET PATTERN NUMBER
3658 016420  006300                                ASL     R0                    ;MAKE IT A WORD ADDRESS
3659 016422                                        IF MKPAT(R0) NE #MT0034 AND MKPAT(R0) NE #MT0999
3660 016442  104511                                  INVALIDATE                  ;INVALIDATE BACKGROUND PATTERN ON 'BANK'
3661 016444                                        END ;OF IF MKPAT(R0)
3662 016444  010637  002142                        MOV     SP,CTLKVEC            ;SAVE VECTOR IN CASE OF ^K
3663 016450  162737  000002  002142                SUB     #2,CTLKVEC
3664 016456  004770  016514                        CALL    @MKPAT(R0)            ;INDEX OFF TABLE
3665 016462                                        IF #SW0 SET.IN @SWR OR ACTFLAG IS TRUE
3666 016500  104506                                  ENASBE                        ;TRAP ON SINGLE BIT ERRORS
3667 016502                                        ELSE
3668 016504  104472                                  ECCINIT                       ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
3669 016506                                        END ;OF IF #SW0
3670 016506  005037  002074                        CLR     NOPAR                 ;INDICATE PARITY ACTION
3671 016512  000207                                RETURN
3672
3673                                               ;WARNING IF YOU CHANGE THIS TABLE ALSO
3674                                               ;CHANGE ''$DDW0'' - ''$DDW5'' (THE PATTERN BIT MAP)
3675                                               ;PAT    TIME              DISCRIPTION
3676 016514                                MKPAT:  ;NOTE MT0034 MUST BE FIRST & LAST
3677 016514  024616                                MT0034  ;<1 SEC           ;SOFT ERROR - BACKGROUND PATTERN TEST
3678 016516  020556                                MT0017  ;<1 SEC           ;HOLDING 1'S & 0'S TEST
3679 016520  020050                                MT0007  ;<1 SEC           ;ADDRESS BIT TEST
3680 016522  017036                                MT0001  ;<1 SEC           ;ADDRESS TEST
3681 016524  017154                                MT0002  ;<1 SEC           ;COMPLEMENT ADDRESS TEST
3682 016526  017542                                MT0004  ; 1 SEC           ;ROTATING ZEROS TEST
3683 016530  017662                                MT0005  ; 1 SEC           ;ROTATING ONES TEST
3684 016532  021630                                MT0021  ; 1 SEC           ;MARCHING 0'S & 1'S TEST
3685 016534  020600                                MT0020  ;<1 SEC           ;MARCHING 1'S & 0'S IN CHECK BITS
3686 016536  022052                                MT0022  ;10 SEC           ;REFRESH & SHIFTING DIAGONAL  TEST
3687 016540  022450                                MT0026  ;<1 SEC           ;RANDOM DATA TEST
3688 016542  022150                                MT0024  ;20 SEC           ;FAST GALLOPING PATTERN TEST
3689 016544  023712                                MT0031  ; 3 SEC           ;SOB-A-LONG TEST
3690 016546  024102                                MT0032  ;<1 SEC           ;WRITE RECOVERY TEST
3691 016550  024430                                MT0033  ;35 SEC           ;BRANCH GOBBLE TEST
3692 016552  024616                                MT0034  ;<1 SEC           ;SOFT ERROR - BACKGROUND PATTERN TEST
3693                                               ;NOTE MT0034 MUST BE FIRST & LAST
3694 016554  025074                                MT0999  ; 0 SEC           ;NULL TEST
3695 016556  025074                                MT0999  ; 0 SEC           ;NULL TEST
3696 016560  025074                                MT0999  ; 0 SEC           ;NULL TEST
3697 016562  025074                                MT0999  ; 0 SEC           ;NULL TEST
3698 016564  025074                                MT0999  ; 0 SEC           ;NULL TEST
3699 016566  025074                                MT0999  ; 0 SEC           ;NULL TEST
3700 016570  025074                                MT0999  ; 0 SEC           ;NULL TEST
3701 016572  025074                                MT0999  ; 0 SEC           ;NULL TEST
```

```
3704 016574                          MJTEST: SUBTST  <<SUBR  PARITY TEST DISPATCH>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       SUBR    PARITY TEST DISPATCH
                                     ;*****************************************************************************
3708 016574  012737  000002  002074          MOV     #2,NOPAR                ;INDICATE PARITY ACTION
3709 016602  012737  000002  002276          MOV     #2,PCBUMP               ;TRAPS ADD 2 TO PC
3710 016610  012737  060000  002362          MOV     #FIRST,TESTADD
3711 016616  012737  060002  002364          MOV     #FIRST+2,TESTADD+2
3712 016624  013700  002110                  MOV     PATTERN,R0              ;GET PATTERN NUMBER
3713 016630  006300                          ASL     R0                      ;MAKE IT A WORD ADDRESS
3714 016632                                  IF MJPAT(R0) NE #MT0034 AND MJPAT(R0) NE #MT0999
3715 016652  104511                            INVALIDATE                    ;INVALIDATE BACKGROUND PATTERN ON 'BANK''
3716 016654                                  END ;OF IF MJPAT(R0)
3717 016654  010637  002142                  MOV     SP,CTLKVEC              ;SAVE VECTOR IN CASE OF ^K
3718 016660  162737  000002  002142          SUB     #2,CTLKVEC
3719 016666  004770  016700                  CALL    @MJPAT(R0)              ;INDEX OFF TABLE
3720 016672  005037  002074                  CLR     NOPAR                   ;INDICATE PARITY ACTION
3721 016676  000207                          RETURN
3722
3723                                         ;WARNING IF YOU CHANGE THIS TABLE ALSO
3724                                         ;CHANGE '$DDW0'' - ''$DDW5'' (THE PATTERN BIT MAP)
3725
3726                                         ;PAT    TIME            DISCRIPTION
3727 016700                          MJPAT:  ;NOTE MT0034 MUST BE FIRST & LAST
3728 016700  024616                          MT0034  ;<1 SEC         ;SOFT ERROR - BACKGROUND PATTERN TEST
3729 016702  020014                          MT0006  ;<1 SEC         ;INITIAL DATA TEST
3730 016704  020556                          MT0017  ;<1 SEC         ;HOLDING 1'S & 0'S TEST
3731 016706  020050                          MT0007  ;<1 SEC         ;ADDRESS BIT TEST
3732 016710  017036                          MT0001  ;<1 SEC         ;ADDRESS TEST
3733 016712  017154                          MT0002  ;<1 SEC         ;COMPLEMENT ADDRESS TEST
3734 016714  017312                          MT0003  ; 1 SEC         ;3 XOR 9 WORST CASE NOISE TEST
3735 016716  017542                          MT0004  ; 1 SEC         ;ROTATING ZEROS TEST
3736 016720  017662                          MT0005  ; 1 SEC         ;ROTATING ONES TEST
3737 016722  021630                          MT0021  ; 1 SEC         ;MARCHING 0'S & 1'S TEST
3738 016724  024762                          MT0035  ;<1 SEC         ;WORSE CASE NOISE PARITY TEST
3739 016726  022052                          MT0022  ;10 SEC         ;REFRESH TEST
3740 016730  022104                          MT0023  ;10 SEC         ;SHIFTING DIAGONAL  TEST
3741 016732  022450                          MT0026  ;<1 SEC         ;RANDOM DATA TEST
3742 016734  022150                          MT0024  ;20 SEC         ;FAST GALLOPING PATTERN TEST
3743 016736  023712                          MT0031  ; 3 SEC         ;SOB-A-LONG TEST
3744 016740  024102                          MT0032  ;<1 SEC         ;WRITE RECOVERY TEST
3745 016742  024430                          MT0033  ;35 SEC         ;BRANCH GOBBLE TEST
3746 016744  024616                          MT0034  ;<1 SEC         ;SOFT ERROR - BACKGROUND PATTERN TEST
3747                                         ;NOTE MT0034 MUST BE FIRST & LAST
3748 016746  025074                          MT0999  ; 0 SEC         ;NULL TEST
3749 016750  025074                          MT0999  ; 0 SEC         ;NULL TEST
3750 016752  025074                          MT0999  ; 0 SEC         ;NULL TEST
3751 016754  025074                          MT0999  ; 0 SEC         ;NULL TEST
3752 016756  025074                          MT0999  ; 0 SEC         ;NULL TEST
```

```
                                        .SBTTL  PATTERNS
                                        .SBTTL  MEMORY TEST SETUP ROUTINES
016760                          MT0000: SUBTST  <<MT0000        SETUP DATA PATTERN TEST>>
                                ;*********************************************************************
                                ;*SUBTEST        MT0000  SETUP DATA PATTERN TEST
                                ;*********************************************************************
016760  005037  002260                  CLR     REALPAT         ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
016764  012700  060000                  MOV     #FIRST,R0
016770  012701  040000                  MOV     #SIZE,R1
016774  004737  035016                  CALL    REGCOPY
017000  005737  002424                  TST     NO22BIT         ;ARE WE ON AN 11/44?
017004  001406                          BEQ     1$              ;BRANCH IF YES
017006  012737  025514  002254          MOV     #MTP000,SUPDOADD ;ELSE DO PATTERN IN MAIN MEMORY
017014  004737  025322                  CALL    SUPD03
017020  000207                          RETURN
017022                          1$:     BMOV    MTP000
017030  004737  025144                  CALL    SUPD01                  ;DO IT IN SUPERVISOR MODE
017034  000207                          RETURN
017036                          MT0001: SUBTST  <<MT0001        SETUP ADDRESS TEST>>
                                ;*********************************************************************
                                ;*SUBTEST        MT0001  SETUP ADDRESS TEST
                                ;*********************************************************************
017036  012737  000001  002260          MOV     #1,REALPAT              ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
017044  012700  060000                  MOV     #FIRST,R0
017050  012701  040000                  MOV     #SIZE,R1
017054  005737  002426                  TST     NOSUPER
017060  001005                          BNE     2$
017062  023737  172252  172254          CMP     SIPAR5,SIPAR6
017070  001007                          BNE     4$
017072  000404                          BR      3$
017074  023737  177652  177654  2$:     CMP     UIPAR5,UIPAR6
017102  001002                          BNE     4$
017104  012701  030000          3$:     MOV     #30000,R1
017110  005002                  4$:     CLR     R2
017112  004737  035016                  CALL    REGCOPY
017116  005737  002424                  TST     NO22BIT                 ;IS THIS AN 11/44?
017122  001406                          BEQ     1$                      ;BRANCH IF IT IS
017124  012737  025540  002254          MOV     #MTP001,SUPDOADD        ;SET UP CALLING ADDRESS
017132  004737  025322                  CALL    SUPD03
017136  000207                          RETURN
017140                          1$:     BMOV    MTP001
017146  004737  025144                  CALL    SUPD01                  ;DO IT IN SUPERVISOR MODE
017152  000207                          RETURN
017154                          MT0002: SUBTST  <<MT0002        SETUP COMPLEMENT ADDRESS TEST>>
                                ;*********************************************************************
                                ;*SUBTEST        MT0002  SETUP COMPLEMENT ADDRESS TEST
                                ;*********************************************************************
017154  012737  000002  002260          MOV     #2,REALPAT              ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
017162  012700  160000                  MOV     #LAST+2,R0
017166  012701  040000                  MOV     #SIZE,R1
017172  012704  060000                  MOV     #FIRST,R4
017176  012705  100001                  MOV     #100001,R5
017202  005737  002426                  TST     NOSUPER
017206  001005                          BNE     2$
017210  023737  172252  172254          CMP     SIPAR5,SIPAR6
017216  001013                          BNE     4$
```

```
 3802 017220  000404                          BR      3$
 3803 017222  023737  177652  177654  2$:     CMP     UIPAR5,UIPAR6
 3804 017230  001006                          BNE     4$
 3805 017232  012701  030000          3$:     MOV     #30000,R1
 3806 017236  012700  140000                  MOV     #140000,R0
 3807 017242  012705  120001                  MOV     #120001,R5
 3808 017246  012702  000001          4$:     MOV     #1,R2
 3809 017252  010103                          MOV     R1,R3
 3810 017254  005737  002424                  TST     NO22BIT                 ;IS THIS AN 11/44?
 3811 017260  001406                          BEQ     1$                      ;BRANCH IF TRUE
 3812 017262  012737  025572  002254          MOV     #MTP002,SUPDOADD        ;SET UP CALLING ADDRESS
 3813 017270  004737  025322                  CALL    SUPD03
 3814 017274  000207                          RETURN
 3815 017276                          1$:     BMOV    MTP002
 3816 017304  004737  025144                  CALL    SUPD01
 3817 017310  000207                          RETURN
```

```
3820 017312                                MT0003: SUBTST  <<MT0003        SETUP 3 XOR 9 WORST CASE NOISE TEST>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       MT0003  SETUP 3 XOR 9 WORST CASE NOISE TEST
                                           ;*****************************************************************************
3821 017312                                        IF EUFLAG IS TRUE THEN $RETURN
3822 017322  012737  000003  002260               MOV     #3,REALPAT              ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3823 017330  005037  002276                       CLR     PCBUMP                  ;TRAPS DO NOT ADD TO PC
3824 017334  004737  035026          1$:          CALL    FLIPWARN                ;SETUP WARNING CONSTANTS & R2
3825 017340  012701  060000          2$:          MOV     #FIRST,R1               ;R1 <-- STARTING ADDRESS
3826 017344  012703  020000                       MOV     #20000,R3
3827 017350  072327  177770                       ASH     #-8.,R3                 ;R3 <-- R3 / 256.
3828 017354  012702  000004                       MOV     #4,R2                   ;SMALL LOOP SIZE
3829 017360  012705  000 00                       MOV     #64.,R5                 ;MEDIUM LOOP SIZE
3830 017364  005737  002424                       TST     NO22BIT                 ;IS THIS AN 11/44?
3831 017370  001415                               BEQ     3$                      ;BRANCH IF IT IS
3832 017372  104415                               SAVREG
3833 017374  012737  025624  002254               MOV     #MTPA03,SUPDOADD
3834 017402  004737  025322                       CALL    SUPD03                  ;DO IT IN MAIN MEMORY
3835 017406  104416                               RESREG
3836 017410  012737  025664  002254               MOV     #MTPB03,SUPDOADD
3837 017416  004737  025336                       CALL    SUPD04
3838 017422  000442                               BR      4$
3839 017424                          3$:          BMOV    MTPA03
3840 017432  104415                               SAVREG
3841 017434  004737  025144                       CALL    SUPD01
3842 017440                                       BMOV    MTPB03
3843 017446                                       BMOV    MTPC03,KDPAR0,8.
3844 017460                                       BMOV    MTPD03,SDPAR0,8.
3845 017472  012737  172360  177642               MOV     #KDPAR0,UIPAR1          ;SET UP PAR LINKS
3846 017500  012737  172260  172374               MOV     #SDPAR0,KDPAR6
3847 017506  012737  177644  172276               MOV     #UIPAR2,SDPAR7
3848 017514  012737  001032  172272               MOV     #1032,SDPAR5            ;CHANGE INST TO BR .+66 (BR TO KDPAR1)
3849 017522  104416                               RESREG
3850 017524  004737  025160                       CALL    SUPD02
3851 017530  022737  000003  002556    4$:        CMP     #3,FLIPLOC              ;DONE WITH 4 PATTERNS
3852                                               ;[(0,177777);(177777,0);(401,177777);(177777,401)]?
3853 017536  001276                               BNE     1$                      ;NO - LOOP
3854 017540  000207                               RETURN
3855
3856 017542                                MT0004: SUBTST  <<MT0004        SETUP ROTATING ZEROS TEST>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       MT0004  SETUP ROTATING ZEROS TEST
                                           ;*****************************************************************************
3857 017542  012737  000004  002260               MOV     #4,REALPAT              ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3858 017550  012737  000004  002276               MOV     #4,PCBUMP               ;TRAPS ADD 4 TO PC
3859 017556  013702  002554                       MOV     ONES,R2
3860 017562  004737  035156                       CALL    BACKGND                 ;WRITE BACKGROUND OF ONES
3861 017566  012700  060000                       MOV     #FIRST,R0
3862 017572  012701  040000                       MOV     #SIZE,R1
3863 017576  005737  002424                       TST     NO22BIT                 ;IS THIS AN 11/44?
3864 017602  001406                               BEQ     1$                      ;BRANCH IF IT IS
3865 017604  012737  025762  002254               MOV     #MTPA04,SUPDOADD        ;SET UP LINKS
3866 017612  004737  025336                       CALL    SUPD04
3867 017616  000207                               RETURN
3868 017620                          1$:          BMOV    MTPA04
3869 017626                                       BMOV    MTPB04,KDPAR0,8.
3870 017640  012737  172360  177652               MOV     #KDPAR0,UIPAR5
```

```
 3871 017646  012737  177654  172376          MOV     #UIPAR6,KDPAR7
 3872 017654  004737  025160                  CALL    SUPD02
 3873 017660  000207                          RETURN
 3874 017662                          MT0005: SUBTST  <<MT0005        SETUP ROTATING ONES TEST>>
                                      ;*****************************************************************************
                                      ;*SUBTEST       MT0005  SETUP ROTATING ONES TEST
                                      ;*****************************************************************************
 3875 017662  012737  000005  002260          MOV     #5,REALPAT              ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
 3876 017670  012737  000004  002276          MOV     #4,PCBUMP               ;TRAPS ADD 4 TO PC
 3877 017676  005002                          CLR     R2
 3878 017700  004737  035156                  CALL    BACKGND                 ;WRITE BACKGROUND OF ZEROS
 3879 017704  012700  060000                  MOV     #FIRST,R0
 3880 017710  012701  040000                  MOV     #SIZE,R1
 3881 017714  005737  002424                  TST     NO22BIT                 ;IS THIS AN 11/44?
 3882 017720  001414                          BEQ     1$                      ;BRANCH IF IT IS
 3883 017722  012737  026036  002254          MOV     #MTP005,SUPDOADD        ;SET UP LINKS
 3884 017730  012737  026052  026034          MOV     #MTP005+14,MTPB04+16
 3885 017736  004737  025336                  CALL    SUPD04
 3886 017742  012737  025776  026034          MOV     #MTPA04+14,MTPB04+16    ;RESET TEST'S ORIGINAL VALUE
 3887 017750  000207                          RETURN
 3888 017752                          1$:     BMOV    MTP005
 3889 017760                                  BMOV    MTPB04,KDPAR0,8.
 3890 017772  012737  172360  177652          MOV     #KDPAR0,UIPAR5
 3891 020000  012737  177654  172376          MOV     #UIPAR6,KDPAR7
 3892 020006  004737  025160                  CALL    SUPD02
 3893 020012  000207                          RETURN
```

```
                                MT0006: SUBTST  <<MT0006        SETUP INITIAL DATA TEST>>
                                ;*********************************************************************
                                ;*SUBTEST        MT0006  SETUP INITIAL DATA TEST
                                ;*********************************************************************
3897 020014  012737  000006  002260          MOV     #6,REALPAT              ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3898 020022  012737  000004  002276          MOV     #4,PCBUMP               ;TRAPS ADD 4 TO PC
3899 020030  012701  002362                  MOV     #TESTADD,R1
3900 020034  012737  026072  002254          MOV     #MTP006,SUPDOADD
3901 020042  004737  025322                  CALL    SUPD03                  ;DO IT IN SUPERVISOR MODE
3902 020046  000207                          RETURN
3903 020050                     MT0007: SUBTST  <<MT0007        SETUP ADDRESS BIT TEST>>
                                ;*********************************************************************
                                ;*SUBTEST        MT0007  SETUP ADDRESS BIT TEST
                                ;*********************************************************************
3904 020050  012737  000007  002260          MOV     #7,REALPAT              ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3905 020056  005002                          CLR     R2
3906 020060  004737  035156                  CALL    BACKGND                 ;OF ZEROS
3907 020064  012701  060000                  MOV     #FIRST,R1
3908 020070  012702  000001                  MOV     #1,R2
3909 020074  050201                          BIS     R2,R1
3910 020076  012737  026272  002254          MOV     #MTP007,SUPDOADD
3911 020104  004737  025322                  CALL    SUPD03                  ;DO IT IN SUPERVISOR MODE
3912 020110  000207                          RETURN
3913 020112                     MT0010: SUBTST  <<MT0010        SETUP BYTE ADDRESSING TEST>>
                                ;*********************************************************************
                                ;*SUBTEST        MT0010  SETUP BYTE ADDRESSING TEST
                                ;*********************************************************************
3914 020112  012737  000010  002260          MOV     #10,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3915 020120  012737  000004  002276          MOV     #4,PCBUMP               ;TRAPS ADD 4 TO PC
3916 020126  013704  002362                  MOV     TESTADD,R4
3917 020132  012737  026372  002254          MOV     #MTP010,SUPDOADD
3918 020140  004737  025322                  CALL    SUPD03                  ;DO IT IN SUPERVISOR MODE
3919 020144  000207                          RETURN
```

```
3922 020146                          MT0011: SUBTST  <<MT0011        SETUP CREATE SINGLE BIT ERROR TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        MT0011  SETUP CREATE SINGLE BIT ERROR TEST
                                     ;*****************************************************************************
3923 020146                                  IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
3924 020162                                    IF $PASS NE #0 THEN $RETURN
3925 020172                                  END; OF IF ACTFLAG
3926 020172  012737  000011  002260          MOV     #11,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3927 020200  012737  026500  002254          MOV     #MTP011,SUPDOADD
3928 020206  004737  025322                  CALL    SUPDO3                  ;DO IT IN SUPERVISOR MODE
3929 020212  000207                          RETURN
3930 020214                          MT0012: SUBTST  <<MT0012        SETUP WRITE BYTE CLEARS SBE TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        MT0012  SETUP WRITE BYTE CLEARS SBE TEST
                                     ;*****************************************************************************
3931 020214                                  IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
3932 020230                                    IF $PASS NE #0 THEN $RETURN
3933 020240                                  END; OF IF ACTFLAG
3934 020240  012737  000012  002260          MOV     #12,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3935 020246  013700  002102                  MOV     BANKINDEX,R0
3936 020252                                  IF #BIT12 SET.IN CONFIG+2(R0)
3937 020262  012705  040000                    MOV   #40000,R5
3938 020266                                  ELSE
3939 020270  012705  000002                    MOV   #2,R5
3940 020274                                  END; OF IF #BIT12
3941 020274  012737  027244  002254          MOV     #MTP012,SUPDOADD
3942 020302  004737  025322                  CALL    SUPDO3                  ;DO IT IN SUPERVISOR MODE
3943 020306  000207                          RETURN
3944 020310                          MT0013: SUBTST  <<MT0013        SETUP CREATE DOUBLE BIT ERROR TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        MT0013  SETUP CREATE DOUBLE BIT ERROR TEST
                                     ;*****************************************************************************
3945 020310                                  IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
3946 020324                                    IF $PASS NE #0 THEN $RETURN
3947 020334                                  END; OF IF ACTFLAG
3948 020334  012737  000013  002260          MOV     #13,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3949 020342  012737  027632  002254          MOV     #MTP013,SUPDOADD
3950 020350  012737  000003  002074          MOV     #3,NOPAR                ;INDICATE PARITY ACRION
3951 020356  004737  025322                  CALL    SUPDO3                  ;DO IT IN SUPERVISOR MODE
3952 020362  000207                          RETURN
3953 020364                          MT0014: SUBTST  <<MT0014        SETUP WRITE INHIBIT DURING DATIP WITH DBE>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        MT0014  SETUP WRITE INHIBIT DURING DATIP WITH DBE
                                     ;*****************************************************************************
3954 020364                                  IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
3955 020400                                    IF $PASS NE #0 THEN $RETURN
3956 020410                                  END; OF IF ACTFLAG
3957 020410                                  IF KFLAG IS FALSE THEN $RETURN
3958 020420  012737  000014  002260          MOV     #14,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3959 020426  012737  030346  002254          MOV     #MTP014,SUPDOADD
3960 020434  004737  025322                  CALL    SUPDO3                  ;DO IT IN SUPERVISOR MODE
3961 020440  000207                          RETURN
```

```
3964 020442                                MT0015: SUBTST  <<MT0015        SETUP WRITE INHIBIT OF BYTE WITH DBE>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       MT0015  SETUP WRITE INHIBIT OF BYTE WITH DBE
                                           ;*****************************************************************************
3965 020442                                        IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
3966 020456                                          IF $PASS NE #0 THEN $RETURN
3967 020466                                        END ;OF IF ACTFLAG
3968 020466  012737  000015  002260               MOV     #15,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3969 020474  012737  031130  002254               MOV     #MTP015,SUPDOADD
3970 020502  004737  025322                       CALL    SUPD03                  ;DO IT IN SUPERVISOR MODE
3971 020506  000207                              RETURN
3972 020510                                MT0016: SUBTST  <<MT0016        SETUP WRITE INHIBIT OF WORD WITH DBE>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       MT0016  SETUP WRITE INHIBIT OF WORD WITH DBE
                                           ;*****************************************************************************
3973 020510                                        IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
3974 020524                                          IF $PASS NE #0 THEN $RETURN
3975 020534                                        END ;OF IF ACTFLAG
3976 020534  012737  000016  002260               MOV     #16,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3977 020542  012737  031674  002254               MOV     #MTP016,SUPDOADD
3978 020550  004737  025322                       CALL    SUPD03                  ;DO IT IN SUPERVISOR MODE
3979 020554  000207                              RETURN
3980 020556                                MT0017: SUBTST  <<MT0017        SETUP HOLDING 1'S & 0'S>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       MT0017  SETUP HOLDING 1'S & 0'S
                                           ;*****************************************************************************
3981 020556  012737  000017  002260               MOV     #17,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3982 020564  012737  032456  002254               MOV     #MTP017,SUPDOADD
3983 020572  004737  025322                       CALL    SUPD03                  ;DO IT IN SUPERVISOR MODE
3984 020576  000207                              RETURN
```

```
3987 020600                                    MT0020: SUBTST  <<MT0020        SETUP MARCHING 0'S & 1'S IN CHECKBITS TEST>>
                                                       ;*****************************************************************************
                                                       ;*SUBTEST       MT0020  SETUP MARCHING 0'S & 1'S IN CHECKBITS TEST
                                                       ;*****************************************************************************
3988 020600                                                    IF ACTFLAG IS TRUE OR ACTFLAG IS TRUE
3989 020614                                                      IF $PASS NE #0 THEN $RETURN
3990 020624                                                    END ;OF IF ACTFLAG
3991 020624  012737  000020  002260                           MOV     #20,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
3992 020632  012737  000003  002074                           MOV     #3,NOPAR                ;INDICATE PARITY ACTION
3993 020640  005001                                           CLR     R1                      ;CLEAR LOOP COUNTER
3994 020642  005004                                           CLR     R4                      ;CLEAR INTERLEAVE ODD/EVEN FLAG
3995 020644  012700  060000                                   MOV     #FIRST,R0
3996 020650  013702  002102                                   MOV     BANKINDEX,R2            ;SET BANK INDEX
3997 020654                                    MTL020: IF INTFLAG IS FALSE
3998 020662                                              BEGIN MTB020
3999 020662                                              IF NO22BIT IS TRUE
4000 020670                                                IF LASTBANK EQ BANK
4001 020700  012737  140000  002362                           MOV       #140000,TESTADD ;SET UP 12K NON-INTERLEAVED VIRT ADDR
4002 020706  012705  140002                                   MOV       #140002,R5
4003 020712                                                   LEAVE     MTB020
4004 020714                                                IF; OF IF LASTBANK
4005 020714                                              END; OF IF NO22BIT
4006 020714  012737  160000  002362                       MOV   #LAST+2,TESTADD
4007 020722  012705  160002                               MOV   #LAST+4,R5
4008 020726                                               END   MTB020
4009 020726  010537  002364                               MOV   R5,TESTADD+2              ;SET UP NON-INTERLEAVED VIRT. ADDR.
4010 020732                                            ELSE
4011 020734  005737  002312                               TST   SKIPMK                    ;IS THIS BANK IN SKIP RANGE?
4012 020740  001401                                       BEQ   1$                        ;BANK IS OUT OF RANGE - DO TEST
4013 020742  000207                                       RETURN                          ;LEAVE TEST-BANK'S ALREADY TESTED
4014 020744  012737  120000  002362  1$:                  MOV   #120000,TESTADD           ;SET UP 1ST INTERLEAVED VIRT. ADDR.
4015 020752  012705  160000                               MOV   #LAST+2,R5                ;SET UP END OF BANK FLAG
4016 020756  010537  002364                               MOV   R5,TESTADD+2              ;SET UP 2ND INT'L. VIRT  ADDR.
4017 020762  005237  002232                               INC   SPLTCSR                   ;FLAG THE MAPPING ROUTINE FOR INTERLEAVING
4018 020766  005201                                       INC   R1                        ;SET LOOP COUNTER FOR INTERLEAVING
4019 020770  005204                                       INC   R4                        ;SET ODD/EVEN FLAG
4020 020772                                            END; OF IF INTFLAG
4021 020772  016203  002624                            MOV     CONFIG(R2),R3           ;SET UP CSR NUMBER
4022 020776                                            IF R4 EQ #2                     ;IF THE SECOND TIME AROUND
4023 021004  060437  002362                              ADD   R4,TESTADD
4024 021010  060437  002364                              ADD   R4,TESTADD+2            ;TEST THE A1 ASSERTED ADDRESSES
4025 021014  060400                                      ADD   R4,R0
4026 021016  060405                                      ADD   R4,R5
4027 021020  072327  177775                              ASH   #-3,R3                  ;MOVE INTERLEAVED CSR NUMBER
4028 021024                                            ELSE
4029 021026  006303                                      ASL   R3                      ;MOVE CSR NUMBER
4030 021030                                            END; IF R4
4031 021030  000303                                    SWAB    R3
4032 021032  042703  177741                            BIC     #^C36,R3
4033 021036  010337  002146                            MOV     R3,CSRNO              ;MOVE R3 INTO CSR NUMBER
4034 021042                                            IF #SW0 SET.IN @SWR
4035 021052  104506                                      ENASBE                      ;TRAP ON SINGLE BIT ERRORS
4036 021054                                            ELSE
4037 021056  104472                                      ECCINIT                     ;TRAP ON UNCORRECTABLE ERRORS
4038 021060                                            END; OF IF #SW0
4039 021060                                            PUSH    R2,R4
4040 021064                                            FOR MTV020 := #0 TO R1
```

```
4041 021070                                    PUSH  R1
4042 021072  005002                            CLR   R2                        ;PATTERN TO WRITE INTO BANK
4043 021074  004737  035156                    CALL  BACKGND                   ;SET UP ZEROS IN BANK
4044 021100                                    IF NO22BIT IS TRUE AND MTV020 EQ #1 AND BANK EQ #3
4045 021126  162737  020000  002362              SUB #20000,TESTADD            ;SET UP 12K INTERLEAVED BANK
4046 021134  162705  020000                      SUB #20000,R5
4047 021140  010537  002364                      MOV R5,TESTADD+2
4048 021144                                    END; OF IF NO22BIT
4049 021144  004737  021220                    CALL  MT020Z                    ;START TEST
4050 021150  005237  002232                    INC   SPLTCSR                   ;UPDATE INTERLEAVED MAPPING FLAG
4051 021154                                    POP   R1
4052 021156                                  END; OF FOR MTV020
4053 021170                                  POP     R4,R2
4054 021174  005001                          CLR     R1                        ;RESET LOOP FLAG
4055 021176  005037  002232                  CLR     SPLTCSR                   ;RESET INTERLEAVED MAP FLAG
4056 021202  022704  000001                  CMP     #1,R4                     ;ODD/EVEN FLAG SET?
4057 021206  001622                          BEQ     MTL020                    ;BRANCH IF TRUE
4058 021210  005037  002074                  CLR     NOPAR                     ;INDICATE PARITY ACTION
4059 021214  000207                          RETURN
4060 021216  000000              MTV020:     0                                 ;VARIABLE FOR PAT 20
```

```
4062 021220  012702  000004          MT020Z: MOV     #4,R2                   ;SET UP WORD INCR/DECR AMOUNT
4063 021224  012700  060000                  MOV     #FIRST,R0
4064 021230  013701  002362                  MOV     TESTADD,R1
4065 021234  013704  002364                  MOV     TESTADD+2,R4
4066 021240  012703  100000                  MOV     #BIT15,R3
4067 021244                                  IF #SW11 SET.IN @SWR OR QVFLAG IS TRUE
4068 021262                                    GOTO  MT020Y
4069 021264                                  END ;OF IF #SW11
4070 021264  005737  002424                  TST     NO22BIT                 ;IS THIS AN 11/44?
4071 021270  001411                          BEQ     1$                      ;BRANCH IF IT IS
4072 021272  012737  032534  002254          MOV     #MTPA20,SUPDOADD
4073 021300  012737  032550  002264          MOV     #MTPA20+14,PARTHERE     ;VECTOR FOR TRAPS
4074 021306  004737  025322                  CALL    SUPD03
4075 021312  000410                          BR      2$
4076 021314                          1$:     BMOV    MTPA20
4077 021322  012737  177654  002264          MOV     #UIPAR6,PARTHERE        ;VECTOR FOR TRAPS
4078 021330  004737  025144                  CALL    SUPD01
4079 021334  005737  002424          2$:     TST     NO22BIT                 ;IS THIS AN 11/44?
4080 021340  001411                          BEQ     4$                      ;BRANCH IF IT IS
4081 021342  012737  032564  002254          MOV     #MTPB20,SUPDOADD
4082 021350  012737  032574  002264          MOV     #MTPB20+10,PARTHERE     ;VECTOR FOR TRAPS
4083 021356  004737  025336                  CALL    SUPD04
4084 021362  000410                          BR      MT020Y
4085 021364                          4$:     BMOV    MTPB20
4086 021372  012737  177650  002264          MOV     #UIPAR4,PARTHERE        ;VECTOR FOR TRAPS
4087 021400  004737  025160                  CALL    SUPD02
4088 021404  005737  002134          MT020Y: TST     INTFLAG                 ;ARE WE INTERLEAVED?
4089 021410  001405                          BEQ     7$                      ;BRANCH IF NOT INTERLEAVED
4090 021412  162701  040000                  SUB     #40000,R1               ;RESET FIRST WORD TO BEGINNING OF BANK
4091 021416  162704  040000                  SUB     #40000,R4               ;RESET SECOND WORD TO BEGINNING OF BANK
4092 021422  000404                          BR      8$
4093 021424  012701  060000          7$:     MOV     #FIRST,R1               ;RESET FIRST WORD TO BEGINNING OF BANK
4094 021430  012704  060002                  MOV     #FIRST+2,R4             ;RESET SECOND WORD TO BEGINNING OF BANK
4095 021434  005737  002424          8$:     TST     NO22BIT                 ;IS THIS AN 11/44?
4096 021440  001411                          BEQ     1$                      ;BRANCH IF IT IS
4097 021442  012737  032614  002254          MOV     #MTPC20,SUPDOADD
4098 021450  012737  032624  002264          MOV     #MTPC20+10,PARTHERE     ;VECTOR FOR TRAPS
4099 021456  004737  025322                  CALL    SUPD03
4100 021462  000410                          BR      2$
4101 021464                          1$:     BMOV    MTPC20
4102 021472  012737  177650  002264          MOV     #UIPAR4,PARTHERE        ;VECTOR FOR TRAPS
4103 021500  004737  025144                  CALL    SUPD01
4104 021504  005737  002424          2$:     TST     NO22BIT                 ;IS THIS AN 11/44?
4105 021510  001411                          BEQ     3$                      ;BRANCH IF IT IS
4106 021512  012737  032644  002254          MOV     #MTPD20,SUPDOADD
4107 021520  012737  032660  002264          MOV     #MTPD20+14,PARTHERE     ;VECTOR FOR TRAPS
4108 021526  004737  025336                  CALL    SUPD04
4109 021532  000410                          BR      4$
4110 021534                          3$:     BMOV    MTPD20
4111 021542  012737  177654  002264          MOV     #UIPAR6,PARTHERE        ;VECTOR FOR TRAPS
4112 021550  004737  025160                  CALL    SUPD02
4113 021554  005737  002424          4$:     TST     NO22BIT                 ;IS THIS AN 11/44?
4114 021560  001411                          BEQ     5$                      ;BRANCH IF IT IS
4115 021562  012737  032674  002254          MOV     #MTPE20,SUPDOADD
4116 021570  012737  032704  002264          MOV     #MTPE20+10,PARTHERE     ;VECTOR FOR TRAPS
4117 021576  004737  025336                  CALL    SUPD04
4118 021602  000410                          BR      6$
```

```
 4119 021604                                5$:     BMOV    MTPE20
 4120 021612  012737  177650  002264               MOV     #UIPAR4,PARTHERE        ;VECTOR FOR TRAPS
 4121 021620  004737  025160                       CALL    SUPD02
 4122 021624  104503                        6$:     CLR1CSR                         ;CLEAR 1 SELECTED CSR
 4123 021626  000207                                RETURN
```

```
                                        MT0021: SUBTST  <<MT0021        SETUP MARCHING 0'S & 1'S TEST>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       MT0021  SETUP MARCHING 0'S & 1'S TEST
                                        ;*****************************************************************************
                                                SET NOSCOPE
4128 021636  012737  000021  002260             MOV     #21,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4129 021644  013702  002572                     MOV     BAKPAT,R2
4130 021650  004737  035156                     CALL    BACKGND
4131 021654  010203                             MOV     R2,R3
4132 021656  000303                             SWAB    R3
4133 021660  012701  160000                     MOV     #LAST+2,R1
4134 021664  010105                             MOV     R1,R5
4135 021666  012704  060000                     MOV     #FIRST,R4
4136 021672  005737  002424                     TST     NO22BIT         ;IS THIS AN 11/44?
4137 021676  001435                             BEQ     1$              ;BRANCH IF IT IS
4138 021700  022737  000007  002100             CMP     #7,BANK
4139 021706  001003                             BNE     3$
4140 021710  012701  140000                     MOV     #140000,R1
4141 021714  010105                             MOV     R1,R5
4142 021716  012737  032720  002254  3$:        MOV     #MTPA21,SUPDOADD
4143 021724  004737  025322                     CALL    SUPDO3
4144 021730  012737  032750  002254             MOV     #MTPB21,SUPDOADD
4145 021736  004737  025336                     CALL    SUPDO4
4146 021742  010401                             MOV     R4,R1
4147 021744  012737  033004  002254             MOV     #MTPC21,SUPDOADD
4148 021752  004737  025336                     CALL    SUPDO4
4149 021756  012737  033040  002254             MOV     #MTPD21,SUPDOADD
4150 021764  004737  025336                     CALL    SUPDO4
4151 021770  000425                             BR      2$
4152 021772                             1$:     BMOV    MTPA21
4153 022000  004737  025144                     CALL    SUPDO1
4154
4155 022004                                     BMOV    MTPB21
4156 022012  004737  025160                     CALL    SUPDO2
4157
4158 022016  010401                             MOV     R4,R1
4159 022020                                     BMOV    MTPC21
4160 022026  004737  025160                     CALL    SUPDO2
4161
4162 022032                                     BMOV    MTPD21
4163 022040  004737  025160                     CALL    SUPDO2
4164 022044  005037  002410             2$:     CLR     NOSCOPE
4165 022050  000207                             RETURN
```

```
4167 022052                          MT0022: SUBTST  <<MT0022        SETUP REFRESH & SHIFTING DIAGONAL TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MT0022  SETUP REFRESH & SHIFTING DIAGONAL TEST
                                     ;*****************************************************************************
4168 022052  004737  025110                     CALL    KAMITEST                ;CHECK FOR KAMIKAZE MODE
4169 022056                                     ON.ERROR THEN $RETURN           ;IF NOT IN KAMIKAZE MODE RETURN
4170 022062  012737  000022  002260             MOV     #22,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4171 022070  012737  033070  002254             MOV     #MTP022,SUPDOADD
4172 022076  004737  025322                     CALL    SUPDO3                  ;DO IT IN SUPERVISOR MODE
4173 022102  000207                            RETURN
4174
4175 022104                          MT0023: SUBTST  <<MT0023        SHIFTING DIAGONAL TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MT0023  SHIFTING DIAGONAL TEST
                                     ;*****************************************************************************
4176 022104  004737  025110                     CALL    KAMITEST                ;CHECK FOR KAMIKAZE MODE
4177 022110                                     ON.ERROR THEN $RETURN           ;IF NOT IN KAMIKAZE MODE RETURN
4178 022114  012737  000023  002260             MOV     #23,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4179 022122  012737  033070  002254             MOV     #MTP022,SUPDOADD
4180 022130                                     SET     DIAGFLAG                ;IDENTIFY DIAGONAL TEST TO MTP022
4181 022136  004737  025322                     CALL    SUPDO3                  ;DO IT IN SUPERVISOR MODE
4182 022142  005037  002002                     CLR     DIAGFLAG
4183 022146  000207                            RETURN
```

```
                                        MT0024: SUBTST  <<MT0024        SETUP FAST GALLOPING PATTERN TEST>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       MT0024  SETUP FAST GALLOPING PATTERN TEST
                                        ;*****************************************************************************
                                        ;
                                                CALL    KAMITEST                ;CHECK FOR KAMIKAZE MODE
                                                ON.ERROR THEN $RETURN           ;IF NOT IN KAMIKAZE MODE RETURN
                                                SET     NOSCOPE
                                                MOV     #24,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
                                                MOV     BAKPAT,R2
                                                CALL    BACKGND
                                                MOV     R2,R3
                                                MOV     R3,R4
                                                SWAB    R4
                                                MOV     #FIRST,R1
                                                MOV     #LAST,R5
                                                TST     NO22BIT
                                                BEQ     1$
                                                CMP     #7,BANK
                                                BNE     3$
                                                MOV     #137776,R5
                                        3$:     SAVREG
                                                MOV     #MTPB24,SUPDOADD
                                                BR      2$
                                        1$:     CMP     #177,LASTBANK
                                                BNE     4$
                                                MOV     #137776,R5
                                        4$:     SAVREG
                                                BMOV    MTPA24
                                                BMOV    MTPB24,SDPAR0,8.
                                                BMOV    MTPC24,KDPAR0,8.
                                                MOV     #SDPAR0,SUPDOADD
                                                MOV     #SDPAR0,UDPAR7          ;SET UP PAR LINKS
                                                MOV     #KDPAR0,SDPAR5
                                                MOV     #UDPAR0,KDPAR6
                                        2$:     CALL    SUPD04
                                        ;
                                        ;DO IT AGAIN FOR COMPLEMENT DATA
                                                RESREG
                                                SWAB    R2
                                                SWAB    R3
                                                CALL    SUPD04
                                                CLR     NOSCOPE
                                                RETURN
                                        MT0025: SUBTST  <<MT0025        SETUP INTERRUPT ENABLE TEST>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       MT0025  SETUP INTERRUPT ENABLE TEST
                                        ;*****************************************************************************
                                        ;
                                                IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
                                                  IF $PASS NE #0 THEN $RETURN
                                                END ;OF IF ACTFLAG
                                                MOV     #25,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
                                                MOV     #MTP025,SUPDOADD
                                                CALL    SUPD03                  ;DO IT IN SUPERVISOR MODE
                                                RETURN
```

```
4235 022450                          MT0026: SUBTST  <<MT0026        SETUP RANDOM DATA TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MT0026  SETUP RANDOM DATA TEST
                                     ;*****************************************************************************
4236 022450  012737  000026  002260          MOV     #26,REALPAT
4237 022456  005037  002276                  CLR     PCBUMP                  ;TRAPS DO NOT ADD TO THE PC
4238 022462  013703  002544                  MOV     SEEDLO,R3               ;INITIALIZE RANDOM NUMBERS
4239 022466  013702  002542                  MOV     SEEDHI,R2
4240 022472  010305                         MOV     R3,R5
4241 022474  010204                         MOV     R2,R4
4242 022476  012701  060000                  MOV     #FIRST,R1
4243 022502  012700  020000                  MOV     #SIZE/2,R0
4244 022506  005737  002424                  TST     NO22BIT                 ;DO WE HAVE AN 11/44?
4245 022512  001433                         BEQ     1$                      ;BRANCH IF WE DO
4246 022514  022737  000007  002100          CMP     #7,BANK
4247 022522  001002                         BNE     3$
4248 022524  012700  014000                  MOV     #14000,R0
4249 022530  104415                3$:      SAVREG
4250 022532  012737  034310  034410          MOV     #MTPA26+4,MTPD26+14
4251 022540  012737  034304  002254          MOV     #MTPA26,SUPDOADD
4252 022546  004737  025322                  CALL    SUPDO3
4253 022552  005037  034334                  CLR     RANODD                  ;FOR ERROR REPORTING
4254 022556  012737  034324  034410          MOV     #MTPB26+4,MTPD26+14     ;SET UP NEXT LINK
4255 022564  012737  034320  002254          MOV     #MTPB26,SUPDOADD
4256 022572  104416                         RESREG
4257 022574  004737  025322                  CALL    SUPDO3
4258 022600  000452                         BR      2$
4259 022602  022737  000177  002100  1$:     CMP     #177,BANK
4260 022610  001002                         BNE     4$
4261 022612  012700  014000                  MOV     #14000,R0
4262 022616  104415                4$:      SAVREG
4263 022620                                  BMOV    MTPA26                  ;WRITE ROUTINE TO FAST MEMORY
4264 022626                                  BMOV    MTPC26,KDPAR0,8.        ;RANDOM SUBPROGRAM TO FAST MEMORY
4265 022640  012737  000730  172376          MOV     #730,KDPAR7             ;WRITES 'BR .-116'' IN (BR SDPAR0)
4266 022646                                  BMOV    MTPD26,SDPAR0,8.        ;RANDOM SUBSUBPROGRAM TO FAST MEMORY
4267 022660  012737  172360  177642          MOV     #KDPAR0,UIPAR1
4268 022666  012737  177644  172274          MOV     #UIPAR2,SDPAR6
4269 022674  004737  025144                  CALL    SUPDO1                  ;WRITE RANDOM DATA
4270 022700  005037  034334                  CLR     RANODD                  ;FOR ERROR REPORTING
4271 022704                                  BMOV    MTPB26                  ;READ ROUTINE TO FAST MEMORY
4272 022712  012737  172360  177642          MOV     #KDPAR0,UIPAR1          ;SET UP PAR LINK
4273 022720  104416                         RESREG
4274 022722  004737  025144                  CALL    SUPDO1                  ;READ RANDOM DATA
4275 022726  010337  002544          2$:     MOV     R3,SEEDLO               ;UPDATE FOR NEW RANDOM NUMBERS
4276 022732  010237  002542                  MOV     R2,SEEDHI
4277 022736  000207                         RETURN
```

```
 4280 022740                                MT0027: SUBTST  <<MT0027        UNIQUE BANK TEST>>
                                            ;*****************************************************************************
                                            ;*SUBTEST       MT0027  UNIQUE BANK TEST
                                            ;*****************************************************************************
 4281                                               ;MAKE SURE THAT EACH BANK CAN HAVE UNIQUE DATA
 4282                                               ;WRITE AND READ THE BANK NUMBER IN EACH BANK (EXCEPT WHERE THE PROGRAM IS)
 4283 022740  012737  000027  002260          MOV     #27,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
 4284 022746  104502                          CLRCSR                  ;CLEAR CSRS
 4285 022750  005737  002424                  TST     NO22BIT                 ;IS THIS AN 11/44?
 4286 022754  001404                          BEQ     1$                      ;BRANCH IF TRUE
 4287 022756  012737  025322  002472          MOV     #SUPD03,LINK1           ;SET UP LINK
 4288 022764  000414                          BR      STAR27                  ;BRANCH TO RUN
 4289 022766                                1$:     BMOV    MTP034
 4290 022774  012737  177646  002254    WARN7:  MOV     #UIPAR3,SUPDOADD
 4291 023002  012737  025144  002472          MOV     #SUPD01,LINK1           ;SET UP LINK
 4292 023010                                  SET NOFSMODE
 4293 023016                                STAR27: FOR I := #1 TO #2
 4294 023024                                    FOR BANK := #0 TO LASTBANK
 4295 023030  004737  042700                      CALL EXBANK
 4296 023034                                      IF ACFLAG IS TRUE AND RRFLAG IS FALSE
 4297 023050  104511                                INVALIDATE              ;INVALIDATE BACKGROUND PATTERN ON 'BANK'
 4298 023052                                        LET R2 := BANK
 4299 023056  012700  060000                        MOV     #FIRST,R0
 4300 023062  010004                                MOV     R0,R4
 4301 023064  012701  040000                        MOV     #SIZE,R1
 4302 023070  010103                                MOV     R1,R3
 4303 023072                                        IF I EQ #1
 4304 023102  005737  002424                          TST     NO22BIT
 4305 023106  001403                                  BEQ     2$
 4306 023110  012737  034630  002254                  MOV     #MTP034,SUPDOADD
 4307 023116  004777  157350                    2$: CALL    @LINK1
 4308 023122                                        END ;OF IF
 4309 023122                                        IF I EQ #2
 4310 023132  005737  002424                          TST     NO22BIT
 4311 023136  001403                                  BEQ     3$
 4312 023140  012737  034636  002254                  MOV     #MTP034+6,SUPDOADD
 4313 023146  004737  025322                    3$: CALL    SUPD03
 4314 023152                                        END ;OF IF
 4315 023152                                    END ;OF IF
 4316 023152                                END ;OF FOR BANK
 4317 023166                            END ;OF FOR I
 4318 023202                            IF FS7FLAG IS TRUE
 4319 023210  005037  002376              CLR   NOFSMODE
 4320 023214  000207                      RETURN
 4321 023216                            END ;OF IF FS7FLAG
 4322 023216                            FOR I := #1 TO #2
 4323 023224                              FOR BANK := LASTBANK DOWNTO #0
 4324 023232  004737  042700                CALL EXBANK
 4325 023236                                IF ACFLAG IS TRUE AND RRFLAG IS FALSE
 4326 023252                                  LET R2 := BANK
 4327 023256  005102                          COM     R2
 4328 023260  012700  060000                  MOV     #FIRST,R0
 4329 023264  010004                          MOV     R0,R4
 4330 023266  012701  040000                  MOV     #SIZE,R1
 4331 023272  010103                          MOV     R1,R3
 4332 023274                                  IF I EQ #1
 4333 023304  005737  002424                    TST     NO22BIT
```

```
 4334 023310  001403                              BEQ     4$
 4335 023312  012737  034630  002254              MOV     #MTP034,SUPDOADD
 4336 023320  004777  157146                    4$: CALL    @LINK1
 4337 023324                                      END ;OF IF
 4338 023324                                      IF I EQ #2
 4339 023334  005737  002424                        TST     NO22BIT
 4340 023340  001403                              BEQ     5$
 4341 023342  012737  034636  002254              MOV     #MTP034+6,SUPDOADD
 4342 023350  004737  025322                    5$: CALL    SUPDO3
 4343 023354                                      END ;OF IF
 4344 023354                                    END ;OF IF
 4345 023354                                  END ;OF FOR BANK
 4346 023370                                END ;OF FOR I
 4347 023404  005037  002376                  CLR     NOFSMODE
 4348 023410  000207                          RETURN
```

```
4351 023412                                MT0030: SUBTST  <<MT0030        SETUP FLUSH OUT DBE'S TEST>>
                                                   ;*****************************************************************************
                                                   ;*SUBTEST       MT0030  SETUP FLUSH OUT DBE'S TEST
                                                   ;*****************************************************************************
4352 023412  005037  002256                               CLR     PASFLG
4353 023416                                               SET     FULLREL
4354 023424  012737  000030  002260  MTA030:  MOV     #30,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4355 023432  012737  000001  002074           MOV     #1,NOPAR                ;INDICATE COUNT PARITY ERRORS
4356 023440  005737  002424                    TST     NO22BIT
4357 023444  001007                           BNE     4$
4358 023446                                    BMOV    MTP030
4359 023454  012737  025144  002472           MOV     #SUPD01,LINK1
4360 023462  000406                           BR      1$
4361 023464  012737  025322  002472  4$:      MOV     #SUPD03,LINK1
4362 023472  012737  034412  002254           MOV     #MTP030,SUPDOADD
4363 023500  104470                    1$:    ECCDIS                          ;DISABLE ERROR CORRECTION
4364 023502                                    SET     NOFSMODE,NOSCOPE
4365 023516                                    FOR BANK := #0 TO LASTBANK
4366 023522  004737  042700                      CALL  EXBANK
4367 023526                                      IF MKFLAG IS TRUE
4368 023534                                        IF ACFLAG IS TRUE AND RRFLAG IS FALSE
4369 023550  012701  040000                          MOV       #SIZE,R1
4370 023554  012700  060000                          MOV       #FIRST,R0
4371 023560  004777  156706                          CALL      @LINK1
4372 023564                                        END ;OF IF ACFLAG
4373 023564                                      END ;OF IF MKFLAG
4374 023564                                    END ;OF FOR
4375 023600                                    IF PASFLG IS FALSE
4376 023606                                      SET PASFLG
4377 023614  104502                             CLRCSR                          ;CLEAR CSRS
4378 023616  004737  041154                     CALL  RELOCATE
4379 023622                                     ON.ERROR
4380 023624  104472                               ECCINIT                 ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
4381 023626                                       CLEAR       NOFSMODE,NOSCOPE,FULLREL
4382 023642  000207                               RETURN
4383 023644                                     END ;OF ON.ERROR
4384 023644  013737  002270  002100             MOV   NEWBANK,BANK
4385 023652  004737  042700                     CALL  EXBANK
4386 023656  004737  023424                     CALL  MTA030
4387 023662  104472                             ECCINIT                 ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
4388 023664  004737  042042                     CALL  UNRELOCATE
4389 023670  000207                             RETURN
4390 023672                                    END ;OF IF PASFLG
4391 023672  104472                           ECCINIT           ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
4392 023674                                    CLEAR   NOFSMODE,NOSCOPE,FULLREL
4393 023710  000207                           RETURN
```

```
4396 023712                                  MT0031: SUBTST  <<MT0031        SETUP SOB-A-LONG TEST>>
                                             ;*****************************************************************************
                                             ;*SUBTEST       MT0031  SETUP SOB-A-LONG TEST
                                             ;*****************************************************************************
4397 023712  004737  025110                          CALL    KAMITEST                ;CHECK FOR KAMIKAZE MODE
4398 023716                                          ON.ERROR THEN $RETURN           ;IF NOT IN KAMIKAZE MODE RETURN
4399 023722                                          SET     NOSCOPE
4400 023730  012737  000031  002260                  MOV     #31,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4401 023736  005037  002074                          CLR     NOPAR                   ;SETUP PARITY ACTION
4402 023742                                          MAP     BANK                    ;MAP FIRST SO BLOCK MOVE WORKS
4403 023756                                          TESTAREA                        ;ENTER TEST MODE
4404 023764                                          BMOV    MTP031,FIRST,SOBLENGTH/2
4405 023776  104417                                  KERNEL                          ;ENTER KERNEL MODE
4406 024000  013702  002532                          MOV     SOBK,R2
4407 024004  010200                                  MOV     R2,R0
4408 024006  012701  100776                          MOV     #100776,R1              ;COMPLEMENT OF INSTRUCTION 'SOB R0,DOT''
4409 024012  012705  060056                          MOV     #FIRST+SOBLENGTH,R5
4410 024016  012737  060002  002254                  MOV     #FIRST+2,SUPDOADD
4411 024024  012737  160000  002472                  MOV     #LAST+2,LINK1
4412 024032  005737  002426                          TST     NOSUPER
4413 024036  001005                                  BNE     1$
4414 024040  023737  172252  172254                  CMP     SIPAR5,SIPAR6
4415 024046  001405                                  BEQ     2$
4416 024050  000407                                  BR      3$
4417 024052  023737  177652  177654  1$:             CMP     UIPAR5,UIPAR6
4418 024060  001003                                  BNE     3$
4419 024062  012737  140000  002472  2$:             MOV     #140000,LINK1
4420 024070  004737  025336          3$:             CALL    SUPDO4
4421 024074  005037  002410                          CLR     NOSCOPE
4422 024100  000207                                  RETURN
```

```
4425 024102                                MT0032: SUBTST  <<MT0032        SETUP WRITE RECOVERY TEST>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       MT0032  SETUP WRITE RECOVERY TEST
                                           ;*****************************************************************************
4426 024102  004737  025110                        CALL    KAMITEST                ;CHECK FOR KAMIKAZE MODE
4427 024106                                        ON.ERROR THEN $RETURN           ;IF NOT IN KAMIKAZE MODE RETURN
4428 024112                                        SET     NOSCOPE
4429 024120  012737  000032  002260               MOV     #32,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
4430 024126  005037  002074                       CLR     NOPAR                   ;SETUP PARITY ACTION
4431 024132                                        MAP     BANK                    ;MAP FIRST SO THAT THE BLOCK MOVE WORKS
4432 024146  012700  010247                       MOV     #10247,R0               ;OP CODE OF INSTRUCTION 'MOV    R2,-(PC)''
4433 024152  012701  177667                       MOV     #177667,R1              ;OP CODE OF COMPLEMENT OF INSTRUCTION ''JMP (R0)''
4434 024156  012702  020000                       MOV     #SIZE/2,R2              ;USED FOR 1/2 BANK LOOP
4435 024162  010237  002472                       MOV     R2,LINK1
4436 024166  012703  060000                       MOV     #FIRST,R3
4437 024172  012704  160000                       MOV     #LAST+2,R4
4438 024176  005037  002474                       CLR     LINK2
4439 024202  005737  002426                       TST     NOSUPER
4440 024206  001005                              BNE     1$
4441 024210  023737  172252  172254               CMP     SIPAR5,SIPAR6
4442 024216  001405                              BEQ     2$
4443 024220  000415                              BR      3$
4444 024222  023737  177652  177654  1$:          CMP     UIPAR5,UIPAR6
4445 024230  001011                              BNE     3$
4446 024232  012704  140000          2$:          MOV     #140000,R4
4447 024236  012702  014000                       MOV     #14000,R2
4448 024242  010237  002472                       MOV     R2,LINK1
4449 024246  012737  000001  002474               MOV     #1,LINK2
4450
4451 024254                        3$:          TESTAREA                        ;ENTER TEST MODE
4452                                            ;MOVE TEST TO MEMORY UNDER TEST
4453 024262  010023                4$:          MOV     R0,(R3)+
4454 024264  010144                             MOV     R1,-(R4)
4455 024266  077203                             SOB     R2,4$
4456
4457 024270  005737  002424                      TST     NO22BIT
4458 024274  001003                             BNE     5$
4459                                            ;MOVE LAST PART OF TEST TO FASTCITY
4460 024276                                      BMOV    MTP032
4461 024304  104417                5$:          KERNEL                          ;ENTER KERNEL MODE
4462
4463 024306  012702  005141                      MOV     #5141,R2                ;OP CODE OF INSTRUCTION ''COM    -(R1)''
4464 024312  012700  024426                      MOV     #10$,R0                 ;ADDRESS TO RETURN TO IN R0
4465 024316  012701  160000                      MOV     #LAST+2,R1              ;TOP OF BANK
4466 024322  012737  060000  002254               MOV     #FIRST,SUPDOADD
4467 024330  005737  002474                      TST     LINK2
4468 024334  001402                             BEQ     6$
4469 024336  012701  140000                      MOV     #140000,R1
4470 024342  004737  025336          6$:          CALL    SUPD04
4471 024346  012703  020000                      MOV     #SIZE/2,R3
4472 024352  012705  000110                      MOV     #110,R5
4473 024356  012704  060000                      MOV     #FIRST,R4
4474 024362  005737  002474                      TST     LINK2
4475 024366  001402                             BEQ     7$
4476 024370  012703  014000                      MOV     #14000,R3
4477 024374  005737  002424          7$:          TST     NO22BIT
4478 024400  001406                             BEQ     8$
```

```
 4479 024402  012737  034500  002254          MOV     #MTP032,SUPDOADD
 4480 024410  004737  025336                  CALL    SUPD04
 4481 024414  000402                          BR      9$
 4482 024416  004737  025160          8$:     CALL    SUPD02
 4483 024422  005037  002410          9$:     CLR     NOSCOPE
 4484 024426  000207                  10$:    RETURN                  ;THIS RETURN ACTS AS A NORMAL RETURN FROM MT0032
 4485                                                                 ;ALSO A RETURN FROM THE ''CALL   SUPD04'' ABOVE
 4486
```

```
                                MT0033: SUBTST  <<MT0033        SETUP BRANCH GOBBLE TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MT0033  SETUP BRANCH GOBBLE TEST
                                ;*****************************************************************************
                                ;
 4490 024430  004737  025110            CALL    KAMITEST                ;CHECK FOR KAMIKAZE MODE
 4491 024434                            ON.ERROR THEN $RETURN           ;IF NOT IN KAMIKAZE MODE RETURN
 4492 024440                            SET     NOSCOPE
 4493 024446  012737  000033  002260    MOV     #33,REALPAT             ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
 4494 024454  005037  002074            CLR     NOPAR                   ;SETUP PARITY ACTION
 4495 024460                            MAP     BANK                    ;MAP FIRST SO THAT BLOCK MOVE WORKS
 4496
 4497 024474                            TESTAREA                        ;ENTER TEST MODE
 4498 024502                            BMOV    MTP033,FIRST,GBLENGTH/2
 4499 024514  104417                    KERNEL                          ;ENTER KERNEL MODE
 4500
 4501 024516  012705  060076            MOV     #FIRST+GBLENGTH,R5
 4502 024522  012737  060004  002254    MOV     #FIRST+4,SUPDOADD
 4503 024530  012701  060002            MOV     #FIRST+2,R1
 4504 024534  012702  060003            MOV     #FIRST+3,R2
 4505 024540  012737  160000  002472    MOV     #LAST+2,LINK1
 4506 024546  005737  002426            TST     NOSUPER
 4507 024552  001005                    BNE     1$
 4508 024554  023737  172252  172254    CMP     SIPAR5,SIPAR6
 4509 024562  001405                    BEQ     2$
 4510 024564  000407                    BR      3$
 4511 024566  023737  177652  177654  1$:       CMP     UIPAR5,UIPAR6
 4512 024574  001003                    BNE     3$
 4513 024576  012737  140000  002472  2$:       MOV     #140000,LINK1
 4514
 4515 024604  004737  025336          3$:       CALL    SUPDO4
 4516 024610  005037  002410                    CLR     NOSCOPE
 4517 024614  000207                    RETURN
 4518
 4519 024616                            MT0034: SUBTST  <<MT0034        SOFT ERROR - BACKGROUND PATTERN TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MT0034  SOFT ERROR - BACKGROUND PATTERN TEST
                                ;*****************************************************************************
 4520 024616  012737  000034  002260            MOV     #34,REALPAT
 4521 024624  012700  060000                    MOV     #FIRST,R0
 4522 024630  012701  040000                    MOV     #SIZE,R1
 4523 024634  013702  002560                    MOV     SOFTPAT,R2
 4524 024640  010103                            MOV     R1,R3
 4525 024642  013705  002102                    MOV     BANKINDEX,R5
 4526 024646  010004                            MOV     R0,R4
 4527 024650  005737  002424                    TST     NO22BIT                 ;IS THIS AN 11/44?
 4528 024654  001006                            BNE     1$                      ;BRANCH IF NOT
 4529 024656                                    BMOV    MTP034
 4530 024664  012737  177646  002254            MOV     #UIPAR3,SUPDOADD
 4531 024672                            1$:     IF #BIT13 SET.IN CONFIG+2(R5)
 4532                                      ;BACKGROUND PATTERN IS VALID
 4533 024702  005737  002424                      TST   NO22BIT
 4534 024706  001403                              BEQ   2$
 4535 024710  012737  034636  002254              MOV   #MTP034+6,SUPDOADD
 4536 024716  004737  025322          2$:         CALL  SUPDO3                  ;READ IT
 4537 024722                                    ELSE
 4538                                      ;BACKGROUND PATTERN HAS BEEN INVALIDATED
 4539 024724  005737  002424                      TST   NO22BIT
```

```
4540 024730  001406                        BEQ   3$
4541 024732  012737  034630  002254        MOV   #MTP034,SUPDOADD
4542 024740  004737  025322                CALL  SUPDO3
4543 024744  000402                        BR    4$
4544 024746  004737  025144          3$:   CALL  SUPDO1                    ;WRITE IT
4545 024752  052765  020000  002626  4$:   BIS   #BIT13,CONFIG+2(R5)       ;VALIDATE IT
4546 024760                        END ;OF IF #BIT13
4547 024760  000207                RETURN
4548
4549 024762                MT0035: SUBTST <<MT0035        SETUP WORST CASE NOISE PARITY TEST>>
                           ;*****************************************************************************
                           ;*SUBTEST       MT0035  SETUP WORST CASE NOISE PARITY TEST
                           ;*****************************************************************************
4550 024762  012737  000035  002260        MOV   #35,REALPAT             ;SET UP TEST NUMBER FOR DISPLAY
4551 024770  013703  002102                MOV   BANKINDEX,R3
4552 024774  016301  002624                MOV   CONFIG(R3),R1
4553 025000  000301                        SWAB  R1
4554 025002  042701  177760                BIC   #^C17,R1
4555 025006  006301                        ASL   R1
4556 025010  010137  002146                MOV   R1,CSRNO
4557 025014  023737  002146  002502        CMP   CSRNO,PGMCSR
4558 025022  001001                        BNE   1$
4559 025024  000207                        RETURN
4560 025026  012702  052524          1$:   MOV   #52524,R2
4561 025032  004737  035156                CALL  BACKGND                 ;WRITE BACKROUND OF ALMOST ALT. 1'S AND 0'S
4562 025036  012737  034654  002254        MOV   #MTP035,SUPDOADD
4563 025044  004737  025322                CALL  SUPDO3
4564 025050                                IF QVFLAG IS TRUE THEN $RETURN
4565 025060  005102                        COM   R2
4566 025062  004737  035156                CALL  BACKGND                 ;WRITE COMPLEMENT PATTERN INTO MUT
4567 025066  004737  025336                CALL  SUPDO4
4568 025072  000207                        RETURN
```

```
4571 025074                          MT0999: SUBTST  <<MT0999        SETUP NULL TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MT0999  SETUP NULL TEST
                                     ;*****************************************************************************
4572 025074  005037  002260                 CLR     REALPAT
4573 025100                                 SET     NULLFLAG
4574 025106  000207                         RETURN
4575
4576 025110                          KAMITEST:SUBTST <<CHECK FOR KAMIKAZE MODE>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       CHECK FOR KAMIKAZE MODE
                                     ;*****************************************************************************
4577 025110                                 IF KAMIKAZE IS TRUE OR ACTFLAG IS TRUE OR APTFLAG IS TRUE
4578 025132                                   $RETURN NOERROR               ;RUN THE TEST
4579 025136                                 ELSE
4580 025140                                   $RETURN ERROR                 ;DON'T RUN THE TEST
4581 025144                                 END ;OF IF KAMIKAZE
```

```
4584 025144                                SUPD01: SUBTST  <<SUBR  EXECUTE PATTERN IN SUPERVISOR>>
                                           ;*************************************************************************
                                           ;*SUBTEST       SUBR    EXECUTE PATTERN IN SUPERVISOR
                                           ;*************************************************************************
4585 025144                                        MAP     BANK                    ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
4586 025160  004737  054374                SUPD02: CALL    GETDIS
4587 025164                                        PUSH    $LPERR,$LPADR
4588 025174  010037  002152                        MOV     R0,SUPDR0
4589 025200  012700  002154                        MOV     #SUPDR1,R0
4590 025204  010120                                MOV     R1,(R0)+
4591 025206  010220                                MOV     R2,(R0)+
4592 025210  010320                                MOV     R3,(R0)+
4593 025212  010420                                MOV     R4,(R0)+
4594 025214  010520                                MOV     R5,(R0)+
4595 025216  010620                                MOV     SP,(R0)+
4596 025220  013700  002152                        MOV     SUPDR0,R0
4597 025224  012737  025240  002562                MOV     #TAG4$,$LPADR
4598 025232  013737  002562  002564                MOV     $LPADR,$LPERR
4599 025240  012700  002170                TAG4$:  MOV     #SUPDR6+2,R0
4600 025244  014006                                MOV     -(R0),SP
4601 025246  014005                                MOV     -(R0),R5
4602 025250  014004                                MOV     -(R0),R4
4603 025252  014003                                MOV     -(R0),R3
4604 025254  014002                                MOV     -(R0),R2
4605 025256  014001                                MOV     -(R0),R1
4606 025260  014000                                MOV     -(R0),R0
4607 025262                                        SUPERVISOR              ;ENTER SUPERVISOR MODE
4608 025270  012706  000740                        MOV     #SUPSTK,SSP
4609 025274  104424                                CACHOFF                 ;TURN CACHE OFF
4610 025276  004737  177640                        CALL    FASTCITY                ;CALL TO THE USER INSTRUCTION PAR'S
4611 025302  104423                                CACHON                  ;TURN CACHE ON
4612 025304  104417                                KERNEL                  ;ENTER KERNEL MODE
4613 025306  000004                                SCOPE
4614 025310                                        POP     $LPADR,$LPERR
4615 025320  000207                                RETURN
```

```
4618 025322                        SUPD03: MAP     BANK                    ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
4619 025336  004737  054374        SUPD04: CALL    GETDIS
4620 025342                                PUSH    $LPERR,$LPADR
4621 025352  010037  002152                MOV     R0,SUPDR0
4622 025356  012700  002154                MOV     #SUPDR1,R0
4623 025362  010120                        MOV     R1,(R0)+
4624 025364  010220                        MOV     R2,(R0)+
4625 025366  010320                        MOV     R3,(R0)+
4626 025370  010420                        MOV     R4,(R0)+
4627 025372  010520                        MOV     R5,(R0)+
4628 025374  010620                        MOV     SP,(R0)+
4629 025376  013700  002152                MOV     SUPDR0,R0
4630 025402  012737  025416  002562        MOV     #TBG4$,$LPADR
4631 025410  013737  002562  002564        MOV     $LPADR,$LPERR
4632 025416  012700  002170        TBG4$:  MOV     #SUPDR6+2,R0
4633 025422  014006                        MOV     -(R0),SP
4634 025424  014005                        MOV     -(R0),R5
4635 025426  014004                        MOV     -(R0),R4
4636 025430  014003                        MOV     -(R0),R3
4637 025432  014002                        MOV     -(R0),R2
4638 025434  014001                        MOV     -(R0),R1
4639 025436  014000                        MOV     -(R0),R0
4640 025440                                TESTAREA                        ;ENTER SUPERVISOR MODE
4641 025446  005737  002426                TST     NOSUPER
4642 025452  001403                        BEQ     1$
4643 025454  012706  000700                MOV     #USESTK,USP
4644 025460  000402                        BR      2$
4645 025462  012706  000740        1$:     MOV     #SUPSTK,SSP
4646 025466  104424                2$:     CACHOFF                         ;TURN CACHE OFF
4647 025470  004777  154560                CALL    @SUPDOADD
4648 025474  104423                        CACHON                          ;TURN CACHE ON
4649 025476  104417                        KERNEL                          ;ENTER KERNEL MODE
4650 025500  000004                        SCOPE
4651 025502                                POP     $LPADR,$LPERR
4652 025512  000207                        RETURN
```

```
                                .SBTTL  MEMORY TEST PATTERN ROUTINES
                        ;*******************************************************************************
                        ; PATTERN REGISTER CONVENTIONS
                        ;       R0      FIRST ADDRESS OF PATTERN (FIRST,LAST+2,ETC)
                        ;       R1      NUMBER OF ADDRESSES IN PATTERN (SIZE)
                        ;       R2      DATA FOR PATTERN (ONES,52525,ETC)
                        ;       R3      COPY OF R1 (IF NECESSARY)
                        ;       R4      COPY OF R0 (IF NECESSARY)
                        ;       R5      COPY OF R2 (IF NECESSARY)
                        ;*******************************************************************************
025514                  MTP000: SUBTST  <<MTP000        BASIC DATA TEST>>
                        ;*******************************************************************************
                        ;*SUBTEST       MTP000  BASIC DATA TEST
                        ;*******************************************************************************
025514  010220          1$:     MOV     R2,(R0)+        ;V177640
025516  077102                  SOB     R1,MTP000       ;V177642
025520  000240                  NOP                     ;V177644
025522  012401          2$:     MOV     (R4)+,R1        ;V177646
025524  020102                  CMP     R1,R2           ;V177650
025526  001402                  BEQ     3$              ;V177652
025530  104430                  PERR02                  ;V177654
025532  000240                  NOP                     ;V177656
025534  077306          3$:     SOB     R3,2$           ;V177660
025536  000207                  RETURN                  ;V177662
025540                  MTP001: SUBTST  <<MTP001        ADDRESS TEST>>
                        ;*******************************************************************************
                        ;*SUBTEST       MTP001  ADDRESS TEST
                        ;*******************************************************************************
025540  010220          3$:     MOV     R2,(R0)+        ;V177640
025542  062702  000002          ADD     #2,R2           ;V177642
025546  077104                  SOB     R1,3$           ;V177646
025550  000240                  NOP                     ;V177650
025552  012400          1$:     MOV     (R4)+,R0        ;V177652
025554  020005                  CMP     R0,R5           ;V177654
025556  001401                  BEQ     2$              ;V177656
025560  104427                  PERR01                  ;V177660
025562  062705  000002  2$:     ADD     #2,R5           ;V177662
025566  077307                  SOB     R3,1$           ;V177666
025570  000207                  RETURN                  ;V177672
025572                  MTP002: SUBTST  <<MTP002        COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)>>
                        ;*******************************************************************************
                        ;*SUBTEST       MTP002  COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)
                        ;*******************************************************************************
025572  010540          3$:     MOV     R5,-(R0)        ;V177640
025574  062705  000002          ADD     #2,R5           ;V177642
025600  077104                  SOB     R1,3$           ;V177646
025602  000240                  NOP                     ;V177650
025604  162702  000002  1$:     SUB     #2,R2           ;V177652
025610  012401                  MOV     (R4)+,R1        ;V177656
025612  020102                  CMP     R1,R2           ;V177660
025614  001401                  BEQ     2$              ;V177662
025616  104430                  PERR02                  ;V177664
025620  077307          2$:     SOB     R3,1$           ;V177666
025622  000207                  RETURN                  ;V177670
```

```
4702 025624                          MTPA03: SUBTST  <<MTPA03        3 XOR 9 WORST CASE NOISE TEST (WRITE)>>
                                     ;*********************************************************************************
                                     ;*SUBTEST       MTPA03  3 XOR 9 WORST CASE NOISE TEST (WRITE)
                                     ;*********************************************************************************
4703                                         ;R1 = ADDRESS
4704                                         ;R2 = SMALL LOOP CONSTANT
4705                                         ;R3 = NUM OF ADD TO TEST (LARGE LOOP)
4706                                         ;R4 = GOOD DATA
4707                                         ;R5 = MEDIUM LOOP CONSTANT
4708                                         .ENABL  LSB
4709 025624  010421                  1$:     MOV     R4,(R1)+        ;V177640
4710 025626  010421                          MOV     R4,(R1)+        ;V177642
4711 025630  077203                          SOB     R2,1$           ;V177644
4712 025632  005104                          COM     R4              ;V177646
4713 025634  052704                          BIS     (PC)+,R4        ;V177650
4714 025636  000401                  WARN2:  401                     ;V177652 WARNING LOCATION IS MODIFIED BEFORE LOADING
4715 025640  012702  000004                  MOV     #4,R2           ;V177654
4716 025644  077511                          SOB     R5,1$           ;V177660
4717 025646  005104                          COM     R4              ;V177662
4718 025650  052704                          BIS     (PC)+,R4        ;V177664
4719 025652  000401                  WARN3:  401                     ;V177666 WARNING LOCATION IS MODIFIED BEFORE LOADING
4720 025654  012705  000100                  MOV     #64.,R5         ;V177670
4721 025660  077317                          SOB     R3,1$           ;V177674
4722 025662  000207                          RETURN                  ;V177676
4723                                         .DSABL  LSB
4724
4725 025664                          MTPB03: SUBTST  <<MTPB03        3 XOR 9 WORST CASE NOISE TEST (READ)>>
                                     ;*********************************************************************************
                                     ;*SUBTEST       MTPB03  3 XOR 9 WORST CASE NOISE TEST (READ)
                                     ;*********************************************************************************
4726                                         .ENABL  LSB
4727 025664  000137  025724          1$:     JMP     @#MTPC03        ;V177640        GO TO V172360
4728 025670  077203                          SOB     R2,1$           ;V177644
4729 025672  005104                          COM     R4              ;V177646
4730 025674  052704                          BIS     (PC)+,R4        ;V177650
4731 025676  000401                  WARN4:  401                     ;V177652 WARNING LOCATION IS MODIFIED BEFORE LOADING
4732 025700  012702  000004                  MOV     #4,R2           ;V177654
4733 025704  077511                          SOB     R5,1$           ;V177660
4734 025706  005104                          COM     R4              ;V177662
4735 025710  052704                          BIS     (PC)+,R4        ;V177664
4736 025712  000401                  WARN5:  401                     ;V177666 WARNING LOCATION IS MODIFIED BEFORE LOADING
4737 025714  012705  000100                  MOV     #64.,R5         ;V177670
4738 025720  077317                          SOB     R3,1$           ;V177674
4739 025722  000207                          RETURN                  ;V177676
4740                                         .DSABL  LSB
```

```
 4743 025724                          MTPC03: SUBTST  <<MTPC03        TEST DATA SUBPROGRAM>>
                                      ;*********************************************************************
                                      ;*SUBTEST       MTPC03  TEST DATA SUBPROGRAM
                                      ;*********************************************************************
 4744 025724  020421                          CMP     R4,(R1)+        ;V172360
 4745 025726  001401                          BEQ     1$              ;V172362
 4746 025730  104431                          PERR03                  ;V172364
 4747 025732  005141                  1$:     COM     -(R1)           ;V172366
 4748 025734  005111                          COM     (R1)            ;V172370
 4749 025736  000137  025742                  JMP     @#MTPD03        ;V172372        GO TO V172260
 4750
 4751 025742                          MTPD03: SUBTST  <<MTPD03        TEST DATA SUBSUBPROGRAM>>
                                      ;*********************************************************************
                                      ;*SUBTEST       MTPD03  TEST DATA SUBSUBPROGRAM
                                      ;*********************************************************************
 4752 025742  020421                          CMP     R4,(R1)+        ;V172260
 4753 025744  001401                          BEQ     1$              ;V172262
 4754 025746  104431                          PERR03                  ;V172264
 4755 025750  005127                  1$:     COM     (PC)+           ;V172266
 4756 025752  000000                          0                       ;V172270
 4757 025754  001363                          BNE     MTPC03          ;V172272        GO TO V172360
 4758 025756  000137  025670                  JMP     @#MTPB03+4      ;V172274        GO TO V177644
```

```
 4761 025762                            MTPA04: SUBTST  <<MTPA04         ROTATING ZEROS TEST>>
                                        ;*****************************************************************************
                                        ;*SUBTEST        MTPA04  ROTATING ZEROS TEST
                                        ;*****************************************************************************
 4762 025762  012705  000010            1$:     MOV     #8.,R5           ;V177640
 4763 025766  010504                            MOV     R5,R4            ;V177644
 4764 025770  000241                            CLC                      ;V177646
 4765 025772  000137  026016                    JMP     @#MTPB04         ;V177650
 4766 025776  016004  177776                    MOV     -2(R0),R4        ;V177654
 4767 026002  103402                            BCS     2$               ;V177660
 4768 026004  020204                            CMP     R2,R4            ;V177662
 4769 026006  001401                            BEQ     3$               ;V177664
 4770 026010  104432                    2$:     PERR04                   ;V177666
 4771 026012  077115                    3$:     SOB     R1,1$            ;V177670
 4772 026014  000207                            RETURN                   ;V177672
 4773
 4774 026016                            MTPB04: SUBTST  <<MTPB04         SUBR    ROTATING BIT>>
                                        ;*****************************************************************************
                                        ;*SUBTEST        MTPB04  SUBR    ROTATING BIT
                                        ;*****************************************************************************
 4775 026016  106110                    1$:     ROLB    (R0)             ;V172360
 4776 026020  077502                            SOB     R5,1$            ;V172362
 4777 026022  106120                            ROLB    (R0)+            ;V172364
 4778 026024  106110                    2$:     ROLB    (R0)             ;V172366
 4779 026026  077402                            SOB     R4,2$            ;V172370
 4780 026030  106120                            ROLB    (R0)+            ;V172372
 4781 026032  000137  025776                    JMP     @#MTPA04+14      ;V172374
 4782
 4783 026036                            MTP005: SUBTST  <<MTP005         ROTATION ONES TEST>>
                                        ;*****************************************************************************
                                        ;*SUBTEST        MTP005  ROTATION ONES TEST
                                        ;*****************************************************************************
 4784 026036  012705  000010            1$:     MOV     #8.,R5           ;V177640
 4785 026042  010504                            MOV     R5,R4            ;V177644
 4786 026044  000261                            SEC                      ;V177646
 4787 026046  000137  026016                    JMP     @#MTPB04         ;V177650
 4788 026052  016004  177776                    MOV     -2(R0),R4        ;V177654
 4789 026056  103002                            BCC     2$               ;V177660 IF THIS HAPPENS THE GOOD & BAD MATCH
 4790 026060  020204                            CMP     R2,R4            ;V177662
 4791 026062  001401                            BEQ     3$               ;V177664
 4792 026064  104432                    2$:     PERR04                   ;V177666
 4793 026066  077115                    3$:     SOB     R1,1$            ;V177670
 4794 026070  000207                            RETURN                   ;V177672
```

```
4797 026072                          MTP006: SUBTST  <<MTP006        INITIAL DATA TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MTP006  INITIAL DATA TEST
                                     ;*****************************************************************************
4798                                 ;       THIS TEST CHECKS THE DI/DO LINES BY
4799                                 ;       SHIFTING A 1 THROUGH THE WORD.
4800 026072  012737  000001  002234         MOV     #1,DATBUF       ;SET THE FIRST TEST BIT
4801 026100  005037  002236                 CLR     DATBUF+2        ;CLEAR 2ND WORD
4802 026104  013771  002234  000000  1$:    MOV     DATBUF,@(R1)    ;WRITE TEST WORD 1
4803 026112  013771  002236  000002         MOV     DATBUF+2,@2(R1) ;AND TEST WORD 2
4804 026120  017102  000000                 MOV     @(R1),R2
4805 026124  023702  002234                 CMP     DATBUF,R2       ;NOW READ THEM
4806 026130  001401                         BEQ     2$              ;BR IF FIRST 16 OK
4807 026132  104433                         PERR07                  ;ERROR TRAP
4808
4809 026134  017102  000002         2$:    MOV     @2(R1),R2
4810 026140  023702  002236                 CMP     DATBUF+2,R2     ;NOW READ SECOND WORD
4811 026144  001401                         BEQ     3$              ;BR IF OK
4812 026146  104434                         PERR10                  ;ERROR TRAP
4813
4814 026150  005737  002236         3$:    TST     DATBUF+2        ;HAS LAST BIT BEEN TESTED ?
4815 026154  100405                         BMI     4$              ;MINUS MEANS BIT 31
4816 026156                                 DLEFT   DATBUF          ;NO, SHIFT TEST BIT LEFT
4817 026166  000746                         BR      1$              ;GO WRITE NEW TEST DATA
4818                                                                 ;NOW GOING TO SHIFT A 0 IN DATA DIRECTION
4819 026170  012737  177776  002234  4$:    MOV     #177776,DATBUF  ;PUT A 0 IN BIT 0
4820 026176  012737  177777  002236         MOV     #-1, DATBUF+2   ;AND 1'S IN ALL OTHERS
4821 026204  013771  002234  000000  5$:    MOV     DATBUF,@(R1)    ;WRITE THE DATA
4822 026212  013771  002236  000002         MOV     DATBUF+2,@2(R1) ;2 WORDS WORTH
4823 026220  017102  000000                 MOV     @(R1),R2
4824 026224  023702  002234                 CMP     DATBUF,R2       ;NOW READ FIRST WORD
4825 026230  001401                         BEQ     6$              ;BR IF OK
4826 026232  104433                         PERR07
4827
4828 026234  017102  000002         6$:    MOV     @2(R1),R2
4829 026240  023702  002236                 CMP     DATBUF+2,R2     ;NOW, READ SECOND WORD
4830 026244  001401                         BEQ     7$              ;BR IF OK
4831 026246  104434                         PERR10
4832
4833 026250  005737  002236         7$:    TST     DATBUF+2        ;TESTED BIT 31 YET?
4834 026254  100005                         BPL     8$              ;BR IF YES, WE'RE DONE
4835 026256                                 DLEFT   DATBUF
4836 026266  000746                         BR      5$              ;KEEP GOING
4837 026270  000207                 8$:    RETURN
```

```
                        MTP007: SUBTST  <<MTP007        ADDRESS BIT TEST>>
                        ;*****************************************************************************
                        ;*SUBTEST        MTP007  ADDRESS BIT TEST
                        ;*****************************************************************************
                        ;       THIS TEST CHECKS TO SEE THAT EACH ADDRESS
                        ;       BIT IN EACH 16K BANK CAN BE ASSERTED UNIQUELY.
                        ;       IT CHECKS FOR ADDRESS BITS THAT MAY BE STUCK
                        ;       HIGH, STUCK LOW OR STUCK TOGETHER.
026272  111100                  MOVB    (R1),R0
026274  105700                  TSTB    R0              ;READ AND COMPARE FOR ZEROS
026276  001401                  BEQ     1$              ;BR IF OK
026300  104435                  PERR11
                        
026302  105111          1$:     COMB    (R1)            ;COMPLEMENT THE BYTE
026304  111100                  MOVB    (R1),R0
026306  105700                  TSTB    R0              ;READ FOR NON ZEROS
026310  001001                  BNE     2$              ;BR IF OK
026312  104436                  PERR12
                        
026314  040201          2$:     BIC     R2,R1           ;MASK OFF THE ASSERTED BIT
026316  006302                  ASL     R2              ;SHIFT R2 FOR NEXT BIT
026320  050201                  BIS     R2,R1           ;SET THE NEW BIT INTO R1
026322  011100                  MOV     (R1),R0
026324  005700                  TST     R0              ;READ THE NEW ADDRESS
026326  001401                  BEQ     3$              ;READ FOR ZEROS
026330  104437                  PERR13
                        
026332  005111          3$:     COM     (R1)            ;COMPL THE WORD
026334  011100                  MOV     (R1),R0
026336  005700                  TST     R0              ;READ IT AGAIN
026340  001001                  BNE     4$
026342  104440                  PERR14
                        
026344  022702  100000  4$:     CMP     #100000,R2
026350  001407                  BEQ     5$
026352  022702  010000          CMP     #10000,R2       ;CHECK FOR MSB IN 4K BANK
026356  001356                  BNE     2$              ;NOT LAST BIT, BRANCH
026360  006302                  ASL     R2
026362  012701  160000          MOV     #160000,R1
026366  000752                  BR      2$
026370  000207          5$:     RETURN
```

```
                                MTP010: SUBTST  <<MTP010        BYTE ADDRESSING TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MTP010  BYTE ADDRESSING TEST
                                ;*****************************************************************************
                                        ;TEST 3 THIS TEST CHECKS FOR PROPER
                                        ;       BYTE ADDRESSING WITH ECC DISABLED
 026372  010402                        MOV     R4,R2           ;R4 HAS LOWEST ADDRESS
 026374  010403                        MOV     R4,R3           ;PUT IT IN R3 ALSO
 026376  062702  000004                ADD     #4,R2           ;POINT R2 TO LAST BYTE +1
 026402  012713  177777                MOV     #-1,(R3)        ;WRITE ALL ONES IN
 026406  012763  177777  000002        MOV     #-1,2(R3)       ;THE 4 TEST BYTES
 026414  105013                1$:     CLRB    (R3)            ;CLEAR A BYTE
 026416  010401                        MOV     R4,R1           ;INITIALIZE R1 FOR EACH PASS
 026420  020201                2$:     CMP     R2,R1           ;IF EQUAL, JUST READ LAST BYTE
 026422  001420                        BEQ     6$              ;BR IF EQUAL
 026424  020301                        CMP     R3,R1           ;IS THIS THE BYTE OF ZEROS
 026426  001007                        BNE     4$              ;BR IF NOT
 026430  111100                        MOVB    (R1),R0
                                ;WARNING IF YOU OPTOMIZE CHANGE THE PCBUMP FOR THIS ERROR INCASE OF TRAPS
 026432  022700  000000                CMP     #0,R0           ;IT IS, COMPARE FOR ZEROS
 026436  001401                        BEQ     3$
 026440  104435                        PERR11
 026442  005201                3$:     INC     R1              ;NEXT BYTE
 026444  000765                        BR      2$              ;RETURN
 026446  111100                4$:     MOVB    (R1),R0
 026450  122700  177777                CMPB    #-1,R0          ;ITS NOT THE BYTE OF 0'S, READ 1'S
 026454  001401                        BEQ     5$
 026456  104436                        PERR12
 026460  005201                5$:     INC     R1              ;MOVE TO NEXT BYTE
 026462  000756                        BR      2$
 026464  112713  177777        6$:     MOVB    #-1,(R3)        ;RESTORE 1'S TO BYTE JUST TESTED
 026470  005203                        INC     R3              ;INC TO NEXT BYTE
 026472  020302                        CMP     R3,R2           ;WAS THAT JUST THE LAST ONE?
 026474  001347                        BNE     1$              ;BR IF NO
 026476  000207                        RETURN
```

```
 4916 026500                        MTP011: SUBTST  <<MTP011        SINGLE BIT ERROR TEST>>
                                    ;*****************************************************************************
                                    ;*SUBTEST       MTP011  SINGLE BIT ERROR TEST
                                    ;*****************************************************************************
 4917                                       ;(1)    CREATE A SINGLE BIT ERROR
 4918                                       ;
 4919                                       ;(2)    READ BACK SBE UNCORRECTED (WITH ECC DISABLE)
 4920                                       ;
 4921                                       ;(3)    ENABLE ECC & READ CORRECTED DATA
 4922                                       ;
 4923                                       ;(4)    CHECK THAT THE SBE FLAG WAS SET FROM THE LAST READ
 4924                                       ;
 4925                                       ;(5)    DO (1-4) FOR DATA CONSISTING OF 1 BIT SET IN EACH OF 32
 4926                                       ;       POSITIONS OF A DOUBLE WORD
 4927                                       ;       THEN DO IT AGAIN FOR 1 BIT CLEARED IN EACH OF 32 POSITIONS OF
 4928                                       ;       A DOUBLE WORD
 4929                                       ;       IE (64 TIMES)
 4930                                       ;
 4931                                       ;(6)    DO (1-5) FOR A SBE IN EACH OF 32 BIT POSITIONS
 4932                                       ;       IE (RUN TEST 64 * 32 = 2048 TIMES)
 4933 026500  104503                       CLR1CSR                 ;CLEAR 1 SELECTED CSR
 4934                                       ;BIG LOOP
 4935 026502  012737  000001  002234  MTLA11: MOV     #1,DATBUF       ;INITIAL DATA
 4936 026510  005037  002236                CLR     DATBUF+2        ;32 BITS WORTH
 4937                                       ;MEDIUM LOOP
 4938 026514  012737  000001  002244  MTLB11: MOV     #1,SBEMSK       ;INITIAL ERROR MASK
 4939 026522  005037  002246                CLR     SBEMSK+2        ;32 BITS WORTH
 4940                                       ;LITTLE LOOP
 4941 026526  013737  002234  002240  MTLC11: MOV     DATBUF,TSTDAT   ;
 4942 026534  013737  002236  002242          MOV     DATBUF+2,TSTDAT+2;TO SAVE ORIG DATA
 4943 026542  105737  002256                TSTB    PASFLG  ;COMP DATA ON SECOND PASSONLY
 4944 026546  001404                       BEQ     4$              ;BR IF FIRST PASS
 4945 026550  005137  002240                COM     TSTDAT          ;SECOND PASS, COMP BOTH WORDS
 4946 026554  005137  002242                COM     TSTDAT+2
 4947 026560  013702  002240          4$:   MOV     TSTDAT,R2
 4948 026564  013703  002242                MOV     TSTDAT+2,R3
 4949 026570  012737  002240  002272        MOV     #TSTDAT,SOURCE  ;SET UP ADDRESS FOR CHKGEN
 4950 026576  004737  040360                CALL    CHKGEN          ;GEN CHECKBITS ON TSTDAT
 4951                                       ;*********************************
 4952                                       ;** CREATE A SINGLE BIT ERROR **
 4953                                       ;*********************************
 4954 026602  013701  002244                MOV     SBEMSK,R1
 4955 026606  074137  002240                XOR     R1,TSTDAT
 4956 026612  013701  002246                MOV     SBEMSK+2,R1
 4957 026616  074137  002242                XOR     R1,TSTDAT+2
 4958 026622  013701  002362          MTLD11: MOV     TESTADD,R1      ;FIRST TEST ADDRESS
 4959 026626  013705  002364                MOV     TESTADD+2,R5    ;SECOND TEST ADDRESS
 4960 026632  104471                       ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
 4961 026634  013711  002240                MOV     TSTDAT,(R1)     ;WRITE FIRST 16 BITS
 4962 026640  104475                       CB1CSR                  ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
 4963 026642  013715  002242                MOV     TSTDAT+2,(R5)   ;WRITE SECOND 16 BITS AND
 4964                                                               ;CHECK BITS. WE NOW HAVE CHECKBITS
 4965                                                               ;GENERATED ON DATBUF AND DATA WITH
 4966                                                               ;ONE BIT IN ERROR (AS PER SBEMSK).
 4967 026646  104471                       ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
 4968 026650  011100                       MOV     (R1),R0
 4969 026652  020037  002240                CMP     R0,TSTDAT       ;READ THE LOW WORD (UNCORRECTED)
```

```
4970 026656  001403                        BEQ     6$              ;BR IF OK
4971 026660  010137  002032                MOV     R1,ADDRESS
4972 026664  104455                        PERR31
4973
4974 026666  011500                6$:     MOV     (R5),R0
4975 026670  020037  002242                CMP     R0,TSTDAT+2     ;READ THE HIGH WORD (UNCORRECTED)
4976 026674  001403                        BEQ     7$              ;BR IF OK
4977 026676  010537  002032                MOV     R5,ADDRESS
4978 026702  104455                        PERR31
4979
4980 026704                        7$:     IF KFLAG IS FALSE
4981 026712  104426                          READCSR
4982 026714                                   IF #BIT4 OFF.IN CSR OR #BIT15 OFF.IN CSR
4983 026734  104045                             ERROR       +45
4984 026736                                   END; OF IF #BIT4
4985 026736                                END; OF IF KFLAG
4986 026736  104512                        ERRGEN
4987 026740  104503                        CLR1CSR                 ;CLEAR 1 SELECTED CSR
4988 026742  011100                        MOV     (R1),R0
4989 026744  020002                        CMP     R0,R2           ;SEE IF ITS BEEN CORRECTED
4990 026746  001401                        BEQ     8$              ;IT SHOULD HAVE BEEN
4991 026750  104456                        PERR32
4992
4993 026752  104510                8$:     TSTREAD                 ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
4994 026754  103411                        BCS     9$              ;BR IF IT IS SET
4995 026756                                SET     HEADER          ;ENABLE PRINTING OF ERROR HEADER INFO
4996 026764  010137  002032                MOV     R1,ADDRESS
4997 026770  104460                        PERR34
4998 026772                                SET     HEADER          ;ENABLE PRINTING OF ERROR HEADER INFO
4999
5000 027000  104503                9$:     CLR1CSR                 ;CLEAR 1 SELECTED CSR
5001 027002  011500                        MOV     (R5),R0
5002 027004  020003                        CMP     R0,R3           ;SEE IF ITS BEEN CORRECTED
5003 027006  001401                        BEQ     10$             ;BR IF OK
5004 027010  104456                        PERR32
5005
5006 027012  104510               10$:     TSTREAD                 ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
5007 027014  103411                        BCS     11$             ;BR IF YES
5008 027016                                SET     HEADER          ;ENABLE PRINTING OF ERROR HEADER INFO
5009 027024  010137  002032                MOV     R1,ADDRESS
5010 027030  104460                        PERR34
5011 027032                                SET     HEADER          ;ENABLE PRINTING OF ERROR HEADER INFO
5012 027040  104512               11$:     ERRGEN                  ;TEST ERROR ADDRESS
5013 027042  105737  002256                TSTB    PASFLG
5014 027046  100452                        BMI     15$
5015 027050  005737  002246                TST     SBEMSK+2        ;TEST FOR LAST MASK BIT
5016 027054  100405                        BMI     12$             ;MINUS MEANS BIT 31
5017 027056                                DLEFT   SBEMSK
5018 027066  000617                        BR      MTLC11
5019 027070                        12$:    IF #SW11 SET.IN @SWR THEN GOTO 13$
5020 027100                                IF QVFLAG IS TRUE THEN GOTO 13$
5021 027106  005737  002236                TST     DATBUF+2        ;LAST DATA BIT ?
5022 027112  100406                        BMI     13$             ;WHICH IS BIT 31
5023 027114                                DLEFT   DATBUF
5024 027124  000137  026514                JMP     MTLB11
5025 027130  105737  002256         13$:    TSTB    PASFLG  ;FIRST OR SECOND PASS ?
5026 027134  001004                        BNE     14$             ;NON ZERO MEANS WE'RE DONE
```

```
 5027 027136  105237  002256                    INCB    PASFLG  ;NOT DONE, GO DO SECOND PASS
 5028 027142  000137  026502                    JMP     MTLA11
 5029 027146  052737  000200  002256  14$:      BIS     #BIT7,PASFLG
 5030 027154  005002                            CLR     R2
 5031 027156  005003                            CLR     R3
 5032 027160  005037  002240                    CLR     TSTDAT
 5033 027164  005037  002242                    CLR     TSTDAT+2
 5034 027170  012704  000040                    MOV     #40,R4
 5035 027174  012737  003740  002274  15$:      MOV     #3740,CHECK
 5036 027202  074437  002274                    XOR     R4,CHECK
 5037 027206  006304                            ASL     R4
 5038 027210  032704  020000                    BIT     #BIT13,R4
 5039 027214  001002                            BNE     16$
 5040 027216  000137  026622                    JMP     MTLD11
 5041                                           ;CLEAR OUT ANY DBE'S OR SBE'S
 5042 027222  104471                    16$:    ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
 5043 027224  013701  002362                    MOV     TESTADD,R1
 5044 027230  013705  002364                    MOV     TESTADD+2,R5
 5045 027234                                    CLEAR   (R1),(R5)
 5046 027240  104503                            CLR1CSR                 ;CLEAR 1 SELECTED CSR
 5047 027242  000207                            RETURN
```

SEQ 0499

```
                                MTP012: SUBTST  <<MTP012        WRITE BYTE CLEARS SBE TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST       MTP012  WRITE BYTE CLEARS SBE TEST
                                ;*****************************************************************************
                                                ;SINGLE BIT ERROR TEST TO INSURE THAT A WRITE
                                                ;BYTE CLEARS SINGLE BIT ERRORS.
027244  104503                        CLR1CSR                 ;CLEAR 1 SELECTED CSR
027246  012737  000001  002234        MOV     #1,DATBUF       ;INITIAL DATA
027254  005037  002236                CLR     DATBUF+2        ;32 BITS WORTH
027260  012737  000001  002244  1$:   MOV     #1,SBEMSK       ;INITIAL ERROR MASK
027266  005037  002246                CLR     SBEMSK+2        ;32 BITS WORTH
027272  013737  002234  002240  2$:   MOV     DATBUF,TSTDAT   ;SAVE ORIGINAL DATA
027300  013737  002236  002242        MOV     DATBUF+2,TSTDAT+2;BOTH WORDS
027306  012737  002240  002272        MOV     #TSTDAT,SOURCE  ;NEED ADDRESS FOR CHKGEN
027314  004737  040360                CALL    CHKGEN          ;GENERATE CHECK BITS
027320  013701  002244                MOV     SBEMSK,R1
027324  074137  002240                XOR     R1,TSTDAT
027330  013701  002246                MOV     SBEMSK+2,R1
027334  074137  002242                XOR     R1,TSTDAT+2
027340  013704  002362                MOV     TESTADD,R4      ;FIRST TEST ADDRESS
027344  010401                        MOV     R4,R1           ;PUT IT IN R1 ALSO
027346  104471                        ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
027350  013711  002240                MOV     TSTDAT,(R1)     ;WRITE 16 BITS
027354  104475                        CB1CSR                  ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
027356  060501                        ADD     R5,R1           ;INDEX UP TO SECOND WORD
027360  013711  002242                MOV     TSTDAT+2,(R1)   ;WRITE HIGH WORD+CHECKBITS
027364  104503                        CLR1CSR                 ;CLEAR 1 SELECTED CSR
                                                ;IT'S DANGEROUS IF WE DON'T
027366  012702  002244                MOV     #SBEMSK,R2      ;ADDRESS OF ERROR MASK
027372  160501                        SUB     R5,R1           ;RETURN TO FIRST WORD
027374  112711  177777          3$:   MOVB    #-1,(R1)        ;WRITE A BYTE OF 1'S
027400  005737  002500                TST     KFLAG           ;IS THIS MF11S-K
027404  001403                        BEQ     4$              ;BRANCH IF NOT - IT'S MS11-M
027406  132712  177777                BITB    #-1,(R2)        ;DID THIS BYTE HAVE THE BAD BIT IN IT?
027412  001420                        BEQ     6$              ;NO - BRANCH
027414  104510                  4$:   TSTREAD                 ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
027416  103011                        BCC     5$              ;NO - SKIP
027420                                SET     HEADER          ;ENABLE PRINTING OF ERROR HEADER INFO
027426  010137  002032                MOV     R1,ADDRESS
027432  104017                        ERROR   +17
027434                                SET     HEADER          ;ENABLE PRINTING OF ERROR HEADER INFO

027442  111100                  5$:   MOVB    (R1),R0
027444  122700  177777                CMPB    #-1,R0          ;CHECK DATA
027450  001414                        BEQ     7$              ;BR IF OK
027452  104457                        PERR33

027454  104510                  6$:   TSTREAD                 ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
                                                ;READ THE BYTE
                                                ;SBE ERROR BIT ONLY SET ?
027456  103771                        BCS     5$              ;SHOULD BE SET, BR IF OK
027460                                SET     HEADER          ;ENABLE PRINTING OF ERROR HEADER INFO
027466  010137  002032                MOV     R1,ADDRESS
027472  104460                        PERR34
027474                                SET     HEADER          ;ENABLE PRINTING OF ERROR HEADER INFO

027502  132712  177777          7$:   BITB    #-1,(R2)        ;CHECK FOR LAST BYTE
```

```
 5104 027506  001012                        BNE     8$              ;
 5105 027510  005202                        INC     R2
 5106 027512  005201                        INC     R1              ;MOVE TO NEXT BYTE
 5107 027514  013704  002362                MOV     TESTADD,R4      ;FIRST TEST ADDRESS
 5108 027520  032701  000002                BIT     #2,R1           ;TEST FOR LOWER WORD
 5109 027524  001723                        BEQ     3$              ;BR IF IT'S LOW 16 BITS
 5110 027526  062704  000002                ADD     #2,R4           ;ADJUST POINTER FOR ERROR REPT.
 5111 027532  000720                        BR      3$
 5112 027534  005737  002246          8$:   TST     SBEMSK+2        ;LAST ERROR BIT ?
 5113 027540  100405                        BMI     9$              ;MINUS MEANS BIT 31
 5114 027542                                DLEFT   SBEMSK
 5115 027552  000647                        BR      2$
 5116 027554                          9$:   IF #SW11 SET.IN @SWR THEN GOTO 10$
 5117 027564                                IF QVFLAG IS TRUE THEN GOTO 10$
 5118 027572  005737  002236                TST     DATBUF+2        ;LAST DATA BIT?
 5119 027576  100405                        BMI     10$             ;MINUS = BIT 31
 5120 027600                                DLEFT   DATBUF
 5121 027610  000623                        BR      1$
 5122                                       ;CLEAR OUT ANY DBE'S OR SBE'S
 5123 027612  104471                  10$:  ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
 5124 027614  013701  002362                MOV     TESTADD,R1
 5125 027620  005011                        CLR     (R1)
 5126 027622  060501                        ADD     R5,R1
 5127 027624  005011                        CLR     (R1)
 5128 027626  104503                        CLR1CSR                 ;CLEAR 1 SELECTED CSR
 5129 027630  000207                        RETURN
```

```
5132 027632                                MTP013: SUBTST  <<MTP013        CREATE DOUBLE BIT ERROR TEST>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       MTP013  CREATE DOUBLE BIT ERROR TEST
                                           ;*****************************************************************************
5133                                                       ;DOUBLE BIT ERROR FORCE TO CHECK DOUBLE ERROR LOGIC
5134 027632  104503                                CLR1CSR                 ;CLEAR 1 SELECTED CSR
5135 027634  012701  002362                        MOV     #TESTADD,R1
5136 027640  005037  002234            1$:         CLR     DATBUF          ;MAKE INITIAL DATA
5137 027644  005037  002236                        CLR     DATBUF+2        ;ALL ZEROS
5138 027650  012737  000001  002244    2$:         MOV     #1,SBEMSK       ;INITIAL SINGLE ERROR MASK
5139 027656  005037  002246                        CLR     SBEMSK+2        ;SECOND WORD
5140 027662  012737  000001  002250    3$:         MOV     #1,DBEMSK       ;INITIAL DOUBLE ERROR MASK
5141 027670  005037  002252                        CLR     DBEMSK+2        ;32 BITS HERE ALSO
5142 027674  013737  002234  002240    4$:         MOV     DATBUF,TSTDAT   ;
5143 027702  013737  002236  002242                MOV     DATBUF+2,TSTDAT+2
5144 027710  105737  002256                        TSTB    PASFLG  ;NO COMPLEMENTING FIRST PASS
5145 027714  001404                                BEQ     5$
5146 027716  005137  002240                        COM     TSTDAT          ;COMP FIRST WORD
5147 027722  005137  002242                        COM     TSTDAT+2        ;SECOND WORD
5148 027726  104503                        5$:     CLR1CSR                 ;CLEAR 1 SELECTED CSR
5149 027730  023737  002244  002250              CMP     SBEMSK,DBEMSK   ;CAN'T HAVE THE SAME ERROR BIT SET
5150 027736  001004                                BNE     6$              ;IN BOTH MASKS
5151 027740  023737  002246  002252              CMP     SBEMSK+2,DBEMSK+2;COULD BE EQUAL IN SECOND WORD
5152 027746  001460                                BEQ     13$             ;GO MAKE THEM NOT EQUAL
5153 027750  012737  002240  002272    6$:         MOV     #TSTDAT,SOURCE  ;SOURCE ADDRESS FOR CHKGEN
5154 027756  004737  040360                        CALL    CHKGEN          ;GO GENERATE CHECK BITS
5155 027762  013702  002244                        MOV     SBEMSK,R2
5156 027766  074237  002240                        XOR     R2,TSTDAT
5157 027772  013702  002246                        MOV     SBEMSK+2,R2
5158 027776  074237  002242                        XOR     R2,TSTDAT+2
5159 030002  013702  002250                        MOV     DBEMSK,R2
5160 030006  074237  002240                        XOR     R2,TSTDAT
5161 030012  013702  002252                        MOV     DBEMSK+2,R2
5162 030016  074237  002242                        XOR     R2,TSTDAT+2
5163 030022  104471                        16$:    ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
5164 030024  013731  002240                        MOV     TSTDAT,@(R1)+   ;WRITE 16 BITS
5165 030030  104475                                CB1CSR                  ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
5166 030032  013771  002242  000000              MOV     TSTDAT+2,@(R1)  ;WRITE HIGH WORD
5167 030040  104503                                CLR1CSR                 ;CLEAR 1 SELECTED CSR
5168 030042  162701  000002                        SUB     #2,R1           ;ADJUST TEST ADDRESS
5169 030046  005771  000000                        TST     @(R1)           ;READ THE LOCATION
5170 030052  104501                                WAS1DBE                 ;WAS THERE ANY DOUBLE BIT ERRORS ON 1 SELECTED CSR
5171 030054  103411                                BCS     9$              ;IT SHOULD BE SET
5172 030056                                        SET     HEADER
5173 030064  011137  002032                        MOV     (R1),ADDRESS
5174 030070  104030                                ERROR   +30
5175 030072                                        SET     HEADER
```

```
5178 030100  104512                    9$:     ERRGEN
5179 030102  105737  002256                    TSTB    PASFLG
5180 030106  100452                    BMI     14$
5181 030110  005737  002252            13$:    TST     DBEMSK+2        ;CHECK MASK FOR LAST BIT
5182 030114  100405                    BMI     10$             ;MINUS = BIT31
5183 030116                            DLEFT   DBEMSK
5184 030126  000662                    BR      4$
5185 030130                            10$:    IF #SW11 SET.IN @SWR THEN GOTO 11$
5186 030140                            IF QVFLAG IS TRUE THEN GOTO 11$
5187 030146  005737  002246                    TST     SBEMSK+2        ;CHECK SINGLE ERROR MASK TOO
5188 030152  100405                    BMI     11$             ;BR IF DONE
5189 030154                            DLEFT   SBEMSK
5190 030164  000636                    BR      3$
5191 030166  105737  002256            11$:    TSTB    PASFLG  ;FIRST PASS
5192 030172  001003                    BNE     12$             ;NON ZERO MEANS WE'RE DONE
5193 030174  105237  002256                    INCB    PASFLG  ;FIRST PASS, NOT DONE
5194                                   ;CLEAR OUT ANY DBE'S OR SBE'S
5195 030200  000617                    BR      1$              ;KEEP GOING
5196 030202  052737  000200  002256    12$:    BIS     #BIT7,PASFLG    ;SET UP FOR CHECK BIT TEST
5197 030210  005037  002240                    CLR     TSTDAT
5198 030214  005037  002242                    CLR     TSTDAT+2
5199 030220  012737  000040  002244            MOV     #40,SBEMSK
5200 030226  012737  000100  002250            MOV     #100,DBEMSK
5201 030234  012737  003740  002274    14$:    MOV     #3740,CHECK
5202 030242  013702  002244                    MOV     SBEMSK,R2
5203 030246  074237  002274                    XOR     R2,CHECK
5204 030252  013702  002250                    MOV     DBEMSK,R2
5205 030256  074237  002274                    XOR     R2,CHECK
5206 030262  006337  002250                    ASL     DBEMSK
5207 030266  032737  020000  002250            BIT     #BIT13,DBEMSK
5208 030274  001652                    BEQ     16$
5209 030276  006337  002244                    ASL     SBEMSK
5210 030302  032737  004000  002244            BIT     #BIT11,SBEMSK
5211 030310  001006                    BNE     15$
5212 030312  013737  002244  002250            MOV     SBEMSK,DBEMSK
5213 030320  006337  002250                    ASL     DBEMSK
5214 030324  000743                    BR      14$
5215 030326  104471                    15$:    ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
5216 030330  012701  002362                    MOV     #TESTADD,R1
5217 030334                            CLEAR   @(R1)+,@(R1)
5218 030342  104503                    CLR1CSR                 ;CLEAR 1 SELECTED CSR
5219 030344  000207                    RETURN
```

```
                                MTP014: SUBTST  <<MTP014        WRITE INHIBIT DURING DATIP WITH DBE TEST>>
                                ;*************************************************************************
                                ;*SUBTEST        MTP014  WRITE INHIBIT DURING DATIP WITH DBE TEST
                                ;*************************************************************************
                                        ;THIS TEST CHECKS THE WRITE INHIBIT ON DOUBLE
                                        ;BIT ERRORS DURING A DATIP OPERATION BY USE
                                        ;OF AN 'ASRB' INSTRUCTION.
                                    IF KFLAG IS TRUE THEN $RETURN

                                        ;NOTE- THIS TEST WILL ONLY BE RUN FOR MF11S-K MEMORY.
002234          1$:     CLR     DATBUF          ;INITIAL DATA
002236                  CLR     DATBUF+2        ;2 WORDS WORTH
000001  002244  2$:     MOV     #1,SBEMSK       ;INITIAL ERROR MASK
002246                  CLR     SBEMSK+2        ;
000001  002250  3$:     MOV     #1,DBEMSK       ;DOUBLE ERROR MASK
002252                  CLR     DBEMSK+2        ;2 WORDS
002234  002240  4$:     MOV     DATBUF,TSTDAT   ;PRESERVE ORIG DATA
002236  002242          MOV     DATBUF+2,TSTDAT+2
002256                  TSTB    PASFLG  ;SECOND PASS YET ?
001404                  BEQ     5$              ;BR IF NO
002240                  COM     TSTDAT          ;COMPL DATA  ON SECOND PASS
002242                  COM     TSTDAT+2
104503          5$:     CLR1CSR                 ;CLEAR 1 SELECTED CSR
002250  002244          CMP     DBEMSK,SBEMSK   ;CHECK FOR SAME MASKS
001004                  BNE     6$              ;BR IF OK
002252  002246          CMP     DBEMSK+2,SBEMSK+2
001476                  BEQ     11$             ;BR IF THEY'RE EQUAL
002240  002272  6$:     MOV     #TSTDAT,SOURCE  ;SET UP ADDRESS FOR CHKGEN
040360                  CALL    CHKGEN          ;GENERATE CHECK BITS
002244                  MOV     SBEMSK,R1
002240                  XOR     R1,TSTDAT
002246                  MOV     SBEMSK+2,R1
002242                  XOR     R1,TSTDAT+2
002250                  MOV     DBEMSK,R1
002240                  XOR     R1,TSTDAT
002252                  MOV     DBEMSK+2,R1
002242                  XOR     R1,TSTDAT+2
002362          7$:     MOV     #TESTADD,R1     ;TEST ADDRESS
104471                  ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
002240                  MOV     TSTDAT,@(R1)+   ;WRITE FIRST 16 BITS
104475                  CB1CSR                  ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
002242  000000          MOV     TSTDAT+2,@(R1)  ;SECOND 16 BITS+CHECKBITS
002257                  CLRB    UPPFLG          ;INDICATE LOWER WORD
002362                  MOV     TESTADD,R3      ;TEST ADDRESS
104503          8$:     CLR1CSR                 ;CLEAR 1 SELECTED CSR
106223                  ASRB    (R3)+           ;SPECIAL DATIP INSTRUCTION
015100                  MOV     @-(R1),R0
023700  002240          CMP     TSTDAT,R0       ;CHECK FOR UNCHANGED DATA
001404                  BEQ     9$              ;SHOULD BE UNCHANGED
017137  000000  002032  MOV     @(R1),ADDRESS
104455                  PERR31

062701  000002  9$:     ADD     #2,R1           ;POINT TO UPPER WORD
017100  000000          MOV     @(R1),R0
023700  002242          CMP     TSTDAT+2,R0     ;READ IT
001404                  BEQ     10$             ;BR IF UNCHANGED
017137  000000  002032  MOV     @(R1),ADDRESS
```

```
                        PERR31
                        
030644  122737  000003  002257  10$:    CMPB    #3,UPPFLG               ;LOWER WORD
030652  001403                         BEQ     11$             ;BR IF NO
030654  105237  002257                 INCB    UPPFLG
030660  000744                         BR      8$
030662  105737  002256          11$:    TSTB    PASFLG
030666  100453                         BMI     15$             ;BRANCH IF WE'RE TESTING CHECK BITS
030670  005737  002252                 TST     DBEMSK+2        ;LAST BIT IN MASK ?
030674  100405                         BMI     12$             ;BR IF BIT 31
030676                                 DLEFT   DBEMSK
030706  000641                         BR      4$
030710                          12$:    IF #SW11 SET.IN @SWR THEN GOTO 13$
030720                                  IF QVFLAG IS TRUE THEN GOTO 13$
030726  005737  002246                 TST     SBEMSK+2        ;LAST BIT IN SINGLE ERROR MASK ?
030732  100405                         BMI     13$             ;BR IF YES
030734                                 DLEFT   SBEMSK
030744  000615                         BR      3$
030746  105737  002256          13$:    TSTB    PASFLG  ;WHICH PASS
030752  001004                         BNE     14$             ;BR IF WE'RE DONE
030754  105237  002256                 INCB    PASFLG  ;INDICATE SECOND PASS COMING
                                       ;CLEAR OUT ANY DBE'S OR SBE'S
030760  000137  030356                 JMP     1$              ;GO DO IT!
030764  052737  000200  002256  14$:    BIS     #BIT7,PASFLG
030772  005037  002240                 CLR     TSTDAT
030776  005037  002242                 CLR     TSTDAT+2
031002  012737  000040  002244         MOV     #40,SBEMSK
031010  012737  000100  002250         MOV     #100,DBEMSK
031016  012737  003740  002274  15$:    MOV     #3740,CHECK
031024  013702  002244                 MOV     SBEMSK,R2
031030  074237  002274                 XOR     R2,CHECK
031034  013702  002250                 MOV     DBEMSK,R2
031040  074237  002274                 XOR     R2,CHECK
031044  006337  002250                 ASL     DBEMSK
031050  032737  020000  002250         BIT     #BIT13,DBEMSK
031056  001630                         BEQ     7$
031060  006337  002244                 ASL     SBEMSK
031064  032737  004000  002244         BIT     #BIT11,SBEMSK
031072  001006                         BNE     16$
031074  013737  002244  002250         MOV     SBEMSK,DBEMSK
031102  006337  002250                 ASL     DBEMSK
031106  000743                         BR      15$
031110  104471                  16$:    ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
031112  012701  002362                 MOV     #TESTADD,R1
031116                                 CLEAR   @(R1)+,@(R1)
031124  104503                         CLR1CSR                 ;CLEAR 1 SELECTED CSR
031126  000207                         RETURN
```

```
5330 031130                          MTP015: SUBTST  <<MTP015        WRITE INHIBIT OF BYTE WITH DBE>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MTP015  WRITE INHIBIT OF BYTE WITH DBE
                                     ;*****************************************************************************
5331                                                 ;CHECK FOR WRITE INHIBIT DURING A WRITE BYTE.
5332                                                 ;CHECKS FOR UNCORRECTED DATA.
5333 031130  005037  002234          1$:     CLR     DATBUF          ;INITIAL DATA
5334 031134  005037  002236                  CLR     DATBUF+2        ;32 BITS WORTH
5335 031140  012737  000001  002244  2$:     MOV     #1,SBEMSK       ;SINGLE ERROR MASK
5336 031146  005037  002246                  CLR     SBEMSK+2        ;
5337 031152  012737  000001  002250  3$:     MOV     #1,DBEMSK       ;DOUBLE ERROR MASK
5338 031160  005037  002252                  CLR     DBEMSK+2        ;
5339 031164  013737  002234  002240  4$:     MOV     DATBUF,TSTDAT   ;PRESERVE ORIG DATA
5340 031172  013737  002236  002242          MOV     DATBUF+2,TSTDAT+2
5341 031200  105737  002256                  TSTB    PASFLG  ;WHICH PASS ?
5342 031204  001404                          BEQ     5$              ;FIRST PASS, NO COMPLEMENTING
5343 031206  005137  002240                  COM     TSTDAT
5344 031212  005137  002242                  COM     TSTDAT+2        ;SECOND PASS, COMPLEMENT TSTDAT
5345 031216  104503                  5$:     CLR1CSR                 ;CLEAR 1 SELECTED CSR
5346 031220  023737  002244  002250          CMP     SBEMSK,DBEMSK   ;CHECK FOR SAME MASKS
5347 031226  001004                          BNE     6$              ;BR IF NOT EQUAL
5348 031230  023737  002246  002252          CMP     SBEMSK+2,DBEMSK+2       ;SECOND WORD ALSO
5349 031236  001474                          BEQ     11$             ;BR TO MAKE THEM NOT EQUAL
5350 031240  012737  002240  002272  6$:     MOV     #TSTDAT,SOURCE  ;ADDRESS FOR CHKGEN
5351 031246  004737  040360                  CALL    CHKGEN          ;GO GENERATE CHECK BITS
5352 031252  013701  002244                  MOV     SBEMSK,R1
5353 031256  074137  002240                  XOR     R1,TSTDAT
5354 031262  013701  002246                  MOV     SBEMSK+2,R1
5355 031266  074137  002242                  XOR     R1,TSTDAT+2
5356 031272  013701  002250                  MOV     DBEMSK,R1
5357 031276  074137  002240                  XOR     R1,TSTDAT
5358 031302  013701  002252                  MOV     DBEMSK+2,R1
5359 031306  074137  002242                  XOR     R1,TSTDAT+2
5360 031312  012701  002362          7$:     MOV     #TESTADD,R1     ;TEST LOCATION
5361 031316  104471                          ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
5362 031320  013731  002240                  MOV     TSTDAT,@(R1)+   ;WRITE FIRST 16 BITS
5363                                         ;LOAD CSR WITH IMAGE FROM R2
5364 031324  104475                          CB1CSR                  ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
5365 031326  013771  002242  000000          MOV     TSTDAT+2,@(R1)  ;WRITE SECOND 16 BITS + CHECKBITS
5366 031334  104503                          CLR1CSR                 ;CLEAR 1 SELECTED CSR
5367 031336  013702  002362                  MOV     TESTADD,R2      ;GET ADDRESS OF TEST LOC
5368 031342  010203                          MOV     R2,R3           ;R2 DESIGNATES FIRST BYTE
5369 031344  062703  000003                  ADD     #3,R3           ;R3 DESIGNATES LAST BYTE
5370 031350  112722  000360          8$:     MOVB    #360,(R2)+      ;TRY WRITING A BYTE
5371 031354  012701  002362                  MOV     #TESTADD,R1
5372 031360  017100  000000                  MOV     @(R1),R0
5373 031364  023700  002240                  CMP     TSTDAT,R0       ;CHECK FOR UNCHANGED DATA
5374 031370  001404                          BEQ     9$              ;BR IF OK
5375 031372  017137  000000  002032          MOV     @(R1),ADDRESS
5376 031400  104455                          PERR31
5377
5378 031402  017100  000002          9$:     MOV     @2(R1),R0
5379 031406  023700  002242                  CMP     TSTDAT+2,R0     ;READ SECOND WORD
5380 031412  001404                          BEQ     10$             ;BR IF UNCHANGED
5381 031414  017137  000002  002032          MOV     @2(R1),ADDRESS
5382 031422  104455                          PERR31
5383
```

```
5384 031424  020203                    10$:   CMP     R2,R3            ;TESTED LAST BYTE ?
5385 031426  001350                           BNE     8$               ;BR IF NO
5386 031430  105737  002256           11$:   TSTB    PASFLG
5387 031434  100452                           BMI     15$              ;BRANCH IF TESTING CHECK BITS
5388 031436  005737  002252                   TST     DBEMSK+2         ;CHECKING FOR LAST ERROR BIT
5389 031442  100405                           BMI     12$              ;BR IF DONE HERE
5390 031444                                    DLEFT   DBEMSK
5391 031454  000643                           BR      4$
5392 031456                           12$:   IF #SW11 SET.IN @SWR THEN GOTO 13$
5393 031466                                  IF QVFLAG IS TRUE THEN GOTO 13$
5394 031474  005737  002246                   TST     SBEMSK+2         ;LAST SBE MASK
5395 031500  100405                           BMI     13$              ;BR IF DONE WITH THIS PASS
5396 031502                                    DLEFT   SBEMSK
5397 031512  000617                           BR      3$
5398 031514  105737  002256           13$:   TSTB    PASFLG  ;TEST PASS FLAG
5399 031520  001003                           BNE     14$              ;NON ZERO MEANS WE'RE DONE
5400 031522  105237  002256                   INCB    PASFLG  ;NOT DONR
5401 031526  000600                           BR      1$
5402 031530  052737  000200  002256  14$:   BIS     #BIT7,PASFLG
5403 031536  005037  002240                   CLR     TSTDAT
5404 031542  005037  002242                   CLR     TSTDAT+2
5405 031546  012737  000040  002244           MOV     #40,SBEMSK
5406 031554  012737  000100  002250           MOV     #100,DBEMSK
5407 031562  012737  003740  002274  15$:   MOV     #3740,CHECK
5408 031570  013702  002244                   MOV     SBEMSK,R2
5409 031574  074237  002274                   XOR     R2,CHECK
5410 031600  013702  002250                   MOV     DBEMSK,R2
5411 031604  074237  002274                   XOR     R2,CHECK
5412 031610  006337  002250                   ASL     DBEMSK
5413 031614  032737  020000  002250           BIT     #BIT13,DBEMSK
5414 031622  001633                           BEQ     7$
5415 031624  006337  002244                   ASL     SBEMSK
5416 031630  032737  004000  002244           BIT     #BIT11,SBEMSK
5417 031636  001006                           BNE     16$
5418 031640  013737  002244  002250           MOV     SBEMSK,DBEMSK
5419 031646  006337  002250                   ASL     DBEMSK
5420 031652  000743                           BR      15$
5421 031654  104471                  16$:   ECC1DIS                  ;DISABLE ECC ON 1 SELECTED CSR
5422 031656  012701  002362                   MOV     #TESTADD,R1      ;TEST LOCATION
5423 031662                                    CLEAR   @(R1)+,@(R1)     ;TO ERASE ANY DBE'S FROM TESTING
5424                                            ;RESTORE CSR
5425 031670  104503                           CLR1CSR                  ;CLEAR 1 SELECTED CSR
5426 031672  000207                           RETURN
```

```
5429 031674                                MTP016: SUBTST  <<MTP016        WRITE INHIBIT OF WORD WITH DBE>>
                                           ;*********************************************************************************
                                           ;*SUBTEST       MTP016  WRITE INHIBIT OF WORD WITH DBE
                                           ;*********************************************************************************
5430                                               ;DOUBLE BIT ERROR WRITE CANCEL WITH
5431                                               ;WORD WRITE.
5432                                               ;CHECKS WRITE INHIBIT WITH WORD WRITES TO
5433                                               ;WORD WITH DOUBLE ERROR.
5434 031674  005037  002234          T12A:   CLR     DATBUF          ;BACKGROUND FOR DOUBLE ERRORS
5435 031700  005037  002236                  CLR     DATBUF+2        ;2 WORDS WORTH
5436 031704  012737  000001  002244        MOV     #1,SBEMSK       ;SINGLE ERROR MASK
5437 031712  005037  002246                  CLR     SBEMSK+2        ;
5438 031716  012737  000001  002250  T12B:   MOV     #1,DBEMSK       ;DOUBLE ERROR MASK
5439 031724  005037  002252                  CLR     DBEMSK+2
5440 031730  013737  002234  002240  1$:     MOV     DATBUF,TSTDAT   ;DATA FOR TEST
5441 031736  013737  002236  002242          MOV     DATBUF+2,TSTDAT+2;BOTH WORDS
5442 031744  105737  002256                  TSTB    PASFLG  ;COMP DATA ON SECOND PASS ONLY
5443 031750  001404                          BEQ     2$              ;BR IF FIRST PASS
5444 031752  005137  002240                  COM     TSTDAT          ;COMP FIRST WORD
5445 031756  005137  002242                  COM     TSTDAT+2        ;NOW SECOND WORD
5446 031762  023737  002244  002250  2$:     CMP     SBEMSK,DBEMSK   ;CHECK FOR IDENTICAL MASKS
5447 031770  001004                          BNE     3$              ;BR IF DIFFERENT
5448 031772  023737  002246  002252          CMP     SBEMSK+2,DBEMSK+2;UPPER WORD TOO
5449 032000  001502                          BEQ     8$              ;BR TO MAKE THEM NOT EQUAL
5450 032002  012737  002240  002272  3$:     MOV     #TSTDAT,SOURCE  ;NEED ADDR OF DATA FOR CHKGEN
5451 032010  004737  040360                  CALL    CHKGEN          ;GO GENERATE CHECK BITS
5452 032014  013701  002244                  MOV     SBEMSK,R1
5453 032020  074137  002240                  XOR     R1,TSTDAT
5454 032024  013701  002246                  MOV     SBEMSK+2,R1
5455 032030  074137  002242                  XOR     R1,TSTDAT+2
5456 032034  013701  002250                  MOV     DBEMSK,R1
5457 032040  074137  002240                  XOR     R1,TSTDAT
5458 032044  013701  002252                  MOV     DBEMSK+2,R1
5459 032050  074137  002242                  XOR     R1,TSTDAT+2
5460 032054  012701  002362          4$:     MOV     #TESTADD,R1     ;FIRST TEST ADRRESS
5461 032060  104471                          ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
5462 032062  013731  002240                  MOV     TSTDAT,@(R1)+   ;WRITE FIRST 16 BITS
5463 032066  104475                          CB1CSR                  ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
5464 032070  013771  002242  000000          MOV     TSTDAT+2,@(R1)  ;WRITE SECOND 16 BITS + CHECKBITS
5465 032076  105037  002257                  CLRB    UPPFLG          ;SET FOR 2 LOOPS
5466 032102  162701  000002                  SUB     #2,R1           ;POINT TO LOW WORD
5467 032106  104503                  5$:     CLR1CSR                 ;CLEAR 1 SELECTED CSR
5468 032110  012771  177400  000000          MOV     #177400,@(R1)   ;TRY WRITING LOCATION
5469 032116  012701  002362                  MOV     #TESTADD,R1
5470 032122  017100  000000                  MOV     @(R1),R0
5471 032126  023700  002240                  CMP     TSTDAT,R0       ;CHECK FOR ORIGINAL DATA
5472 032132  001404                          BEQ     6$              ;SHOULD BE UNCHANGED
5473 032134  017137  000000  002032          MOV     @(R1),ADDRESS
5474 032142  104455                          PERR31
5475
5476 032144  062701  000002          6$:     ADD     #2,R1
5477 032150  017100  000000                  MOV     @(R1),R0
5478 032154  023700  002242                  CMP     TSTDAT+2,R0     ;THIS SHOULD BE UNCHANGED ALSO
5479 032160  001404                          BEQ     7$
5480 032162  017137  000000  002032          MOV     @(R1),ADDRESS
5481 032170  104455                          PERR31
```

```
5484 032172  105737  002257          7$:     TSTB    UPPFLG          ;WHICH LOOP ?
5485 032176  001003                          BNE     8$              ;SECOND, BR OUT
5486 032200  105237  002257                  INCB    UPPFLG          ;FIRST, KEEP GOING
5487 032204  000740                          BR      5$
5488 032206  105737  002256          8$:     TSTB    PASFLG
5489 032212  100454                          BMI     12$
5490 032214  005737  002252                  TST     DBEMSK+2        ;LAST BIT ?
5491 032220  100405                          BMI     9$              ;MINUS = BIT 31
5492 032222                                  DLEFT   DBEMSK
5493 032232  000636                          BR      1$
5494 032234                          9$:     IF #SW11 SET.IN @SWR THEN GOTO 10$
5495 032244                                  IF QVFLAG IS TRUE THEN GOTO 10$
5496 032252  005737  002246                  TST     SBEMSK+2        ;LAST BIT IN THIS MASK ?
5497 032256  100406                          BMI     10$             ;BR IF LAST BIT
5498 032260                                  DLEFT   SBEMSK
5499 032270  000137  031716                  JMP     T12B
5500 032274  105737  002256          10$:    TSTB    PASFLG  ;FIRST PASS ?
5501 032300  001004                          BNE     11$             ;BR IF SECOND
5502 032302  105237  002256                  INCB    PASFLG  ;INDICATE SECOND PASS COMING
5503 032306  000137  031674                  JMP     T12A
5504 032312  052737  000200  002256  11$:    BIS     #BIT7,PASFLG
5505 032320  005037  002240                  CLR     TSTDAT
5506 032324  005037  002242                  CLR     TSTDAT+2
5507 032330  012737  000040  002244          MOV     #40,SBEMSK
5508 032336  012737  000100  002250          MOV     #100,DBEMSK
5509 032344  012737  003740  002274  12$:    MOV     #3740,CHECK
5510 032352  013702  002244                  MOV     SBEMSK,R2
5511 032356  074237  002274                  XOR     R2,CHECK
5512 032362  013702  002250                  MOV     DBEMSK,R2
5513 032366  074237  002274                  XOR     R2,CHECK
5514 032372  006337  002250                  ASL     DBEMSK
5515 032376  032737  020000  002250          BIT     #BIT13,DBEMSK
5516 032404  001623                          BEQ     4$
5517 032406  006337  002244                  ASL     SBEMSK
5518 032412  032737  004000  002244          BIT     #BIT11,SBEMSK
5519 032420  001006                          BNE     13$
5520 032422  013737  002244  002250          MOV     SBEMSK,DBEMSK
5521 032430  006337  002250                  ASL     DBEMSK
5522 032434  000743                          BR      12$
5523 032436  104471                  13$:    ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
5524 032440  012701  002362                  MOV     #TESTADD,R1     ;RESTORE TEST ADDRESS
5525 032444  005031                          CLR     @(R1)+          ;CLEAR ANY DBE'S FROM TEST
5526 032446  005071  000000                  CLR     @(R1)
5527 032452  104503                          CLR1CSR                 ;CLEAR 1 SELECTED MK11 CSR
5528 032454  000207                          RETURN
```

```
                                MTP017: SUBTST  <<MTP017        HOLDING 1'S & 0'S TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MTP017  HOLDING 1'S & 0'S TEST
                                ;*****************************************************************************
                                        ;*(1)    THIS TEST CHECKS THE MEMORY FOR THE CAPABILITY
                                        ;*       OF HOLDING 1'S AND 0'S BY WRITING A BACKGROUND
                                        ;*       OF 000377 AND READING IT
                                        ;*(2)    MEMORY IS WRITTEN USING A BYTE AT A TIME
                                        ;*(3)    STEPS 1 & 2 ARE REPEATED WITH A SWAPPED BACKGROUND PATTERN
                                        ;NOTE:   THIS TEST WRITES BYTES & READS WORDS
032456  012701  060000                  MOV     #FIRST,R1
032462  010104                          MOV     R1,R4
032464  012705  160000                  MOV     #LAST+2,R5
032470  012700  000377                  MOV     #377,R0         ;GET THE PATTERN INTO R0
032474  010003                          MOV     R0,R3
032476  000303                          SWAB    R3
032500  110021                  1$:     MOVB    R0,(R1)+        ;WRITE A BYTE
032502  110321                          MOVB    R3,(R1)+        ;WRITE THE MEMORY WITH THE BYTE STORED IN BAKPAT+1
032504  020105                          CMP     R1,R5           ;COMPARE TEST LOC TO TOP + 2
032506  103774                          BLO     1$              ;BRANCH IF LOWER
032510  014102                  2$:     MOV     -(R1),R2
032512  020002                          CMP     R0,R2           ;TEST THE MEMORY TO SEE IF IT CONTAINS
                                                                ;THE WORD STORED IN BAKPAT
032514  001401                          BEQ     3$
032516  104446                          PERR22
032520  020104                  3$:     CMP     R1,R4           ;KEEP ON TESTING THE MEMORY UNTIL
032522  101372                          BHI     2$              ;R1 EQUALS THE LOWEST ADDRESS
032524  000303                          SWAB    R3              ;CHANGE THE DATA PATTERN
032526  000300                          SWAB    R0
032530  001763                          BEQ     1$              ;IF THE DATA PATTERN DOES NOT HAVE LOW
                                                                ; BYTE -0 THEN FALL THRU
032532  000207                          RETURN
```

```
 5564 032534                    MTP020: SUBTST  <<MTP020        MARCHING 1'S & 0'S IN CHECK BITS TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST       MTP020  MARCHING 1'S & 0'S IN CHECK BITS TEST
                                ;*****************************************************************************
 5565                                   ;*THIS TEST IS CONCERNED ONLY WITH THE INTEGRITY
 5566                                   ;*OF THE MOS RAMS THAT STORE THE CHECKBITS.
 5567
 5568                                   ;077 --> 100 DOWN
 5569 032534  160201           MTPA20: SUB     R2,R1           ;V177640
 5570 032536  160204                   SUB     R2,R4           ;V177642
 5571 032540  005711                   TST     (R1)            ;V177644        ;1ST WORD OK?
 5572 032542  001002                   BNE     1$              ;V177646        ;NO - SKIP
 5573 032544  005714                   TST     (R4)            ;V177650        ;2ND WORD OK?
 5574 032546  001401                   BEQ     2$              ;V177652        ;YES - SKIP
 5575 032550  104453           1$:     PERR27                  ;V177654        ;GOOD=000000,,000000,,077
 5576 032552  010314           2$:     MOV     R3,(R4)         ;V177656        ;2ND WORD <= 100000
 5577 032554  005011                   CLR     (R1)            ;V177660        ;CLEAR 1ST WORD
 5578 032556  020100                   CMP     R1,R0           ;V177662        ;ARE WE DONE?
 5579 032560  002365                   BGE     MTPA20          ;V177664        ;BRANCH IF NOT
 5580 032562  000207                   RETURN                  ;V177666
 5581
 5582                                   ;100 --> 077 UP
 5583 032564  005711           MTPB20: TST     (R1)            ;V177640        ;1ST WORD OK?
 5584 032566  001002                   BNE     3$              ;V177642        ;NO - SKIP
 5585 032570  020314                   CMP     R3,(R4)         ;V177644        ;2ND WORD OK?
 5586 032572  001401                   BEQ     4$              ;V177646        ;YES - SKIP
 5587 032574  104452           3$:     PERR26                  ;V177650        ;GOOD=000000,,100000,,100
 5588 032576  005014           4$:     CLR     (R4)            ;V177652        ;CLEAR 2ND WORD
 5589 032600  005011                   CLR     (R1)            ;V177654        ;CLEAR 1ST WORD
 5590 032602  060201                   ADD     R2,R1           ;V177656
 5591 032604  060204                   ADD     R2,R4           ;V177660
 5592 032606  020405                   CMP     R4,R5           ;V177662        ;TOP + 2 YET?
 5593 032610  001365                   BNE     MTPB20          ;V177664        ;NO - LOOP
 5594 032612  000207                   RETURN                  ;V177666
 5595
 5596                                   ;077 --> 100 UP
 5597 032614  005711           MTPC20: TST     (R1)            ;V177640        ;1ST WORD OK?
 5598 032616  001002                   BNE     5$              ;V177642        ;NO - SKIP
 5599 032620  005714                   TST     (R4)            ;V177644        ;2ND WORD OK?
 5600 032622  001401                   BEQ     6$              ;V177646        ;YES - SKIP
 5601 032624  104453           5$:     PERR27                  ;V177650        ;GOOD=000000,,000000,,077
 5602 032626  010314           6$:     MOV     R3,(R4)         ;V177652        ;WRITE 1ST WORD
 5603 032630  005011                   CLR     (R1)            ;V177654        ;WRITE 2ND WORD
 5604 032632  060204                   ADD     R2,R4           ;V177656
 5605 032634  060201                   ADD     R2,R1           ;V177660
 5606 032636  020405                   CMP     R4,R5           ;V177662        ;TOP + 2 YET?
 5607 032640  001365                   BNE     MTPC20          ;V177664        ;NO - LOOP
 5608 032642  000207                   RETURN                  ;V177666
```

```
 5611                                          ;100 --> 077 DOWN
 5612 032644  160201                MTPD20:    SUB     R2,R1           ;V177640
 5613 032646  160204                           SUB     R2,R4           ;V177642
 5614 032650  020314                           CMP     R3,(R4)         ;V177644        ;2ND WORD OK?
 5615 032652  001002                           BNE     7$              ;V177646        ;NO - SKIP
 5616 032654  005711                           TST     (R1)            ;V177650        ;1ST WORD OK?
 5617 032656  001401                           BEQ     8$              ;V177652        ;YES - SKIP
 5618 032660  104452                7$:        PERR26                  ;V177654        ;GOOD=000000,,100000,,100
 5619 032662  005014                8$:        CLR     (R4)            ;V177656        ;WRITE 1ST WORD
 5620 032664  005011                           CLR     (R1)            ;V177660        ;WRITE 2ND WORD
 5621 032666  020100                           CMP     R1,R0           ;V177662
 5622 032670  002365                           BGE     MTPD20          ;V177664
 5623 032672  000207                           RETURN                  ;V177666
 5624
 5625                                          ;077 UP
 5626 032674  005711                MTPE20:    TST     (R1)            ;V177640        ;1ST WORD OK?
 5627 032676  001002                           BNE     9$              ;V177642        ;NO - SKIP
 5628 032700  005714                           TST     (R4)            ;V177644        ;2ND WORD OK?
 5629 032702  001401                           BEQ     10$             ;V177646        ;YES - SKIP
 5630 032704  104453                9$:        PERR27                  ;V177650        ;GOOD=000000,,000000,,077
 5631 032706  060201                10$:       ADD     R2,R1           ;V177652
 5632 032710  060204                           ADD     R2,R4           ;V177654
 5633 032712  020405                           CMP     R4,R5           ;V177656        ;TOP + 2 YET?
 5634 032714  001367                           BNE     MTPE20          ;V177660        ;NO - LOOP
 5635 032716  000207                           RETURN                  ;V177662
```

```
                                MTPA21: SUBTST  <<MTPA21        MARCHING 1'S & 0'S PATTERN TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST       MTPA21  MARCHING 1'S & 0'S PATTERN TEST
                                ;*****************************************************************************
                                        ;READ,BYTESWAP-MODIFY,READ,DOWN
032720  014100                  1$:     MOV     -(R1),R0;V177640
032722  020200                          CMP     R2,R0   ;V177642
032724  001401                          BEQ     2$      ;V177644
032726  104443                          PERR17          ;V177646

032730  000311                  2$:     SWAB    (R1)    ;V177650
032732  011100                          MOV     (R1),R0 ;V177652
032734  020300                          CMP     R3,R0   ;V177654
032736  001401                          BEQ     3$      ;V177656
032740  104444                          PERR20          ;V177660

032742  020401                  3$:     CMP     R4,R1   ;V177662        ;DONE?
032744  001365                          BNE     1$      ;V177664        ;NO - LOOP
032746  000207                          RETURN          ;V177666        ;YES - RETURN

032750                          MTPB21: ;READ,BYTESWAP-MODIFY,READ,UP
032750  011100                  1$:     MOV     (R1),R0 ;V177640
032752  020300                          CMP     R3,R0   ;V177642
032754  001401                          BEQ     2$      ;V177644
032756  104444                          PERR20          ;V177646

032760  000311                  2$:     SWAB    (R1)    ;V177650
032762  011100                          MOV     (R1),R0 ;V177652
032764  020200                          CMP     R2,R0   ;V177654
032766  001401                          BEQ     3$      ;V177656
032770  104443                          PERR17          ;V177660

032772  062701  000002          3$:     ADD     #2,R1   ;V177662
032776  020501                          CMP     R5,R1   ;V177666        ;DONE?
033000  001363                          BNE     1$      ;V177670        ;NO - LOOP
033002  000207                          RETURN          ;V177672        ;YES - RETURN

033004                          MTPC21: ;READ,BYTESWAP-MODIFY,READ,UP
033004  011100                  1$:     MOV     (R1),R0 ;V177640
033006  020200                          CMP     R2,R0   ;V177642
033010  001401                          BEQ     2$      ;V177644
033012  104443                          PERR17          ;V177646

033014  000311                  2$:     SWAB    (R1)    ;V177650
033016  011100                          MOV     (R1),R0 ;V177652
033020  020300                          CMP     R3,R0   ;V177654
033022  001401                          BEQ     3$      ;V177656
033024  104444                          PERR20          ;V177660

033026  062701  000002          3$:     ADD     #2,R1   ;V177662
033032  020501                          CMP     R5,R1   ;V177666        ;DONE?
033034  001363                          BNE     1$      ;V177670        ;NO - LOOP
033036  000207                          RETURN          ;V177672        ;YES - RETURN
```

```
 5690 033040                        MTPD21: ;READ,BYTESWAP-MODIFY,READ,DOWN
 5691 033040  014100                1$:     MOV     -(R1),R0;V177640
 5692 033042  020300                        CMP     R3,R0   ;V177642
 5693 033044  001401                        BEQ     2$      ;V177644
 5694 033046  104444                        PERR20          ;V177646
 5695
 5696 033050  000311                2$:     SWAB    (R1)    ;V177650
 5697 033052  011100                        MOV     (R1),R0 ;V177652
 5698 033054  020200                        CMP     R2,R0   ;V177654
 5699 033056  001401                        BEQ     3$      ;V177656
 5700 033060  104443                        PERR17          ;V177660
 5701
 5702 033062  020401                3$:     CMP     R4,R1   ;V177662        ;DONE?
 5703 033064  001365                        BNE     1$      ;V177664        ;NO - LOOP
 5704 033066  000207                        RETURN          ;V177666        ;YES - RETURN
 5705
```

```
5708 033070                          MTP022: SUBTST  <<MTP022        REFRESH & SHIFTING DIAGONAL TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MTP022  REFRESH & SHIFTING DIAGONAL TEST
                                     ;*****************************************************************************
5709                                         ;(1)    WE WRITE A DIAGONAL PATTERN IN MEMORY (WITH CACHE ON).
5710                                         ;(2)    IF A REFRESH TEST WE DISTURB ALL ROWS FOR > 2 MS (WITH CACHE ON).
5711                                         ;(3)    WE READ & CHECK FOR CORRECTNESS THE DIAGONAL PATTERN
5712                                         ;       (WITH CACHE OFF).
5713         000010                  KDIAG=8.                ;HOW OFTEN A DIAGONAL STRIPE OCCURS (MUST BE A POWER OF 2)
5714 033070                                  FOR EVEN := #1 TO #2                    ;FOR DATA & COMPLEMENT DATA
5715 033076                                    IF EVEN EQ #1
5716 033106                                      LET R2 := ZEROS
5717 033112                                      LET R3 := ONES
5718 033116                                    ELSE
5719 033120                                      LET R2 := ONES
5720 033124                                      LET R3 := ZEROS
5721 033130                                    END ;OF IF EVEN
5722 033130                                    FOR STRIPES := #0 TO #KDIAG-1         ;FOR THE NUMBER OF STRIPES
5723
5724                                             ;WRITE LOOP
5725 033134  104423                             CACHON                          ;TURN CACHE ON
5726 033136                                     LET COUNT := STRIPES
5727 033144                                     LET R1 := #FIRST
5728 033150                                     WHILE R1 LOS #LAST
5729 033156                                       IF COUNT LT #0 THEN LET COUNT := #KDIAG-1
5730 033172                                       IF #374 OFF.IN R1 THEN LET COUNT := COUNT - #1
5731 033204                                       IF COUNT NE #0
5732 033212                                         LET (R1) := R2
5733 033214                                         LET 2(R1) := R2
5734 033220                                       ELSE
5735 033222                                         LET (R1) := R3
5736 033224                                         LET 2(R1) := R3
5737 033230                                       END ;OF IF COUNT
5738 033230                                       LET COUNT := COUNT - #1
5739 033234                                       LET R1 := R1 + #4
5740 033240                                     END ;OF WHILE
5741                                             ;END OF WRITE LOOP
5742
5743 033242                                     IF DIAGFLAG IS FALSE THEN $CALL REFRESH
5744                                             ;READ LOOP
5745 033254                                     LET COUNT := STRIPES
5746 033262                                     LET R1 := #FIRST
5747 033266  104424                             CACHOFF                         ;TURN CACHE OFF
```

```
5749 033270                                     WHILE R1 LOS #LAST
5750 033276                                       IF COUNT LT #0 THEN LET COUNT := #KDIAG-1
5751 033312                                       IF #374 OFF.IN R1 THEN LET COUNT := COUNT - #1
5752 033324                                       IF COUNT NE #0
5753 033332                                         LET R0 := (R1)
5754 033334                                         IF R2 NE R0
5755 033340  104443                                   PERR17
5756 033342                                         END ;OF IF R2
5757 033342                                         LET R0 := 2(R1)
5758 033346                                         IF R2 NE R0
5759 033352  104443                                   PERR17
5760 033354                                         END ;OF IF R2
5761 033354                                       ELSE
5762 033356                                         LET R0 := (R1)
5763 033360                                         IF R3 NE R0
5764 033364  104444                                   PERR20
5765 033366                                         END ;OF IF R3
5766 033366                                         LET R0 := 2(R1)
5767 033372                                         IF R3 NE R0
5768 033376  104444                                   PERR20
5769 033400                                         END ;OF IF R3
5770 033400                                       END ;OF IF COUNT
5771 033400                                       LET COUNT := COUNT - #1
5772 033404                                       LET R1 := R1 + #4
5773 033410                                     END ;OF WHILE
5774                                            ;END OF READ LOOP
5775
5776 033412                                   END ;OF FOR STRIPES
5777 033426                                 END ;OF FOR EVEN
5778 033442  000207                         RETURN
5779
5780 033444                         REFRESH:SUBTST  <<SUBR  REFRESH DELAY>>
                                    ;*****************************************************************************
                                    ;*SUBTEST        SUBR    REFRESH DELAY
                                    ;*****************************************************************************
5781                                        ;DISTURB EACH ROW FOR > 3.2 MS
5782 033444                                 FOR R0 := #FIRST TO #FIRST+374 BY #4
5783 033450  004737  033514                   CALL REFSUB
5784 033454                                 END ;OF FOR R0
5785 033466                                 LET R0 := #FIRST+BIT14
5786 033472                                 WHILE R0 LOS #LAST+BIT14+374
5787 033500  004737  033514                   CALL REFSUB
5788 033504                                   LET R0 := R0 + #4
5789 033510                                 END ;OF WHILE
5790 033512  000207                         RETURN
5791 033514  012704  000640         REFSUB: MOV     #640,R4                 ;TIME FOR A > 3.2 MS LOOP
5792 033520  062700  000002                 ADD     #2,R0
5793 033524  005140                 1$:     COM     -(R0)
5794 033526  005120                         COM     (R0)+
5795 033530  005110                         COM     (R0)
5796 033532  005110                         COM     (R0)
5797 033534  077405                         SOB     R4,1$
5798 033536  162700  000002                 SUB     #2,R0
5799 033542  000207                         RETURN
```

```
 5803 033544                    MTPA24: SUBTST  <<MTPA24        FAST GALLOPING PATTERN TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST       MTPA24  FAST GALLOPING PATTERN TEST
                                ;*****************************************************************************
 5804                                   ;THE TOTAL TEST (INCLUDING SETUP) IS AS FOLLOWS
 5805                                   ;*(1)    THIS TEST WRITES THE MEMORY WITH A BACK GROUND PATTERN
 5806                                   ;*       STORED AT LOCATION BAKPAT
 5807                                   ;*(2)    TEST BEGINS AT LOWEST LOCATION BEING TESTED
 5808                                   ;*       (LETS NAME IT 'A')
 5809                                   ;*(3)    LETS NAME THE 1ST LOCATION IN THE ROW/COLUMN UNDER TEST AS 'B'.
 5810                                   ;*(4)    SWAPS BYTES FOR LOCATION 'A'.
 5811                                   ;*(5)    READS 'A', READS 'B'
 5812                                   ;*(6)    'B' = 'B'+400   (ADDS 64 DOUBLE WORDS TO 'B')
 5813                                   ;*(7)    REPEATS STEPS 5 AND 6 UNTIL 'B' IS GREATER THAN THE
 5814                                   ;*(8)    END OF THE BANK  A+2
 5815                                   ;*(9)    REPEATS  STEPS 3-8 UNTILL 'A' REACHES THE END OF THE BANK
 5816                                   ;*(10)   AFTER EXECUTING THE TEST  DATA IS COMPLEMENTED
 5817                                   ;*       AND STEPS 1-9 ARE REPEATED
 5818                                   ;REGISTERS ARE USED AS FOLLOWS
 5819                                   ;R0      TEST DATA
 5820                                   ;R1      'A'
 5821                                   ;R2      'B'
 5822                                   ;R3      BAKPAT
 5823                                   ;R4      SWAPAT
 5824                                   ;R5      LAST
 5825
 5826                                   ;NOTE THE PATTERN STARTS AT MTPB24!.!!!!.!!.!!..!
 5827
 5828                                   ;UIPAR'S
 5829 033544  011100            1$:     MOV     (R1),R0         ;V177640        ;READ 'A'
 5830 033546  020004                    CMP     R0,R4           ;V177642        ;CHECK 'A'
 5831 033550  001401                    BEQ     2$              ;V177644        ;BR IF OK
 5832 033552  104447                    PERR23                  ;V177646        ;REPORT ERROR
 5833
 5834 033554  011200            2$:     MOV     (R2),R0         ;V177650        ;READ 'B'
 5835 033556  020003                    CMP     R0,R3           ;V177652        ;CHECK 'B'
 5836 033560  001401                    BEQ     3$              ;V177654        ;BR IF OK
 5837 033562  104450                    PERR24                  ;V177656        ;REPORT ERROR
 5838
 5839 033564  062702  000400    3$:     ADD     #400,R2         ;V177660        ;BUMP 'B'
 5840 033570  020205                    CMP     R2,R5           ;V177664        ;AT END YET?
 5841 033572  101764                    BLOS    1$              ;V177666        ;BR IF NO
 5842
 5843 033574  062701  000002            ADD     #2,R1           ;V177670        ;BUMP 'A'
 5844 033600  000137  033604            JMP     @#MTPB24        ;V177674        ;GOTO V177260
```

```
5847 033604                          MTPB24: SUBTST  <<MTPB24        FAST GALLOP PART B>>
                                     ;*********************************************************************************
                                     ;*SUBTEST        MTPB24  FAST GALLOP PART B
                                     ;*********************************************************************************
5848                                         ;SDPAR'S
5849 033604  010411                          MOV     R4,(R1)         ;V172260        ;WRITE 'A'
5850 033606  020105                          CMP     R1,R5           ;V172262        ;DONE?
5851 033610  001001                          BNE     1$              ;V172264        ;BR IF NO
5852 033612  000207                          RETURN                  ;V172266        ;YES - RETURN
5853 033614  000137  033620        1$:       JMP     @#MTPC24        ;V172270        ;GOTO V172360
5854
5855 033620                          MTPC24: SUBTST  <<MTPC24        FAST GALLOP PART C>>
                                     ;*********************************************************************************
                                     ;*SUBTEST        MTPC24  FAST GALLOP PART C
                                     ;*********************************************************************************
5856                                         ;KDPAR'S
5857 033620  010102                          MOV     R1,R2           ;V172360        ;RESET 'B' <--- 'A'
5858 033622  011100                          MOV     (R1),R0         ;V172362        ;READ 'A'
5859 033624  020004                          CMP     R0,R4           ;V172364        ;CHECK 'A'
5860 033626  001401                          BEQ     1$              ;V172366        ;BR IF OK
5861 033630  104447                          PERR23                  ;V172370        ;REPORT ERROR
5862 033632  000137  033564        1$:       JMP     @#MTPA24+20     ;V172372        ;GOTO V177660
```

```
 5865 033636                                MTP025: SUBTST  <<MTP025        INTERRUPT ENABLE TEST>>
                                                    ;*************************************************************************
                                                    ;*SUBTEST       MTP025  INTERRUPT ENABLE TEST
                                                    ;*************************************************************************
 5866 033636  005037  002240                        CLR     TSTDAT          ;GENERATE CHECKBITS ON 0,,0
 5867 033642  005037  002242                        CLR     TSTDAT+2
 5868 033646  012737  002240  002272                MOV     #TSTDAT,SOURCE
 5869 033654  004737  040360                        CALL    CHKGEN
 5870 033660  012737  000003  002074                MOV     #3,NOPAR                ;SETUP PARITY ACTION
 5871 033666  012701  002362                        MOV     #TESTADD,R1     ;FIRST TEST ADDRESS
 5872 033672  012737  033732  002264                MOV     #1$,PARTHERE    ;SETUP TRAP DESTINATION
 5873 033700  004737  034154                        CALL    MTPA25          ;WRITE DATA & CHECKBITS
 5874 033704  104473                                ECC1INIT                ;INITIALIZE 1 SELECTED MK11 CSR
 5875 033706  005771  000000                        TST     @(R1)           ;ACCESS LOCATIONS FOR DBE TRAPS
 5876 033712  005771  000002                        TST     @2(R1)
 5877                                               ;NONE - GOOD - ACCESS FOR SBE TRAPS
 5878 033716  104507                                ENA1SBE                 ;DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
 5879 033720  005771  000000                        TST     @(R1)
 5880 033724  005771  000002                        TST     @2(R1)
 5881 033730  000404                                BR      2$              ;NONE - GOOD - SKIP
 5882 033732  104426                         1$:    READCSR
 5883 033734                                        FATAL   27
 5884 033742  005237  002240                  2$:   INC     TSTDAT          ;CHECK FOR CORRECT ACTION ON SBE'S
 5885 033746  004737  034102                        CALL    MTPD25          ;IN ALL 4 BYTES
 5886 033752  012737  000400  002240                MOV     #400,TSTDAT
 5887 033760  004737  034102                        CALL    MTPD25
 5888 033764  005037  002240                        CLR     TSTDAT
 5889 033770  005237  002242                        INC     TSTDAT+2
 5890 033774  004737  034102                        CALL    MTPD25
 5891 034000  012737  000400  002242                MOV     #400,TSTDAT+2
 5892 034006  004737  034102                        CALL    MTPD25
 5893
 5894 034012  005037  002242                        CLR     TSTDAT+2        ;CHECK FOR CORRECT ACTION ON DBE'S
 5895 034016  012737  000003  002240                MOV     #3,TSTDAT       ;IN ALL 4 BYTES
 5896 034024  004737  034124                        CALL    MTPE25
 5897 034030  012737  001400  002240                MOV     #1400,TSTDAT
 5898 034036  004737  034124                        CALL    MTPE25
 5899 034042  005037  002240                        CLR     TSTDAT
 5900 034046  012737  000003  002242                MOV     #3,TSTDAT+2
 5901 034054  004737  034124                        CALL    MTPE25
 5902 034060  012737  001400  002242                MOV     #1400,TSTDAT+2
 5903 034066  004737  034124                        CALL    MTPE25
 5904 034072  104503                                CLR1CSR                 ;CLEAR 1 SELECTED MK11 CSR
 5905 034074  005037  002074                        CLR     NOPAR           ;INDICATE PARITY ACTION
 5906 034100  000207                                RETURN
 5907
 5908 034102  004737  034154          MTPD25:       CALL    MTPA25          ;WRITE DATA & CHECKBITS
 5909 034106  104471                                ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
 5910 034110  004737  034176                        CALL    MTPB25          ;CHECK FOR NO TRAPS
 5911 034114  104507                                ENA1SBE                 ;DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
 5912 034116  004737  034236                        CALL    MTPC25          ;CHECK FOR EXPECTED TRAP
 5913 034122  000207                                RETURN
```

```
                                        MTPE25: CALL    MTPA25          ;WRITE DATA & CHECKBITS
                                                ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
                                                CALL    MTPB25          ;CHECK FOR NO TRAPS
                                                ;ENABLE DBE TRAPS
                                                ECC1INIT                ;INITIALIZE 1 SELECTED MK11 CSR
                                                CALL    MTPC25          ;CHECK FOR EXPECTED TRAP
                                                ENA1SBE                 ;DISABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
                                                CALL    MTPC25          ;CHECK FOR EXPECTED TRAP
                                                RETURN

                                                ;WRITE TSTDAT & TSTDAT+2 & CHECKBITS
                                        MTPA25: ECC1DIS                 ;DISABLE ECC ON 1 SELECTED CSR
                                                MOV     TSTDAT,@(R1)    ;WRITE FIRST 16 BITS
                                                CB1CSR                  ;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
                                                MOV     TSTDAT+2,@2(R1) ;WRITE 2ND 16 BITS & CHECKBITS
                                                RETURN

                                                ;CHECK FOR NO TRAP OCCURING CONDITION
                                        MTPB25: MOV     #1$,PARTHERE    ;SETUP TRAP DESTINATION
                                                TST     @(R1)           ;ACCESS LOCATIONS
                                                TST     @2(R1)
                                                RETURN                  ;NO TRAP - GOOD - RETURN

                                        1$:     READCSR
                                                MOV     (R1),ADDRESS    ;SAVE VIRTUAL ADDRESS
                                                ERROR   +24
                                                SET     HEADER
                                                RETURN

                                                ;TRAP SHOULD OCCURE TEST
                                        MTPC25: MOV     #1$,PARTHERE    ;SETUP TRAP DESTINATION
                                                TST     @(R1)           ;ACCESS 1ST LOCATION
                                                BR      2$              ;NO TRAP - BAD NEWS - SKIP
                                        1$:     MOV     #3$,PARTHERE    ;SETUP TRAP DESTINATION
                                                TST     @2(R1)          ;ACCESS 2ND LOCATION
                                        2$:     READCSR                 ;NO TRAP - BAD NEWS
                                                MOV     (R1),ADDRESS    ;SAVE VIRTUAL ADDRESS
                                                ERROR   +25
                                                SET     HEADER
                                        3$:     RETURN
```

```
5959 034304                          MTPA26: SUBTST  <<MTPA26        RANDOM DATA (WRITE)>>
                                     ;*********************************************************************************
                                     ;*SUBTEST        MTPA26  RANDOM DATA (WRITE)
                                     ;*********************************************************************************
5960 034304  000137  034354          1$:     JMP     @#MTPC26        ;V177640        GOTO V172360
5961 034310  010221                          MOV     R2,(R1)+        ;V177644
5962 034312  010321                          MOV     R3,(R1)+        ;V177646
5963 034314  077005                          SOB     R0,1$           ;V177650
5964 034316  000207                          RETURN                  ;V177652
5965
5966 034320                          MTPB26: SUBTST  <<MTPB26        RANDOM DATA (READ)>>
                                     ;*********************************************************************************
                                     ;*SUBTEST        MTPB26  RANDOM DATA (READ)
                                     ;*********************************************************************************
5967                                         .DSABL  AMA
5968                                         .ENABL  LSB
5969 034320  000137  034354          1$:     JMP     @#MTPC26        ;V177640        GOTO V172360
5970 034324  020221                          CMP     R2,(R1)+        ;V177644
5971 034326  001401                          BEQ     2$              ;V177646
5972 034330  104451                          PERR25                  ;V177650
5973 034332  005127                  2$:     COM     (PC)+           ;V177652
5974 034334  000000                  RANODD: 0                       ;V177654        FOR ERROR REPORTING
5975 034336  020321                          CMP     R3,(R1)+        ;V177656
5976 034340  001401                          BEQ     3$              ;V177660
5977 034342  104451                          PERR25                  ;V177662
5978 034344  005167  177764          3$:     COM     RANODD          ;V177664
5979 034350  077015                          SOB     R0,1$           ;V177670
5980 034352  000207                          RETURN                  ;V177672
5981                                         .DSABL  LSB
5982                                         .ENABL  AMA
5983
5984 034354                          MTPC26: SUBTST  <<RANDOM NUMBER SUBPROGRAM>>
                                     ;*********************************************************************************
                                     ;*SUBTEST        RANDOM NUMBER SUBPROGRAM
                                     ;*********************************************************************************
5985                                         ;CALLER MUST SETUP
5986                                         ;       MOV     SEEDLO,R3
5987                                         ;       MOV     SEEDHI,R2
5988                                         ;       MOV     R3,R5
5989                                         ;       MOV     R2,R4
5990 034354  073427  000007                  ASHC    #7,R4           ;V172360
5991 034360  060305                          ADD     R3,R5           ;V172364
5992 034362  005504                          ADC     R4              ;V172366
5993 034364  060204                          ADD     R2,R4           ;V172370
5994 034366  062705  001057                  ADD     #1057,R5        ;V172372
5995 034372  000240                          NOP                     ;V172376        GOTO V172260
5996
5997 034374                          MTPD26: SUBTST  <<RANDOM NUMBER SUBSUBPROGRAM>>
                                     ;*********************************************************************************
                                     ;*SUBTEST        RANDOM NUMBER SUBSUBPROGRAM
                                     ;*********************************************************************************
5998 034374  005504                          ADC     R4              ;V172260
5999 034376  062704  047401                  ADD     #47401,R4       ;V172262
6000 034402  010503                          MOV     R5,R3           ;V172266
6001 034404  010402                          MOV     R4,R2           ;V172270
6002 034406  000137  034310                  JMP     @#MTPA26+4      ;V172272        GOTO V177644
```

```
 6005 034412                    MTP030: SUBTST  <<MT0030        FLUSH OUT DBE'S>>
                                ;*********************************************************************************
                                ;*SUBTEST        MT0030  FLUSH OUT DBE'S
                                ;*********************************************************************************
 6006 034412  011002            1$:     MOV     (R0),R2         ;V177640
 6007 034414  010220                    MOV     R2,(R0)+        ;V177642
 6008 034416  077103                    SOB     R1,1$           ;V177644
 6009 034420  000207                    RETURN                  ;V177646
 6010
 6011 034422                    MTP031: SUBTST  <<MTP031        SOB-A-LONG TEST>>
                                ;*********************************************************************************
                                ;*SUBTEST        MTP031  SOB-A-LONG TEST
                                ;*********************************************************************************
 6012                                   .DSABL  AMA
 6013 034422  000000                    0                       ;MOVE TERMINATOR
 6014 034424  077001            1$:     SOB     R0,1$           ;SOB TILL R0 UNDERFLOWS
 6015 034426  005167  177772            COM     1$              ;WRITE COMPLEMENT OF SOB
 6016 034432  020167  177766            CMP     R1,1$           ;READ & CHECK FOR NOT ''SOB R0,DOT''
 6017 034436  001403                    BEQ     2$              ;OK - SKIP
 6018 034440  104454                    PERR30
 6019 034442  010167  177756            MOV     R1,1$
 6020 034446  005167  177752    2$:     COM     1$              ;CORRECT SOB INSTRUCTION
 6021 034452  010200                    MOV     R2,R0           ;REINITIALIZE SOB CONSTANT
 6022                                   ;UPDATE MOVE REGISTERS
 6023 034454  010503                    MOV     R5,R3
 6024 034456  005725                    TST     (R5)+           ;BUMP (SAFELY) BY 2
 6025 034460  010504                    MOV     R5,R4
 6026 034462  020537  002472            CMP     R5,@#LINK1      ;DONE?
 6027 034466  001001                    BNE     3$              ;NO - SKIP
 6028 034470  000207                    RETURN                  ;YES
 6029
 6030 034472  014344            3$:     MOV     -(R3),-(R4)
 6031 034474  001376                    BNE     3$
 6032 034476  000752                    BR      1$
 6033         000056            SOBLENGTH=.-MTP031
 6034                                   .ENABL  AMA
```

```
6062 034500                          MTP032: SUBTST  <<MTP032        WRITE RECOVERY TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MTP032  WRITE RECOVERY TEST
                                     ;*****************************************************************************
6063                                         ;THE TEST ACTUALLY EXECUTED ALREADY IN THE MEMORY UNDER TEST.
6064                                         ;THIS CODE INSURES THAT IT CHANGED MEMORY TO HAVE
6065                                         ;1/2 BANK OF #5141      WHICH IS A "COM -(R1)" INSTRUCTION AND
6066                                         ;1/2 BANK OF #110       WHICH IS A "JMP (R0)"   INSTRUCTION.
6067
6068 034500  012401                  1$:    MOV     (R4)+,R1        ;V177640        ;GET DATA FROM LOWER 1/2 BANK
6069 034502  020102                         CMP     R1,R2           ;V177642        ;IS IT #5141?
6070 034504  001401                         BEQ     2$              ;V177644        ;YES - SKIP
6071 034506  104430                         PERR02                  ;V177646        ;NO - TAKE ERROR TRAP
6072 034510  077305                  2$:    SOB     R3,1$           ;V177650        ;LOOP FOR 1/2 BANK
6073 034512  013703  002472                 MOV     @#LINK1,R3      ;V177652        ;RESTORE LOOP SIZE
6074 034516  012400                  3$:    MOV     (R4)+,R0        ;V177656        ;GET DATA FROM UPPER 1/2 BANK
6075 034520  020005                         CMP     R0,R5           ;V177660        ;IS IT #110?
6076 034522  001401                         BEQ     4$              ;V177662        ;YES - SKIP
6077 034524  104427                         PERR01                  ;V177664        ;NO- TAKE ERROR TRAP
6078 034526  077305                  4$:    SOB     R3,3$           ;V177666        ;LOOP FOR 1/2 BANK
6079 034530  000207                         RETURN
```

```
                                MTP033: SUBTST  <<MTP033        BRANCH GOBBLE TEST>>
                                ;*****************************************************************************
                                ;*SUBTEST        MTP033  BRANCH GOBBLE TEST
                                ;*****************************************************************************
                                        .DSABL  AMA
034532  000000                  0                               ;MOVE TERMINATOR
034534  000000                BGTEST: 0                         ;TEST WORD (TWO BYTES)
034536  000261                BRGOBB: SEC                       ;SET CARRY (TO BE ADDED TO 'BGTEST')
034540  105511                        ADCB    (R1)            ;INCREMENT LOW BYTE OF 'BGTEST'
034542  100402                        BMI     1$              ;BRANCH WHEN BIT7 IS SET
034544  105212                        INCB    (R2)            ;INCREMENT HIGH BYTE OF 'BGTEST'
034546  000773                        BR      BRGOBB          ;LOOP 128 TIMES

                                ;NOW CHECK FOR CORRECT CONDITION CODES
034550  102401                1$:     BVS     2$              ;BR IF V-BIT SET (SHOULD BE)
034552  104461                        PERR35                  ;NO - REPORT ERROR AND ABORT TEST
                                                              ;COND CODES NOT EQUAL TO 1010
034554  000242                2$:     CLV                     ;CLEAR V-BIT
034556  105212                        INCB    (R2)            ;INCREMENT HIGH BYTE OF 'BGTEST' ONCE MORE
034560  103402                        BCS     3$              ;BR IF C-BIT SET (SHOULD NOT BE)
034562  102001                        BVC     3$              ;BR IF V-BIT CLEAR (SHOULD NOT BE)
034564  100401                        BMI     4$              ;BR IF N-BIT SET (SHOULD BE)
034566  104461                3$:     PERR35                  ;NO - REPORT ERROR AND ABORT TEST
                                                              ;COND CODES NOT EQUAL TO 1010

                                ;UPDATE TEST POINTERS
034570  010701                4$:     MOV     PC,R1
034572  162701  000036        5$:     SUB     #5$-BGTEST,R1
034576  010102                        MOV     R1,R2
034600  005202                        INC     R2

                                ;UPDATE MOVE REGISTERS
034602  010503                        MOV     R5,R3
034604  005725                        TST     (R5)+           ;BUMP (SAFELY) BY 2
034606  010504                        MOV     R5,R4

                                ;DONE?
034610  020537  002472                CMP     R5,@#LINK1      ;DONE?
034614  001001                        BNE     6$              ;NO - SKIP
034616  000207                        RETURN                  ;YES - RETURN

                                ;MOVE CODE 1 LOCATION
034620  014344                6$:     MOV     -(R3),-(R4)
034622  001376                        BNE     6$
034624  005011                        CLR     (R1)            ;CLEAR TEST WORD 'BGTEST'
034626  000743                        BR      BRGOBB          ;RUN MOVED CODE AGAIN
        000076                GBLENGTH=.-MTP033
                                        .ENABL  AMA
```

```
 6128 034630                        MTP034: SUBTST  <<MTP034        SOFT ERROR - BACKROUND PATTERN TEST>>
                                    ;*********************************************************************************
                                    ;*SUBTEST        MTP034  SOFT ERROR - BACKROUND PATTERN TEST
                                    ;*********************************************************************************
 6129 034630  010220                1$:     MOV     R2,(R0)+        ;V177640
 6130 034632  077102                        SOB     R1,MTP034       ;V177642
 6131 034634  000240                        NOP                     ;V177644
 6132 034636  012401                2$:     MOV     (R4)+,R1        ;V177646
 6133 034640  020102                        CMP     R1,R2           ;V177650
 6134 034642  001402                        BEQ     3$              ;V177652
 6135 034644  104430                        PERR02                  ;V177654
 6136 034646  000240                        NOP                     ;V177656
 6137 034650  077306                3$:     SOB     R3,2$           ;V177660
 6138 034652  000207                        RETURN                  ;V177662
```

```
 6140 034654                            MTP035:SUBTST   <<MTP035        WORST CASE NOISE PARITY TEST>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       MTP035  WORST CASE NOISE PARITY TEST
                                        ;*****************************************************************************
 6141 034654  012737  000003  002074          MOV     #3,NOPAR        ;SET PARITY TRAPS TO RETURN TO 'PARTHERE'
 6142
 6143 034662                                    FOR R0 := #FIRST TO #LAST BY #4000
 6144 034666  012737  000005  002144              MOV   #BIT2!BIT0,CSR  ;SET WRITE WRONG PARITY & PAR. TRAPS INTO CSR
 6145 034674  104425                              LOADCSR
 6146 034676  012737  034732  002264              MOV   #1$,PARTHERE
 6147 034704  011010                              MOV   (R0),(R0)       ;WWP TEST LOCATION
 6148 034706  005710                              TST   (R0)
 6149 034710  010037  002032                      MOV   R0,ADDRESS
 6150 034714  104050                              ERROR +50
 6151 034716  004737  053464                      CALL  PERBNK
 6152 034722  032763  002000  002626              BIT   #BIT10,CONFIG+2(R3)
 6153 034730  001002                              BNE   2$
 6154 034732  104426                        1$:   READCSR
 6155 034734  104512                              ERRGEN
 6156
 6157 034736  104503                        2$:   CLR1CSR
 6158 034740  011010                              MOV   (R0),(R0)       ;CLEAR WRONG PARITY IN MEMORY
 6159 034742  012737  000001  002144              MOV   #BIT0,CSR
 6160 034750  104425                              LOADCSR
 6161 034752  012737  034764  002264              MOV   #3$,PARTHERE
 6162 034760  005710                              TST   (R0)
 6163 034762  000405                              BR    4$
 6164 034764  010037  002032                3$:   MOV   R0,ADDRESS
 6165 034770  104050                              ERROR +50
 6166 034772  004737  053464                      CALL  PERBNK
 6167 034776                                4$:   END; OF FOR
 6168
 6169 035010  005037  002074                    CLR     NOPAR           ;RESET PARITY TRAP ACTION
 6170 035014  000207                            RETURN
```

```
                                        .SBTTL  MISC SUBROUTINES

                                REGCOPY:SUBTST  <<SUBR  COPY R0 TO R4,R1 TO R3, & R2 TO R5>>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    COPY R0 TO R4,R1 TO R3, & R2 TO R5
                                ;*****************************************************************************
035016  010004                          MOV     R0,R4
035020  010103                          MOV     R1,R3
035022  010205                          MOV     R2,R5
035024  000207                          RETURN

                                FLIPWARN:SUBTST <<FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS>>
                                ;*****************************************************************************
                                ;*SUBTEST        FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS
                                ;*****************************************************************************
035026                                  PUSH    R0
035030  005237  002556                   INC     FLIPLOC
035034  042737  177774  002556          BIC     #^C3,FLIPLOC
035042  022737  000001  002556          CMP     #1,FLIPLOC
035050  001414                          BEQ     1$
035052  022737  000002  002556          CMP     #2,FLIPLOC
035060  001413                          BEQ     2$
035062  022737  000003  002556          CMP     #3,FLIPLOC
035070  001414                          BEQ     3$
035072  005000                          CLR     R0
035074  013704  002554                  MOV     ONES,R4
035100  000414                          BR      4$
035102                          1$:     CLEAR   R0,R4
035106  000411                          BR      4$
035110  012700  000401          2$:     MOV     #401,R0
035114  013704  002554                  MOV     ONES,R4
035120  000404                          BR      4$
035122  012700  000401          3$:     MOV     #401,R0
035126  012704  000401                  MOV     #401,R4
035132  010037  025636          4$:     MOV     R0,WARN2
035136  010037  025652                  MOV     R0,WARN3
035142  010037  025676                  MOV     R0,WARN4
035146  010037  025712                  MOV     R0,WARN5
035152                                  POP     R0
035154  000207                          RETURN
```

```
6207 035156                                BACKGND:SUBTST  <<SUBR  WRITE BACKGROUND>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       SUBR    WRITE BACKGROUND
                                           ;*****************************************************************************
6208                                               ;WRITES DATA FROM R2
6209 035156  104415                               SAVREG
6210 035160  012700  060000                       MOV     #FIRST,R0
6211 035164  012701  040000                       MOV     #SIZE,R1
6212 035170  005737  002424                       TST     NO22BIT
6213 035174  001415                               BEQ     WARN6B
6214 035176  012737  000207  025520  WARN6A:      MOV     #207,MTP000+4          ;WARNING PUTTING 'RETURN'' AFTER WRITE
6215 035204  012737  025514  002254               MOV     #MTP000,SUPDOADD
6216 035212  004737  025322                       CALL    SUPD03
6217 035216  012737  000240  025520               MOV     #240,MTP000+4          ;RESTORE 'NOP' AFTER WRITE
6218 035224  104416                               RESREG
6219 035226  000207                               RETURN
6220 035230                                WARN6B: BMOV    MTP000
6221 035236  012737  000207  177644  WARN6:      MOV     #207,UIPAR2            ;WARNING PUTTING 'RETURN'' INSTRUCTION AFTER WRITE
6222 035244  004737  025144                       CALL    SUPD01
6223 035250  104416                               RESREG
6224 035252  000207                               RETURN
```

```
 6227 035254                            PCONFIG:SUBTST  <<SUBR  PRINT CONFIGURATION MAP>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       SUBR    PRINT CONFIGURATION MAP
                                        ;*****************************************************************************
 6228 035254                                    PUSH    TKVEC,TKVEC+2,R0
 6229 035266  010637  035570                    MOV     SP,PCONFS               ;SAVE LAST GOOD SP
 6230 035272  012737  035536  000060            MOV     #PCONF2,TKVEC
 6231 035300  012737  000340  000062            MOV     #340,TKVEC+2
 6232 035306  017700  145272                    MOV     @$TKB,R0                ;KILL ANY OLD INTERRUPT
 6233 035312  042737  000200  177776            BIC     #BIT7,PSW               ;LOWER CPU PRIORITY TO 140
 6234 035320  052777  000100  145254            BIS     #BIT6,@$TKS             ;ENABLE KEYBOARD INTERRUPTS
 6235
 6236 035326                                    TYPE    MSG001
 6237 035332                                    TYPE    MSG002
 6238 035336                                    IF LASTBANK EQ #7
 6239 035346  012700  000010                      MOV   #8.,R0
 6240 035352  010004                              MOV   R0,R4
 6241 035354                                      CLEAR R1,R3
 6242 035360                                      TYPE  MSG115
 6243 035364  004737  035572                      CALL  TCONFIG
 6244 035370  000462                              BR    PCONF2
 6245 035372                                    END; OF IF LASTBANK
 6246 035372                                    TYPE    MSG003
 6247                                           ;IF FAT PAPER ON TERMINAL GOTO 1$
 6248 035376                                    IF #SW4 SET.IN @SWR THEN JUMPTO PCONF1
 6249 035412  012700  000074                    MOV     #60.,R0
 6250 035416  010004                            MOV     R0,R4
 6251 035420                                    CLEAR   R1,R3
 6252 035424                                    TYPE    MSG004
 6253 035430  004737  035572                    CALL    TCONFIG                 ;GO TYPE CONFIGURATION (1ST HALF)
 6254 035434                                    TYPE    $CRLF
 6255 035440                                    TYPE    MSG017                  ;PRINT SPACE(S)
 6256 035444                                    TYPE    MSG011
 6257 035450                                    TYPE    $CRLF
 6258 035454                                    TYPE    MSG017                  ;PRINT SPACE(S)
 6259 035460                                    TYPE    MSG012
 6260 035464  012701  000360                    MOV     #60.*2*2,R1
 6261 035470  010103                            MOV     R1,R3
 6262 035472  004737  035572                    CALL    TCONFIG
 6263 035476  000417                            BR      PCONF2
 6264
 6265 035500  012700  000170            PCONF1: MOV     #120.,R0
 6266 035504  010004                            MOV     R0,R4
 6267 035506                                    CLEAR   R1,R3
 6268 035512                                    TYPE    MSG014                  ;SPACE
 6269 035516                                    TYPE    MSG011
 6270 035522                                    TYPE    MSG004
 6271 035526                                    TYPE    MSG012
 6272 035532  004737  035572                    CALL    TCONFIG
 6273
 6274 035536  013706  035570            PCONF2: MOV     PCONFS,SP               ;RESTORE STACK
 6275 035542  042777  000100  145032            BIC     #BIT6,@$TKS
 6276 035550  117700  145030                    MOVB    @$TKB,R0                ;READ CHAR TO KILL FLAG
 6277 035554                                    POP     R0,TKVEC+2,TKVEC
 6278 035566  000207                            RETURN
 6279
 6280 035570  000000            PCONFS: 0                                       ;STACK SAVED HERE!
```

```
                                        SUBTST  <<SUBR  TYPE CONFIGURATION>>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    TYPE CONFIGURATION
                                ;*****************************************************************************
                                ;*****************************************************************************
6284                            ;
6285                            ;CALL:  MOV     #N,R0           ;N=NUMBER OF CHARACTERS
6286                            ;       MOV     R0,R4           ;BACKUP
6287                            ;       MOV     #K.R1           ;INDEX CONSTANT
6288                            ;       MOV     R1,R3           ;BACKUP
6289                            ;       CALL    TCONFIG         ;ACTUAL CALL
6290                            ;       RETURN                  ;ONLY RETURN
6291                            ;*****************************************************************************
6292
6293                                    ;***********
6294                                    ;** ERROR **
6295                                    ;***********
6296 035572                    TCONFIG:TYPE    MSG005
6297 035576  032761  000001  002624  1$:     BIT     #BIT0,CONFIG(R1)        ;ERROR ON THIS BANK?
6298 035604  001403                          BEQ     2$                      ;NO - SKIP
6299 035606                                  TYPE    MSG013                  ;PRINT 'X'
6300 035612  000402                          BR      3$
6301 035614                            2$:   TYPE    MSG014                  ;PRINT SPACE
6302 035620  062701  000004            3$:   ADD     #4,R1                   ;BUMP POINTER
6303 035624  077014                          SOB     R0,1$                   ;LOOP TILL DONE
6304 035626  010400                          MOV     R4,R0
6305 035630  010301                          MOV     R3,R1
6306
6307                                    ;***********
6308                                    ;** CPU'S **
6309                                    ;***********
6310 035632                                  TYPE    MSG008
6311 035636  016105  002624            4$:   MOV     CONFIG(R1),R5
6312 035642  006205                          ASR     R5                      ;GET CPU BITS
6313 035644  042705  177760                  BIC     #^C17,R5                ;CLEAR NON INTERESTING BITS
6314 035650  005705                          TST     R5                      ;IS THERE ANYTHING THERE?
6315 035652  001003                          BNE     8$                      ;YES - BRANCH.
6316 035654  112705  000040                  MOVB    #' ,R5                  ;NO - MOVE A BLANK INTO R5
6317 035660  000402                          BR      9$                      ;BRANCH OVER NEXT INSTRUCTION
6323 035662  062705  000060            8$:   ADD     #60,R5                  ;MAKE ASCII
6324 035666  110537  073603            9$:   MOVB    R5,MSG015               ;PLUG INTO MEMORY
6325 035672                                  TYPE    MSG015
6326 035676  062701  000004                  ADD     #4,R1                   ;BUMP POINTER
6327 035702  077023                          SOB     R0,4$                   ;LOOP TILL DONE
6328 035704  010400                          MOV     R4,R0
6329 035706  010301                          MOV     R3,R1
```

```
                                                ;***************
                                                ;** INTERLEAVE **
                                                ;***************
035710                                          TYPE    MSG007
                                                ;THIS IS AN ENTRY POINT FROM ERROR REPORTS
035714  032761  010000  002626  TCFIG1: BIT     #BIT12,CONFIG+2(R1)
035722  001014                        BNE     1$
035724  032761  000002  002624        BIT     #BIT1,CONFIG(R1)        ;IS THERE ANY MEMORY HERE?
035732  001004                        BNE     18$                     ;BRANCH IF MEMORY PRESENT.
035734  112737  000040  073603        MOVB    #' ,MSG015              ;MOVE A BLANK IN TO BE PRINTED
035742  000424                        BR      16$                     ;BRANCH TO TYPE ROUTINE
035744  112737  000055  073603  18$:   MOVB    #'-,MSG015
035752  000420                        BR      16$
035754  016105  002624          1$:   MOV     CONFIG(R1),R5
035760  042705  007777                BIC     #^C170000,R5            ;GET CSR INTERLEAVE
035764  000305                        SWAB    R5
035766  072527  177774                ASH     #-4,R5
035772  022705  000012                CMP     #10.,R5
035776  100002                        BPL     2$
036000  062705  000007                ADD     #7,R5
036004  062705  000060          2$:   ADD     #60,R5                  ;MAKE ASCII
036010  110537  073603                MOVB    R5,MSG015               ;PLUG INTO MEMORY
036014                          16$:   TYPE    MSG015
036020                                IF NOTAB NE #0 THEN $RETURN
036030  062701  000004                ADD     #4,R1                   ;BUMP POINTER
036034  077051                        SOB     R0,TCFIG1               ;LOOP TILL DONE
036036  010400                        MOV     R4,R0
036040  010301                        MOV     R3,R1

                                                ;*****************
                                                ;** MEMORY TYPE **
                                                ;*****************
                                                .ENABL  LSB
036042                                          TYPE    MSG009
036046  033761  002104  002624  TCFIG2: BIT     CPUBIT,CONFIG(R1)
036054  001447                        BEQ     17$
036056  016105  002626                MOV     CONFIG+2(R1),R5
036062  000305                        SWAB    R5                      ;GET MEMORY TYPE
036064  042705  177770                BIC     #^C7,R5                 ;CLEAR NON INTERESTING BITS
036070  005705                        TST     R5
036072  001440                        BEQ     17$
036074  032705  000004                BIT     #BIT2,R5
036100  001004                        BNE     4$
036102  112737  000102  073603        MOVB    #'B,MSG015
036110  000434                        BR      8$
036112  032705  000002          4$:   BIT     #BIT1,R5
036116  001013                        BNE     6$
036120  032705  000001                BIT     #BIT0,R5
036124  001004                        BNE     5$
036126  112737  000115  073603        MOVB    #'M,MSG015
036134  000422                        BR      8$
036136  112737  000113  073603  5$:   MOVB    #'K,MSG015
036144  000416                        BR      8$
036146  032705  000001          6$:   BIT     #BIT0,R5
036152  001004                        BNE     7$
036154  112737  000114  073603        MOVB    #'L,MSG015
036162  000407                        BR      8$
```

```
 6389 036164  112737  000120  073603  7$:     MOVB    #'P,MSG015
 6390 036172  000403                          BR      8$
 6391 036174  112737  000040  073603  17$:    MOVB    #' ,MSG015
 6392 036202                          8$:     TYPE    MSG015
 6393 036206                                  IF NOTAB NE #0 THEN $RETURN
 6394 036216  062701  000004                  ADD     #4,R1                   ;BUMP POINTER
 6395 036222  077067                          SOB     R0,TCFIG2               ;LOOP TILL DONE
 6396 036224  010400                          MOV     R4,R0
 6397 036226  010301                          MOV     R3,R1
 6398                                         .DSABL  LSB
 6399
 6400                                         ;*********
 6401                                         ;** CSR **
 6402                                         ;*********
 6403 036230                                  TYPE    MSG016
 6404 036234  112737  000040  073603  TCFIG3: MOVB    #' ,MSG015
 6405 036242  016105  002624                  MOV     CONFIG(R1),R5
 6406 036246  032705  000002                  BIT     #BIT1,R5
 6407 036252  001414                          BEQ     16$
 6408 036254  042705  170377                  BIC     #^C7400,R5
 6409 036260  000305                          SWAB    R5
 6410 036262  022705  000012                  CMP     #10.,R5
 6411 036266  100002                          BPL     10$
 6412 036270  062705  000007                  ADD     #7,R5
 6413 036274  062705  000060          10$:    ADD     #60,R5                  ;MAKE ASCII
 6414 036300  110537  073603                  MOVB    R5,MSG015               ;PLUG INTO MEMORY
 6415 036304                          16$:    TYPE    MSG015
 6416 036310                                  IF NOTAB NE #0 THEN $RETURN
 6417 036320  062701  000004                  ADD     #4,R1                   ;BUMP POINTER
 6418 036324  077035                          SOB     R0,TCFIG3               ;LOOP TILL DONE
 6419 036326  010400                          MOV     R4,R0
 6420 036330  010301                          MOV     R3,R1
 6421
 6422                                         ;****************
 6423                                         ;** PROTECTED **
 6424                                         ;****************
 6425 036332                                  TYPE    MSG010
 6426 036336  105761  002624          11$:    TSTB    CONFIG(R1)              ;BANK PROTECTED?
 6427 036342  100004                          BPL     12$                     ;NO - SKIP
 6428 036344  112737  000120  073603          MOVB    #'P,MSG015
 6429 036352  000407                          BR      13$
 6430 036354  032761  000100  002624  12$:    BIT     #BIT6,CONFIG(R1)        ;PROTECTED REGION OF ECC?
 6431 036362  001406                          BEQ     14$                     ;NO - SKIP
 6432 036364  112737  000111  073603          MOVB    #'I,MSG015
 6433 036372                          13$:    TYPE    MSG015
 6434 036376  000402                          BR      15$
 6435 036400                          14$:    TYPE    MSG014                  ;PRINT SPACE
 6436 036404  062701  000004          15$:    ADD     #4,R1                   ;BUMP POINTER
 6437 036410  077026                          SOB     R0,11$                  ;LOOP TILL DONE
 6438 036412  010400                          MOV     R4,R0
 6439 036414  010301                          MOV     R3,R1
 6440 036416  000207                          RETURN
```

```
                                        .SBTTL  TRAP    PARITY ERROR HANDLER
                                ;*******************************************************************************
                                ;VECTOR TO HERE FROM TRAPS TO 114
                                ;IGNORE ERRORS BUT COUNT IF NOPAR FLAG = 1.
                                ;*******************************************************************************
                                ;
                                ;       CODE    ACTION
                                ;
                                ;       --0-    PRINT UNEXPECTED PARITY TRAP
                                ;         1     COUNT ERROR
                                ;         2     SET ''ABORT'' / SETUP 'BADPC'' / RETURN VIA PCBUMP
                                ;         3     RETURN VIA 'PARTHERE''
                                ;
036420  022737  000001  002074  PARITY: CMP     #1,NOPAR                ;COUNTING PARITY ERRORS?
036426  001003                          BNE     1$                      ;NO - SKIP
036430  005237  002070                  INC     PARCNT                  ;PARITY ERROR COUNTER + 1
036434  000002                          RTI
036436  022737  000002  002074  1$:     CMP     #2,NOPAR                ;ACTION CODE = 2 ?
036444  001013                          BNE     2$                      ;NO - SKIP
036446                                  SET     ABORTFLAG               ;YES
036454  004737  036626                  CALL    BADSTACK                ;FIND BAD SP,PC,PSW OFF STACK
036460  063716  002276                  ADD     PCBUMP,(SP)             ;UPDATE RETURN PC
036464  042766  000004  000002          BIC     #BIT2,2(SP)             ;SHOW FAILURE BY .NE.
036472  000002                          RTI
036474  022737  000003  002074  2$:     CMP     #3,NOPAR                ;ACTION CODE = 3 ?
036502  001003                          BNE     3$                      ;NO - SKIP
036504  013716  002264                  MOV     PARTHERE,(SP)
036510  000002                          RTI
036512  004737  036626          3$:     CALL    BADSTACK                ;FIND BAD SP,PC,PSW OFF STACK
036516                                  FATAL   32
```

```
                                        .SBTTL  TRAP    NON-EXISTANT MEMORY (HOLES) HANDLER
                                ;*******************************************************************************
                                ;VECTOR TO HERE (SOMETIMES) FROM TRAPS TO 4
                                ;       CODE IN NONEM DETERMINES ACTION AS FOLLOWS:
                                ; 1)    IGNORE ERRORS BUT COUNT IF NONEM (NO NON-EXISTANT MEMORY) FLAG = 1.
                                ; 2)    TO EXIT PATTERN 0 DURING SIZING IF NON-EXIST MEM ERROR
                                ;*******************************************************************************

036524  022737  000001  002076  NONEXIST:CMP     #1,NONEM                ;COUNTING NON-EXISTANT MEMORY ERRORS?
036532  001011                          BNE     2$                      ;NO - SKIP
036534  005237  002066                  INC     NEMCNT                  ;BUMP NON-EXISTANT MEMORY COUNTER
036540  022737  000001  002066          CMP     #1,NEMCNT               ;FIRST ERROR?
036546  001002                          BNE     1$                      ;NO - SKIP
036550  010037  002032                  MOV     R0,ADDRESS              ;ASSUME R0 CONTAINS THE ADDRESS ACCESSED
036554  000002                  1$:     RTI
036556  005237  002066          2$:     INC     NEMCNT                  ;BUMP NON-EXISTANT MEMORY COUNTER
036562  012701  000001                  MOV     #1,R1                   ;DUMMY UP R1 FOR A FORCED SOB EXIT
036566  000002                          RTI

                                ;*******************************************************************************
                                        .SBTTL  TRAP    TIMEOUT (TRAP TO 4) HANDLER
036570  004737  036626          TIMEOUT:CALL    BADSTACK                ;FIND BAD SP,PC,PSW OFF STACK
036574                                  FATAL   6
                                ;*******************************************************************************
                                        .SBTTL  TRAP    MEMORY MANAGEMENT (TRAP TO 250) HANDLER
036602  004737  036626          MMTRAP: CALL    BADSTACK                ;FIND BAD SP,PC,PSW OFF STACK
036606                                  FATAL   7
                                        .SBTTL  TRAP    RESERVED INSTRUCTION HANDLER
036614  004737  036626          PDP1105:CALL    BADSTACK                ;FIND BAD SP,PC,PSW OFF STACK
036620                                  FATAL   5


                                BADSTACK:SUBTST <<FIND BAD SP, PC, & PSW FROM STACK>>
                                ;*******************************************************************************
                                ;*SUBTEST        FIND BAD SP, PC, & PSW FROM STACK
                                ;*******************************************************************************
036626  010637  002024                  MOV     SP,BADSP
036632  062737  000002  002024          ADD     #2,BADSP
036640  016637  000002  002020          MOV     2(SP),BADPC
036646  016637  000004  002030          MOV     4(SP),BADPSW
036654  000207                          RETURN
```

```
                                        .SBTTL  TRAP    KERNEL TRAP HANDLER
                                ;*******************************************************************************
                                ;KERNEL IS A TRAP THAT COMES HERE
                                ;*******************************************************************************
036656  042766  140000  000002  $KERNEL:      BIC     #140000,2(SP)
036664  000002                  RTI
                                ;*******************************************************************************
                                        .SBTTL  TRAP    ENERGIZE TRAP HANDLER
036666  052737  000001  177572  $ENERGIZE:BIS   #BIT0,MMR0
036674  000002                  RTI
                                ;*******************************************************************************
                                        .SBTTL  TRAP    DEENERGIZE TRAP HANDLER
036676  042737  000001  177572  $DEENERGIZE:BIC #BIT0,MMR0
036704  000002                  RTI
                                ;*******************************************************************************
                                        .SBTTL  TRAP    CACHON TRAP HANDLER
036706  005737  002514          $CACHN: TST     CACHKN          ;IS THERE A CACHE
036712  001406                  BEQ     1$              ;NO - RETURN
036714  013737  002514  177746  MOV     CACHKN,CONTRL   ;SETUP CACHE AS PER CONSTANT (USUALLY 1 = FULLY ON)
036722  052737  000001  177746  BIS     #BIT0,CONTRL    ;DISABLE TRAPS (BUT NOT ABORTS)
036730  000002                  1$:     RTI
                                ;*******************************************************************************
                                        .SBTTL  TRAP    CACHOFF TRAP HANDLER
036732  005737  002514          $CACHF: TST     CACHKN          ;IS THERE A CACHE?
036736  001403                  BEQ     1$              ;NO - RETURN
                                ;DISABLE TRAPS (NOT ABORTS), FORCE MISSES, FLUSH, BYPASS
036740  053737  002520  177746  BIS     CACHKF,CONTRL
036746  000002                  1$:     RTI
```

```
                                                .SBTTL  TRAP    LOAD CSR TRAP HANDLER
                                                ;LOAD CORRECT CSR WITH DATA IN CSR
                                                ;PROGRAM CSR'S ASSERT INHIBIT MODE POINTER WHEN LOADED
036750                                  $LOADC: PUSH    R0,R1                   ;SAVE REGISTERS
036754  013700  002146                          MOV     CSRNO,R0                ;CREATE CSR ADDRESS
036760                                          IF INHECC IS TRUE THEN GOTO 3$  ;DON'T WANT INH. MODE POINTER ON
036766  005737  002502                          TST     PGMCSR                  ;PROGRAM IN INTERLEAVED SPACE?
036772  100007                                  BPL     1$                      ;BRANCH IF NOT
036774  113701  002503                          MOVB    PGMCSR+1,R1             ;CHECK SECOND CSR
037000  042701  177740                          BIC     #^C37,R1                ;CLEAR UNNECESSARY BITS
037004  020137  002146                          CMP     R1,CSRNO                ;IS THIS THE CURRENT CSR?
037010  001404                                  BEQ     2$                      ;BRANCH IF IT IS
037012  123737  002502  002146  1$:             CMPB    PGMCSR,CSRNO            ;IS THIS THE CURRENT CSR?
037020  001003                                  BNE     3$                      ;BRANCH IF NOT
037022  052737  020000  002144  2$:             BIS     #BIT13,CSR              ;SET THE INHIBIT MODE POINTER TO  1ST 16K
037030  013760  002144  172100  3$:             MOV     CSR,CSRADD(R0)          ;LOAD THE CSR
037036                                          POP     R1,R0                   ;RESTORE REGISTERS
037042  000002                                  RTI
                                                .SBTTL  TRAP    READ CSR TRAP HANDLER
                                                ;READ THE CORRECT CSR INTO LOCATIONS CSR
037044                                  $READC: PUSH    R0
037046  013700  002146                          MOV     CSRNO,R0
037052  016037  172100  002144                  MOV     CSRADD(R0),CSR          ;READ IT
037060                                          POP     R0
037062  000002                                  RTI
```

```
6578                                            .SBTTL  TRAP    TEST (R1) & READ CSR CAREFULLY
6579 037064                             $TSTRD: PUSH    R0,R2,R3
6580 037072  012700  172100                    MOV     #CSRADD,R0              ;CREATE CSR ADDRESS
6581 037076  063700  002146                    ADD     CSRNO,R0
6582 037102  005002                            CLR     R2
6583 037104  005737  002502                    TST     PGMCSR
6584 037110  100007                            BPL     1$
6585 037112  113703  002503                    MOVB    PGMCSR+1,R3
6586 037116  042703  000200                    BIC     #BIT7,R3
6587 037122  020337  002146                    CMP     R3,CSRNO
6588 037126  001404                            BEQ     2$
6589 037130  123737  002502  002146  1$:       CMPB    PGMCSR,CSRNO
6590 037136  001002                            BNE     3$
6591 037140  012702  020000          2$:       MOV     #BIT13,R2
6592 037144  005737  002424          3$:       TST     NO22BIT         ;IS THIS AN 11/44?
6593 037150  001403                            BEQ     4$              ;BRANCH IF IT IS
6594 037152  004737  037240                    CALL    TSTRD1
6595 037156  000405                            BR      5$
6596 037160                             4$:     BMOV    TSTRD1
6597 037166  004737  177640                    CALL    FASTCITY                ;CALL TO THE USER INSTRUCTION PAR'S
6598                                            ;IF SINGLE BIT ERROR ONLY - SET CARRY BIT
6599 037172                             5$:     POP     R3,R2,R0
6600 037200                                     IF #BIT4 SET.IN CSR AND #BIT15 OFF.IN CSR
6601 037220  052766  000001  000002               BIS   #BIT0,2(SP)
6602 037226                                     ELSE
6603 037230  042766  000001  000002               BIC   #BIT0,2(SP)
6604 037236                                     END ;OF IF #BIT4
6605 037236  000002                            RTI
6606
6607 037240  010210                     TSTRD1: MOV     R2,(R0)                 ;V177640
6608 037242                                     TESTAREA                        ;V177642 ;ENTER SUPERVISOR MODE
6609 037250  105711                             TSTB    (R1)                    ;V177646
6610 037252  042737  140000  177776             BIC     #BIT15!BIT14,PSW        ;V177650
6611 037260  011037  002144                     MOV     (R0),CSR                ;V177656
6612 037264  000207                             RETURN                          ;V177662
```

```
6615                                                    .SBTTL  TRAP    ECC DISABLE ALL CSR'S TRAP HANDLER
6616 037266  012737  000002  002144  $ECCDIS:MOV       #BIT1,CSR
6617 037274  004737  040012                  CALL      CSROUT
6618 037300  000002                          RTI
6619                                                    .SBTTL  TRAP    ECC DISABLE OF 1 SELECTED CSR TRAP HANDLER
6620 037302  012737  000002  002144  $ECC1DIS:MOV      #BIT1,CSR
6621 037310  104425                        LOADCSR
6622 037312  000002                          RTI
6623                                                    .SBTTL  TRAP    INITIALIZE ALL CSR'S TRAP HANDLER
6624 037314  012737  000001  002144  $ECCINIT:MOV      #BIT0,CSR
6625 037322  004737  040012                  CALL      CSROUT
6626 037326  000002                          RTI
6627                                                    .SBTTL  TRAP    INITIALIZE 1 SELECTED CSR TRAP HANDLER
6628 037330  012737  000001  002144  $ECC1INIT:MOV     #BIT0,CSR
6629 037336  104425                        LOADCSR
6630 037340  000002                          RTI
6631                                                    .SBTTL  TRAP    ENABLE SBE PARITY TRAPS ON ALL CSR'S
6632 037342  012737  000003  002144  $ENASBE:MOV       #BIT0!BIT1,CSR
6633 037350  004737  040012                  CALL      CSROUT
6634 037354  000002                          RTI
6635                                                    .SBTTL  TRAP    ENABLE SBE PARITY TRAPS ON 1 SELECTED CSR
6636 037356  012737  000003  002144  $ENA1SBE:MOV      #BIT0!BIT1,CSR
6637 037364  104425                        LOADCSR
6638 037366  000002                          RTI
6639                                                    .SBTTL  TRAP    WRITE CHECKBITS THRU ALL CSR'S TRAP HANDLER
6640 037370  013737  002274  002144  $CBCSR: MOV       CHECK,CSR               ;BITS 11-5
6641 037376  052737  000006  002144          BIS       #BIT1!BIT2,CSR          ;CHECK MODE
6642 037404  004737  040012                  CALL CSROUT
6643 037410  000002                          RTI
6644                                                    .SBTTL  TRAP    WRITE CHECKBITS THRU 1 SELECTED CSR TRAP HANDLER
6645 037412  013737  002274  002144  $CB1CSR:MOV       CHECK,CSR               ;BITS 11-5
6646 037420  052737  000006  002144          BIS       #BIT1!BIT2,CSR          ;CHECK MODE
6647 037426  104425                        LOADCSR
6648 037430  000002                          RTI
```

```
                                        .SBTTL  TRAP    WAS THERE A SBE ON ANY CSR TRAP HANDLER
037432                          $WASSBE:PUSH    R1,R4
037436  013701  002216                  MOV     TOTCSRS,R1      ;GET CSR'S BYTE
037442  005004                          CLR     R4
037444                                  BEGIN LWSBE
037444                                    FOR CSRNO := #0 TO #36 BY #2
037450  006301                              ASL         R1
037452                                      ON.ERROR
037454  104426                                READCSR
037456                                        IF #BIT4 SET.IN CSR
037466                                          SET     R4
037472                                          LEAVE LWSBE
037474                                        END ;OF IF #BIT4
037474                                      END ;OF ON.ERROR
037474                                      IF R1 EQ #0 THEN LEAVE LWSBE
037500                                    END ;OF FOR CSRNO
037516                                  END LWSBE
037516  006004                          ROR     R4              ;SET C BIT FOR ERROR
037520                                  POP     R4,R1
037524                                  ON.ERROR
037526  052766  000001  000002            BIS   #BIT0,2(SP)
037534                                  ELSE
037536  042766  000001  000002            BIC   #BIT0,2(SP)
037544                                  END ;OF ON.ERROR
037544  000002                          RTI
                                        .SBTTL  TRAP    WAS THERE A SBE IN 1 SELECTED CSR TRAP HANDLER
                                        ;ON RETURN IF CARRY IS SET THERE WAS A SBE
037546  104426                  $WAS1SBE:READCSR
037550  042766  000001  000002          BIC     #BIT0,2(SP)     ;CLR C BIT ON STACK
037556  032737  000020  002144          BIT     #BIT4,CSR
037564  001403                          BEQ     1$
037566  052766  000001  000002          BIS     #BIT0,2(SP)     ;SET C BIT ON STACK
037574  000002                  1$:     RTI
```

```
                                        .SBTTL  TRAP    WAS THERE A DBE ON ANY CSR TRAP HANDLER
037576                          $WASDBE:PUSH    R1,R4
037602  013701  002216                  MOV     TOTCSRS,R1      ;GET CSR'S BYTE
037606  005004                          CLR     R4
037610                                  BEGIN LWDBE
037610                                    FOR CSRNO := #0 TO #36 BY #2
037614  006301                              ASL         R1
037616                                      ON.ERROR
037620  104426                                READCSR
037622                                        IF #BIT15 SET.IN CSR
037632                                          SET     R4
037636                                          LEAVE LWDBE
037640                                        END ;OF IF #BIT4
037640                                      END ;OF ON.ERROR
037640                                    IF R1 EQ #0 THEN LEAVE LWDBE
037644                                    END ;OF FOR CSRNO
037662                                  END LWDBE
037662  006004                          ROR     R4              ;SET C BIT FOR ERROR
037664                                  POP     R4,R1
037670                                  ON.ERROR
037672  052766  000001  000002            BIS   #BIT0,2(SP)
037700                                  ELSE
037702  042766  000001  000002            BIC   #BIT0,2(SP)
037710                                  END ;OF ON.ERROR
037710  000002                          RTI
                                        .SBTTL  TRAP    WAS THERE A DBE ON 1 SELECTED CSR TRAP HANDLER
                                        ;ON RETURN IF CARRY IS SET THERE WAS A DBE
037712  104426                  $WAS1DBE:READCSR
037714  005737  002144                  TST     CSR             ;DBE?
037720  100004                          BPL     3$              ;NO - SKIP
037722  052766  000001  000002          BIS     #BIT0,2(SP)     ;SET C BIT ON STACK
037730  000002                          RTI
037732  042766  000001  000002  3$:     BIC     #BIT0,2(SP)     ;CLR C BIT ON STACK
037740  000002                          RTI
```

```
 6722                                           .SBTTL  TRAP    CLEAR ALL ECC CSR'S TRAP HANDLER
 6723 037742                            $CLRCSR:CLEAR  CSR
 6724 037746  004737  040012                    CALL    CSROUT
 6725 037752  000002                            RTI
 6726                                           .SBTTL  TRAP    CLEAR 1 SELECTED CSR TRAP HANDLER
 6727 037754                            $CLR1CSR:CLEAR  CSR
 6728 037760  104425                            LOADCSR
 6729 037762  000002                            RTI
 6730                                           .SBTTL  TRAP    ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN ALL CSR'S TRAP HANDLER
 6731                                           ;CHECKBITS ALREADY IN LOC "CSR"
 6732 037764  052737  000006  002144  $CHKDIS:BIS     #BIT1!BIT2,CSR          ;ECC DISABLE & DIAG CHECK MODE
 6733 037772  004737  040012                    CALL    CSROUT
 6734 037776  000002                            RTI
 6735                                           .SBTTL  TRAP    ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN 1 SELECTED CSR
 6736                                           ;CHECKBITS ALREADY IN LOC "CSR"
 6737 040000  052737  000006  002144  $CHK1DIS:BIS    #BIT1!BIT2,CSR          ;ECC DISABLE & DIAG CHECK MODE
 6738 040006  104425                            LOADCSR
 6739 040010  000002                            RTI
```

```
                                CSROUT: SUBTST  <<SUBR  WRITE IN ALL CSR'S>>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    WRITE IN ALL CSR'S
                                ;*****************************************************************************
040012                                  PUSH    R1
040014  013701  002216                  MOV     TOTCSRS,R1      ;GET CSR'S BYTE
040020                                  BEGIN LCSROUT
040020                                    FOR CSRNO := #0 TO #36 BY #2
040024  006301                              ASL         R1
040026                                      ON.ERROR
040030  104425                                LOADCSR
040032                                      END ;OF ON.ERROR
040032                                    IF R1 EQ #0 THEN LEAVE LCSROUT
040036                                    END ;OF FOR CSRNO
040054                                  END LCSROUT
040054                                  POP     R1
040056  000207                          RETURN

040060                          $INVALID:       SUBTST  <<TRAP  INVALIDATE BACKGROUND PATTERN>>
                                ;*****************************************************************************
                                ;*SUBTEST        TRAP    INVALIDATE BACKGROUND PATTERN
                                ;*****************************************************************************
040060                                  PUSH    R0,R1
040064  013701  002100                  MOV     BANK,R1
040070  006301                          ASL     R1
040072  006301                          ASL     R1
040074  042761  020000  002626          BIC     #BIT13,CONFIG+2(R1)
040102                                  POP     R1,R0
040106  000002                          RTI
```

```
6766 040110                          $ERRGEN:    SUBTST<<TRAP    GENERATE AND TEST ERROR ADDRESS>>
                                     ;*****************************************************************************
                                     ;*SUBTEST    TRAP    GENERATE AND TEST ERROR ADDRESS
                                     ;*****************************************************************************
6767 040110                                      PUSH    R0,R1,R2,R3
6768 040120  013703  002102                      MOV     BANKINDEX,R3
6769 040124  005737  002426                      TST     NOSUPER
6770 040130  001003                              BNE     6$
6771 040132  013700  172246                      MOV     SIPAR3,R0               ;GENERATE WHAT ERROR ADDR SHOULD BE
6772 040136  000402                              BR      7$
6773 040140  013700  177646          6$:         MOV     UIPAR3,R0
6774 040144  072027  177773          7$:         ASH     #-5,R0
6775 040150  005737  002130                      TST     EUFLAG
6776 040154  001002                              BNE     1$
6777 040156  042700  177600                      BIC     #^C177,R0
6778 040162  000301                1$:         SWAB    R1                      ;GET CURRENT ADDRESS BITS 11 AND 12
6779 040164  006201                              ASR     R1
6780 040166  006201                              ASR     R1
6781 040170  006201                              ASR     R1
6782 040172  042701  177775                      BIC     #^C2,R1
6783 040176  060100                              ADD     R1,R0                   ;ADD THEM TO THE ADJUSTED PAR VALUE
6784                                             ;GET ERROR ADDRESS FROM CSR UNDER TEST
6785 040200  013701  002144                      MOV     CSR,R1
6786 040204  072127  177773                      ASH     #-5,R1
6787 040210  042701  177600                      BIC     #^C177,R1
6788 040214  005737  002424                      TST     NO22BIT                 ;DO WE LOOK FOR AN EUB BIT?
6789 040220  001024                              BNE     2$                      ;BRANCH IF NOT NECESSARY
6790 040222  005737  002130                      TST     EUFLAG                  ;IS IT EUB?
6791 040226  001421                              BEQ     2$                      ;BRANCH IF NOT
6792 040230                                      PUSH    R0                      ;SAVE GENERATED ERROR ADDRESS
6793 040232  013702  002146                      MOV     CSRNO,R2                ;GET CSR NUMBER
6794 040236  052762  040000  172100              BIS     #BIT14,CSRADD(R2)       ;TURN ON EUB BIT CAREFULLY
6795 040244  016200  172100                      MOV     CSRADD(R2),R0           ;GET CSR CONTENTS
6796 040250  042762  040000  172100              BIC     #BIT14,CSRADD(R2)       ;TURN OFF EUB BIT CAREFULLY
6797 040256  042700  177037                      BIC     #^C740,R0               ;CLEAR EVERYTHING BUT ERROR ADDR
6798 040262  006300                              ASL     R0
6799 040264  006300                              ASL     R0                      ;SHIFT ADDR BITS 18-21 INTO POSITION
6800 040266  060001                              ADD     R0,R1                   ;ADD TO CURRENT ERROR ADDRESS
6801 040270                                      POP     R0
6802 040272  020001                2$:         CMP     R0,R1                   ;COMPARE REAL AND GENERATED ERR. ADDR.
6803 040274  001420                              BEQ     5$                      ;BRANCH IF THEY ARE THE SAME
6804 040276  005737  002134                      TST     INTFLAG                 ;INTERLEAVED?
6805 040302  001411                              BEQ     3$                      ;NO - WE HAVE AN ERROR
6806 040304  062700  000100                      ADD     #100,R0
6807 040310  005737  002136                      TST     INT64K                  ;64K INTERLEAVED MEMORY?
6808 040314  001002                              BNE     4$
6809 040316  062700  000100                      ADD     #100,R0
6810 040322  020001                4$:         CMP     R0,R1
6811 040324  001404                              BEQ     5$
6812 040326  005737  002064          3$:         TST     SKPERR                  ;ARE WE SUPPOSED TO SKIP ERROR P.O.?
6813 040332  001001                              BNE     5$                      ;YES - SKIP ERROR PRINTOUT
6814 040334  104462                              PERR36                          ;ELSE PRINT ERROR ADDRESS ERROR
6815 040336  010137  002430          5$:         MOV     R1,ERRADD               ;SAVE CSR'S ERROR ADDRESS
6816 040342  005037  002064                      CLR     SKPERR                  ;ENABLE THE ERROR PRINTOUT AGAIN
6817 040346                                      POP     R3,R2,R1,R0             ;RESTORE REGISTERS
6818 040356  000002                              RTI
```

```
 6821 040360                            CHKGEN: SUBTST<<SUBR    GENERATE CHECK BITS>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       SUBR    GENERATE CHECK BITS
                                        ;*****************************************************************************
 6822                                           ;CHECK BIT GENERATOR ROUTINE
 6823                                           ;CALLING SEQUENCE IS:
 6824                                           ;       MOV     #WORD1,SOURCE   ;SOURCE = ADDRESS OF DATA
 6825                                           ;       CALL    CHKGEN
 6826                                           ;
 6827                                           ;CHECK BITS RETURNED IN BITS 11-5 OF LOCATION CHECK
 6828                                           ;
 6829 040360                                    PUSH    R0,R1,R2,R3,R4,R5
 6830 040374  012702  000077                    MOV     #77,R2          ;DEFAULT CHECKBITS FOR DOUBLE WORD OF ZEROS
 6831 040400  012703  040466                    MOV     #CHKTAB,R3      ;ADDRESS OF CHECKBIT TABLE
 6832 040404  013705  002272                    MOV     SOURCE,R5       ;GET SOURCE ADDRESS
 6833 040410  012501                            MOV     (R5)+,R1        ;GET LSB'S
 6834 040412  011500                            MOV     (R5),R0         ;GET MSB'S
 6835
 6836 040414  006704                    1$:     SXT     R4              ;EXTEND SIGN OF DOUBLE WORD TO R4
 6837 040416  142304                            BICB    (R3)+,R4        ;ELIMINATE BITS THAT DON'T COUNT
 6838 040420  074402                            XOR     R4,R2           ;COMPLEMENT MASKED BITS IN CHECKBITS
 6839 040422  073027  000001                    ASHC    #1,R0           ;DOUBLE PRECISION LEFT SHIFT R0,,R1
 6840 040426  001372                            BNE     1$              ;LOOP TILL ALL BITS ARE CHECKED
 6841
 6842 040430  042702  177600                    BIC     #^C177,R2               ;KILL ALL JUNK BITS
 6843 040434  000302                            SWAB    R2              ;POSITION CHECKBITS IN BITS 11-5
 6844 040436  006202                            ASR     R2
 6845 040440  006202                            ASR     R2
 6846 040442  006202                            ASR     R2
 6847 040444  010237  002274                    MOV     R2,CHECK
 6848 040450                                    POP     R5,R4,R3,R2,R1,R0
 6849 040464  000207                            RETURN
```

```
 6852 040466                        CHKTAB: ;BYTE #3
 6853 040466      200                       .BYTE   ^C177           ;BIT 31
 6854 040467      301                       .BYTE   ^C076           ;BIT 30
 6855 040470      302                       .BYTE   ^C075           ;BIT 29
 6856 040471      203                       .BYTE   ^C174           ;BIT 28
 6857 040472      304                       .BYTE   ^C073           ;BIT 27
 6858 040473      205                       .BYTE   ^C172           ;BIT 26
 6859 040474      206                       .BYTE   ^C171           ;BIT 25
 6860 040475      307                       .BYTE   ^C070           ;BIT 24
 6861                                       ;BYTE #2
 6862 040476      310                       .BYTE   ^C067           ;BIT 23
 6863 040477      211                       .BYTE   ^C166           ;BIT 22
 6864 040500      212                       .BYTE   ^C165           ;BIT 21
 6865 040501      313                       .BYTE   ^C064           ;BIT 20
 6866 040502      214                       .BYTE   ^C163           ;BIT 19
 6867 040503      315                       .BYTE   ^C062           ;BIT 18
 6868 040504      316                       .BYTE   ^C061           ;BIT 17
 6869 040505      217                       .BYTE   ^C160           ;BIT 16
 6870                                       ;BYTE #1
 6871 040506      320                       .BYTE   ^C057           ;BIT 15
 6872 040507      221                       .BYTE   ^C156           ;BIT 14
 6873 040510      222                       .BYTE   ^C155           ;BIT 13
 6874 040511      323                       .BYTE   ^C054           ;BIT 12
 6875 040512      224                       .BYTE   ^C153           ;BIT 11
 6876 040513      325                       .BYTE   ^C052           ;BIT 10
 6877 040514      326                       .BYTE   ^C051           ;BIT  9
 6878 040515      227                       .BYTE   ^C150           ;BIT  8
 6879                                       ;BYTE #0
 6880 040516      340                       .BYTE   ^C037           ;BIT  7
 6881 040517      241                       .BYTE   ^C136           ;BIT  6
 6882 040520      242                       .BYTE   ^C135           ;BIT  5
 6883 040521      343                       .BYTE   ^C034           ;BIT  4
 6884 040522      244                       .BYTE   ^C133           ;BIT  3
 6885 040523      345                       .BYTE   ^C032           ;BIT  2
 6886 040524      346                       .BYTE   ^C031           ;BIT  1
 6887 040525      247                       .BYTE   ^C130           ;BIT  0
```

```
                                        SUBTST<<SUBR    MAPPER>>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    MAPPER
                                ;*****************************************************************************
                                ;
                                ;THIS SUBROUTINE MAPS THE MEMORY BANK (16K WORDS = 1 BANK)
                                ;IN R3 TO THE TEST PATTERN AREA (SUPERVISOR VIRTUAL (60000 - 157777) FOR
                                ;THE 11/44 AND 11/45-55; USER VIRTUAL (60000 - 157777) FOR ALL OTHER
                                ;PDP-11'S).
                                ;
                                ;CALL    MOV     BANKNO,R3               ;SET UP BANK ARGUEMENT
                                ;       CALL    MAPPER                  ;ACTUAL CALL
                                ;       RETURN                          ;ONLY RETURN

                                        ;SET SUPERVISOR/USER UP FOR 1 TO 1 MAP
040526                          MAPPER: PUSH    R0,R1,R2,R4,R5
040540  012700  172340                  MOV     #KIPAR0,R0              ;FIRST AREA TO MAP TO
040544  012701  172240                  MOV     #SIPAR0,R1              ;FIRST ADDRESS REGISTER
040550  012704  172200                  MOV     #SIPDR0,R4              ;FIRST DESCRIPTOR REGISTER
040554  005737  002426                  TST     NOSUPER                 ;CAN WE USE SUPERVISOR MODE?
040560  001404                          BEQ     4$                      ;YES, BRANCH
040562  012701  177640                  MOV     #UIPAR0,R1              ;FIRST ADDRESS REGISTER
040566  012704  177600                  MOV     #UIPDR0,R4              ;FIRST DESCRIPTOR REGISTER
040572  012702  077406          4$:     MOV     #77406,R2               ;CONSTANT FOR 4K PAGE, UP, R/W
040576  012705  000010                  MOV     #8.,R5                  ;COUNTER
040602  012021                  1$:     MOV     (R0)+,(R1)+             ;PUT IN SUPERVISOR ADDRESS
040604  010224                          MOV     R2,(R4)+                ;PUT IN SUPERVISOR DESCRIPTOR
040606  077503                          SOB     R5,1$                   ;LOOP TILL DONE
040610  012741  177600                  MOV     #177600,-(R1)           ;CORRECT LAST FIELD FOR PERIPHERALS PAGE

                                        ;SET UP SUPERVISOR/USER FOR TEST AREA
040614  022703  000170                  CMP     #120.,R3                ;MAP NOTHING (1 TO 1)?
040620  001512                          BEQ     3$                      ;YES - SKIP
040622  072327  000011                  ASH     #9.,R3                  ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
                                                                        ;FOR MEMORY MANAGEMENT = 1000
040626  012701  172246                  MOV     #SIPAR3,R1              ;SETUP FOR AUTO INCREMENTING
040632  005737  002426                  TST     NOSUPER                 ;DO WE HAVE SUPERVISOR MODE?
040636  001402                          BEQ     5$                      ;YES - BRANCH
040640  012701  177646                  MOV     #UIPAR3,R1              ;SETUP FOR AUTO INCREMENTING
040644  012702  000004          5$:     MOV     #4,R2                   ;COUNTER
040650  010321                  2$:     MOV     R3,(R1)+                ;PLUG IN PAR INFO
040652  062703  000200                  ADD     #200,R3                 ;BUMP ADDRESS 4K
040656  077204                          SOB     R2,2$                   ;LOOP TILL DONE
040660  005737  002232                  TST     SPLTCSR
040664  001442                          BEQ     9$
040666  162701  000010                  SUB     #10,R1
040672  010102                          MOV     R1,R2
040674  062702  000004                  ADD     #4,R2
040700  022737  000001  002232          CMP     #1,SPLTCSR
040706  001403                          BEQ     10$
040710  010200                          MOV     R2,R0
040712  010102                          MOV     R1,R2
040714  010001                          MOV     R0,R1
040716  012122                  10$:    MOV     (R1)+,(R2)+
040720  011112                          MOV     (R1),(R2)
040722  013700  002102                  MOV     BANKINDEX,R0
040726  005737  002136                  TST     INT64K
040732  001403                          BEQ     11$
```

```
6944 040734  012700  004000                    MOV     #4000,R0
6945 040740  000402                            BR      12$
6946 040742  012700  010000            11$:    MOV     #10000,R0
6947 040746  005737  002426            12$:    TST     NOSUPER
6948 040752  001403                            BEQ     13$
6949 040754  012701  177652                    MOV     #UIPAR5,R1
6950 040760  000402                            BR      14$
6951 040762  012701  172252            13$:    MOV     #SIPAR5,R1
6952 040766  060021                    14$:    ADD     R0,(R1)+
6953 040770  060011                            ADD     R0,(R1)
6954                                            ;IF WE ONLY HAVE AN 124K SYSTEM, WE DON'T WANT TO TEST THE
6955                                            ;LAST 4K, WHERE THE UNIBUS DEVICE PAGE IS. INSTEAD, THE
6956                                            ;PROGRAM WILL REMAP THE LAST 4K TO 8-12K. ALSO, IF THERE
6957                                            ;IS A BANK 177 ON AN 11/44, THE PROGRAM WILL REMAP THE LAST
6958                                            ;4K TO 8-12K FOR THE SAME REASON.
6959 040772  022737  000007  002526    9$:     CMP     #7,LASTBANK
6960 041000  001004                            BNE     7$
6961 041002  022703  010000                    CMP     #10000,R3
6962 041006  001017                            BNE     3$
6963 041010  000404                            BR      8$
6964 041012  022737  000177  002526    7$:     CMP     #177,LASTBANK
6965 041020  001012                            BNE     3$
6966 041022  005737  002426            8$:     TST     NOSUPER
6967 041026  001404                            BEQ     6$
6968 041030  013737  177652  177654            MOV     UIPAR5,UIPAR6
6969 041036  000403                            BR      3$
6970 041040  013737  172252  172254    6$:     MOV     SIPAR5,SIPAR6
6971 041046                            3$:     POP     R5,R4,R2,R1,R0
6972 041060  000207                            RETURN
6973                                            .SBTTL  TRAP    MAP KERNEL (ALMOST 1 TO 1) TRAP HANDLER
6974 041062                            $KMAP:  PUSH    R0,R1,R2,R3,R4
6975 041074  005000                            CLR     R0                      ;1ST AREA TO MAP TO
6976 041076  012701  172340                    MOV     #KIPAR0,R1              ;FIRST ADDRESS
6977 041102  012702  077406                    MOV     #77406,R2               ;CONSTANT FOR 4K PAGE,UP,R/W
6978 041106  012703  172300                    MOV     #KIPDR0,R3              ;1ST PAGE DESCRIPTOR REGISTER
6979 041112  012704  000010                    MOV     #8.,R4                  ;COUNTER
6980 041116  010021                    1$:     MOV     R0,(R1)+                ;PUT IN KERNEL ADDRESS
6981 041120  010223                            MOV     R2,(R3)+                ;PUT IN KERNEL DISCRIPTOR
6982 041122  062700  000200                    ADD     #200,R0                 ;ADD ADDRESS CONSTANT FOR 4K CHANGE
6983 041126  077405                            SOB     R4,1$                   ;LOOP TILL DONE
6984 041130  012741  177600                    MOV     #177600,-(R1)           ;THE PERIPHERALS PAGE TO KIPAR7
6985 041134  012741  177400                    MOV     #177400,-(R1)           ;AND NEXT LOWER PAGE TO KIPAR6
6992 041140                                    POP     R4,R3,R2,R1,R0
6993 041152  000002                            RTI
```

```
                                RELOCATE:SUBTST <<RELOCATE PROGRAM>>
                                ;*************************************************************************************
                                ;*SUBTEST       RELOCATE PROGRAM
                                ;*************************************************************************************
                                        IF #SW12 SET.IN @SWR THEN $RETURN ERROR
                                        IF APTFLAG IS TRUE OR ACTFLAG IS TRUE
                                          IF $PASS NE #0 THEN $RETURN ERROR
                                        END; OF IF APTFLAG
                                        BEGIN LOADERBANK
                                          FOR BANK := #1 TO LASTBANK
                                            CALL EXBANK
                                            IF ACFLAG IS TRUE AND PFLAG IS FALSE AND BMFLAG IS FALSE
                                              MOV       BANK,R0
                                              MOV       R0,LOADBANK
                                              MOV       LOADHOME,R1
                                              CALL      BANKMOV
                                              CALL      NEWLOAD          ;MAP NEW LOADER BANK IN KERNEL
                                              MOV       BANKINDEX,R1
                                              BIS       #BIT15,CONFIG+2(R1)      ;MARK LOADER
                                              BIC       #BIT13,CONFIG+2(R1)      ;INVALIDATE BACKGROUND PATTERN
                                              LEAVE LOADERBANK
                                            END ;OF IF ACFLAG
                                          END ;OF FOR BANK
                                          IF #SW13 OFF.IN @SWR
                                            TYPE        MSG075           ;RELOCATION NOT POSSIBLE
                                          END ;OF IF #SW13
                                          $RETURN ERROR
                                        END LOADERBANK
                                        BEGIN FINDBANK
7022 041354  013702  002526           MOV     LASTBANK,R2
7023 041360  006302                   ASL     R2
7024 041362  006302                   ASL     R2              ;R2 <- R2 * 4
7025 041364                             FOR R1 := #2*2 TO R2 BY #4
7026 041370                               IF #BIT7!BIT0 OFF.IN CONFIG(R1) ;IF NO ERRORS & NOT PROGRAM SPACE
7027 041400                                 IF #BIT15 OFF.IN CONFIG+2(R1) ;IF NOT LOADER BANK
7028 041410                                   IF CPUBIT SET.IN CONFIG(R1) ;IF ACCESSABLE
7029 041420                                     IF #BIT9 SET.IN CONFIG+2(R1) THEN LEAVE FINDBANK ;IF PARITY
7030 041430                                       IF #BIT6 SET.IN CONFIG(R1) AND #BIT7 OFF.IN CONFIG(R1)
7031                                                ;IF 1ST PROTECTABLE ECC BANK
7032 041450                                         LEAVE FINDBANK
7033 041452                                       END ;OF IF #BIT6
7034 041452                                       IF INHECC IS FALSE
7035 041460                                         SET   INHECC
7036 041466  010137  002510                        MOV   R1,INHBANK
7037 041472                                       END; OF IF INHECC
7038 041472                                   END ;OF IF CPUBIT
7039 041472                                 END ;OF IF #BIT15
7040 041472                               END ;OF IF #BIT7
7041 041472                             END ;OF FOR
7042 041502                             IF FULLREL IS FALSE
7043 041510                               IF INHECC IS TRUE
7044 041516  013701  002510                 MOV    INHBANK,R1
7045 041522  023727  002260  000030          CMP    REALPAT,#30        ;IS THIS PATTERN 30?
7046 041530  001423                         BEQ    RELENT1            ;YES - SKIP MESSAGE
7047 041532                                  TYPE   MSG123
7048 041536  000420                         BR     RELENT1
7049 041540                               END; OF IF INHECC
```

```
7050 041540                                          END; OF IF FULLREL
7051 041540  005037  002506                          CLR   !NHECC                ;MAKE SURE FLAG IS TURNED OFF!
7052 041544                                          IF #SW13 OFF.IN @SWR
7053 041554  023727  002260  000030                    CMP         REALPAT,#30   ;IS THIS PATTERN 30?
7054 041562  001402                                    BEQ         SKUB          ;YES - SKIP MESSAGE
7055 041564                                            TYPE        MSG075        ;RELOCATION NOT POSSIBLE
7056 041570                                          END ;OF IF #SW13
7057 041570                                SKUB:     $RETURN ERROR
7058 041574                                        END FINDBANK
7059 041574                                        CLEAR   INHECC                ;IF WE RELOCATED PROPERLY, THIS SHOULD BE OFF!
7060 041600  042761  020000  002626      RELENT1: BIC     #BIT13,CONFIG+2(R1)      ;INVALIDATE BACKGROUND PATTERN
7061 041606  005000                              CLR     R0
7062 041610  071027  000004                      DIV     #4,R0
7063 041614                                RELOC1: LET NEWBANK := R0
7064 041620  013737  002502  002504              MOV     PGMCSR,PGMCSR+2          ;SAVE CURRENT PGM. CSR
7065 041626  004737  042516                      CALL    USERMAP                  ;MAP NEWBANK TO USER PAR
7066 041632                                      USER                             ;ENTER USER MODE
7067 041640                                      BMOV    0,100000,SIZE            ;MOVE PROGRAM
7068 041652  104417                              KERNEL                           ;ENTER KERNEL MODE
7069 041654  005737  002424                      TST     N022BIT
7070 041660  001021                              BNE     JMPRL1
7071 041662  042737  000040  172516              BIC     #BIT5,MMR3               ;TURN OFF UNIBUS MAP
7072 041670  013700  002270                      MOV     NEWBANK,R0
7073 041674  006200                              ASR     R0
7074 041676                                      ON.ERROR
7075 041700  012737  100000  170200                MOV   #BIT15,MAPL0
7076 041706                                      END ;OF ON.ERROR
7077 041706  010037  170202                      MOV     R0,MAPH0
7078 041712  004737  042304                      CALL    LOWMAP                   ;SETUP LOWER 16K IN UNIBUS MAP
7079 041716  052737  000040  172516              BIS     #BIT5,MMR3               ;ENERGIZE UNIBUS MAP
7080 041724  042737  000001  177572  JMPRL1:  BIC     #BIT0,MMR0               ;DEENERGIZE MEMORY MANAGEMENT
7081 041732  004737  042600                      CALL    NEWKERNEL
7082 041736  013700  002270                      MOV     NEWBANK,R0
7083 041742  006300                              ASL     R0
7084 041744  006300                              ASL     R0                       ;R0 <- R0 * 4
7085 041746  016002  002624                      MOV     CONFIG(R0),R2
7086 041752  000302                              SWAB    R2
7087 041754  042702  177760                      BIC     #^C17,R2
7088 041760  006302                              ASL     R2
7089 041762  052737  000001  177572              BIS     #BIT0,MMR0               ;ENERGIZE MEMORY MANAGEMENT
7090 041770  010237  002502                      MOV     R2,PGMCSR                ;PUT NEW PGM. CSR INTO PGMCSR
7091 041774  032760  010000  002626              BIT     #BIT12,CONFIG+2(R0)      ;IS THE NEW BANK INTERLEAVED?
7092 042002  001412                              BEQ     1$                       ;BRANCH IF NOT INTERLEAVED
7093 042004  016002  002624                      MOV     CONFIG(R0),R2
7094 042010  042702  007777                      BIC     #^C170000,R2
7095 042014  072227  177775                      ASH     #-3,R2
7096 042020  052702  100000                      BIS     #BIT15,R2
7097 042024  050237  002502                      BIS     R2,PGMCSR
7098 042030                                1$:   SET     RLFLAG
7099 042036                                      $RETURN NOERROR
```

```
                                UNRELOCATE:SUBTST       <<UNRELOCATE PROGRAM>>
                                ;*************************************************************************
                                ;*SUBTEST        UNRELOCATE PROGRAM
                                ;*************************************************************************
                                        ;RESTORE LOADERS
042042                          1$:     PUSH    R0
042044  013701  002402                  MOV     LOADBANK,R1
042050  013700  002536                  MOV     LOADHOME,R0
042054  004737  042350                  CALL    BANKMOV
042060  004737  042646                  CALL    NEWLOAD                 ;MAP NEW LOADER BANK IN KERNEL SPACE
042064                                  PUSH    BANK
042070  013737  002402  002100          MOV     LOADBANK,BANK
042076  004737  042700                  CALL    EXBANK
042102  013701  002102                  MOV     BANKINDEX,R1
042106  042761  100000  002626          BIC     #BIT15,CONFIG+2(R1)     ;CLEAR LOADER FLAG
042114  013737  002536  002100          MOV     LOADHOME,BANK
042122  004737  042700                  CALL    EXBANK
042126  013701  002102                  MOV     BANKINDEX,R1
042132  042761  020000  002626          BIC     #BIT13,CONFIG+2(R1)     ;INVALIDATE BACKGROUND PATTERN
042140                                  POP     BANK
042144                                  CLEAR   INHECC                  ;MAKE SURE ECC TESTS ARE NOT INHIBITED.

                                        ;RESTORE BANK 0
042150  042737  020000  002626          BIC     #BIT13,CONFIG+2         ;INVALIDATE BACKGROUND PATTERN
042156                                  LET NEWBANK := #0
042162  004737  042516                  CALL    USERMAP                 ;MAP NEWBANK TO USER PAR
042166                                  USER                            ;ENTER USER MODE
042174                                  BMOV    0,100000,SIZE           ;MOVE PROGRAM
042206  104417                          KERNEL                          ;ENTER KERNEL MODE
042210  042737  000001  177572          BIC     #BIT0,MMR0              ;DEENERGIZE MEMORY MANAGEMENT
042216  004737  042600                  CALL    NEWKERNEL
042222  013737  002504  002502          MOV     PGMCSR+2,PGMCSR         ;RESTORE PREVIOUS PGM. CSR
042230  052737  000001  177572          BIS     #BIT0,MMR0              ;ENERGIZE MEMORY MANAGEMENT
042236  005037  002124                  CLR     RLFLAG
042242  005737  002424                  TST     NO22BIT
042246  001014                          BNE     1$
042250  042737  000040  172516          BIC     #BIT5,MMR3              ;TURN OFF UNIBUS MAP
042256                                  CLEAR   MAPL0,MAPH0
042266  004737  042304                  CALL    LOWMAP                  ;SETUP LOWER 16K OF UNIBUS MAP
042272  052737  000040  172516          BIS     #BIT5,MMR3              ;ENERGIZE UNIBUS MAP
042300                          1$:     POP     R0
042302  000207                          RETURN

                                LOWMAP: SUBTST  <<SETUP LOWER 16K OF UNIBUS MAP>>
                                ;*************************************************************************
                                ;*SUBTEST        SETUP LOWER 16K OF UNIBUS MAP
                                ;*************************************************************************
042304                                  PUSH    R0,R1,R2
042312  012700  170200                  MOV     #MAPL0,R0
042316  012701  170204                  MOV     #MAPL1,R1
042322  012702  000003                  MOV     #3,R2
042326  012011                  1$:     MOV     (R0)+,(R1)
042330  062721  020000                  ADD     #BIT13,(R1)+
042334  012021                          MOV     (R0)+,(R1)+
042336  077205                          SOB     R2,1$
042340                                  POP     R2,R1,R0
042346  000207                          RETURN
```

```
7155 042350                          BANKMOV:SUBTST  <<MOVE BANKS>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       MOVE BANKS
                                     ;*****************************************************************************
7156                                         ;MOVE 3/4 OF A BANK
7157                                         ;CALLING SEQUENCE
7158                                         ;R0 = DESTINATION BANK
7159                                         ;R1 = SOURCE BANK
7160 042350  104415                         SAVREG
7161 042352  004737  042516                 CALL    USERMAP
7162 042356  104416                         RESREG
7163 042360  104415                         SAVREG
7164 042362  072027  000011                 ASH     #9.,R0
7165 042366  072127  000011                 ASH     #9.,R1
7166 042372  012702  177650                 MOV     #UIPAR4,R2
7167 042376  012703  000200                 MOV     #200,R3
7168
7169 042402  010122                         MOV     R1,(R2)+                ;MAP 1ST HALF BANK
7170 042404  060301                         ADD     R3,R1                   ;BUMP BY 4K
7171 042406  010122                         MOV.    R1,(R2)+
7172 042410  060301                         ADD     R3,R1
7173
7174 042412  010022                         MOV     R0,(R2)+
7175 042414  060300                         ADD     R3,R0
7176 042416  010022                         MOV     R0,(R2)+
7177 042420  060300                         ADD     R3,R0
7178
7179 042422                                 USER
7180 042430                                 BMOV    100000,140000,SIZE/2    ;MOV 1ST HALF BANK
7181 042442  104417                         KERNEL                          ;ENTER KERNEL MODE
7182
7183 042444  012702  177650                 MOV     #UIPAR4,R2
7184
7185 042450  010122                         MOV     R1,(R2)+                ;MAP 2ND HALF BANK
7186 042452  060301                         ADD     R3,R1                   ;BUMP BY 4K
7187 042454  010122                         MOV     R1,(R2)+
7188 042456  060301                         ADD     R3,R1
7189
7190 042460  010022                         MOV     R0,(R2)+
7191 042462  060300                         ADD     R3,R0
7192 042464  010022                         MOV     R0,(R2)+
7193 042466  060300                         ADD     R3,R0
7194
7195 042470                                 USER
7196 042476                                 BMOV    100000,140000,SIZE/4    ;MOV 3ND FOURTH OF BANK
7197 042510  104417                         KERNEL                          ;ENTER KERNEL MODE
7198
7199 042512  104416                         RESREG
7200 042514  000207                         RETURN
```

```
7203 042516                          USERMAP:SUBTST  <<SUBR  MAP USER TO NEW BANK>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        SUBR    MAP USER TO NEW BANK
                                     ;*****************************************************************************
7204 042516  012701  177640                 MOV     #UIPAR0,R1              ;COPY KERNEL PAR'S & PDR'S (0-3)
7205 042522  012702  172340                 MOV     #KIPAR0,R2
7206 042526  012703  177600                 MOV     #UIPDR0,R3
7207 042532  012704  172300                 MOV     #KIPDR0,R4
7208 042536  012705  000004                 MOV     #4,R5
7209 042542  012221                  1$:    MOV     (R2)+,(R1)+
7210 042544  011423                         MOV     (R4),(R3)+
7211 042546  077503                         SOB     R5,1$
7212
7213 042550  013700  002270                 MOV     NEWBANK,R0
7214 042554  072027  000011                 ASH     #9.,R0                  ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
7215                                                                         ;FOR MEMORY MANAGEMENT = 1000
7216 042560  012705  000004                 MOV     #4,R5
7217 042564  010021                  2$:    MOV     R0,(R1)+                ;SETUP UIPAR(4-7)
7218 042566  062700  000200                 ADD     #200,R0                 ;BUMP ADDRESS 4K
7219 042572  011423                         MOV     (R4),(R3)+              ;SETUP UIPDR(4-7)
7220 042574  077505                         SOB     R5,2$
7221 042576  000207                         RETURN
7222
7223 042600                          NEWKERNEL:SUBTST        <<SUBR  SETUP KERNEL PAR'S FOR NEW BANK>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        SUBR    SETUP KERNEL PAR'S FOR NEW BANK
                                     ;*****************************************************************************
7224 042600                                 PUSH    R0,R1,R5
7225 042606  012700  172340                 MOV     #KIPAR0,R0
7226 042612  013701  002270                 MOV     NEWBANK,R1
7227 042616  072127  000011                 ASH     #9.,R1                  ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
7228                                                                         ;FOR MEMORY MANAGEMENT = 1000
7229 042622  012705  000004                 MOV     #4,R5
7230 042626  010120                  1$:    MOV     R1,(R0)+                ;SETUP KIPAR(0-3)
7231 042630  062701  000200                 ADD     #200,R1
7232 042634  077504                         SOB     R5,1$
7233 042636                                 POP     R5,R1,R0
7234 042644  000207                         RETURN
7235
7236 042646                          NEWLOAD:SUBTST  <<SUBR  SETUP KERNEL PAR'S FOR NEW LOADER BANK>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        SUBR    SETUP KERNEL PAR'S FOR NEW LOADER BANK
                                     ;*****************************************************************************
7237                                        ;R0 CONTAINS THE DESTINATION BANK
7238 042646                                 PUSH    R0,R1
7239 042652  012701  172350                 MOV     #KIPAR4,R1
7240 042656  072027  000011                 ASH     #9.,R0                  ;BANK 1 STARTS AT 100000 LESS 6 LSB'S (1000)
7241 042662  010021                         MOV     R0,(R1)+                ;SETUP KIPAR4
7242 042664  062700  000200                 ADD     #200,R0
7243 042670  010021                         MOV     R0,(R1)+                ;SETUP KIPAR5
7244 042672                                 POP     R1,R0
7245 042676  000207                         RETURN
```

```
                                EXBANK: SUBTST  <<SUBR  EXAMINE BANK>>
                                ;*****************************************************************************
                                ;*SUBTEST       SUBR    EXAMINE BANK
                                ;*****************************************************************************
                                ;DOES THE FOLLOWING:
                                ;(1)  SETS UP 'BANKINDEX'' AND R1 BASED ON VALUE OF 'BANK''.
                                ;(2)  SETS THE 'MKFLAG'' IF THE BANK IS ECC.
                                ;(3)  SETS THE 'KFLAG'' IF THE BANK IS MF11S-K.
                                ;(4)  SETS THE 'KPFLAG'' IF THE BANK IS THE PROTECTED REGION OF ECC MEMORY.
                                ;(5)  SETS THE 'ACFLAG'' IF THE BANK CAN BE ACCESSED BY THIS CPU.
                                ;(6)  SETS THE 'PFLAG'' IF THE BANK IS IN PROGRAM SPACE.
                                ;(7)  SETS THE  'RRFLAG'' IF RELOCATION IS REQUIRED TO TEST THIS BANK; HOWEVER,
                                ;     IT COMPLEMENTS THIS FLAG IF THE RELOCATION FLAG 'RLFLAG'' IS SET (THIS IS
                                ;     NECESSARY FOR THE USE OF THE RECURSIVE 'MODE'' SUBROUTINES). THE 'RRFLAG''
                                ;     IS ALWAYS SET TO DISABLE TESTING IF FIELD SERVICE MODE  'SELECTED BANKS''
                                ;     ARE BEING TESTED AND THIS BANK IS NOT SELECTED.
                                ;(8)  SETS THE 'BMFLAG'' IF THE BANK IS A BAD MEMORY;  HOWEVER, IT  COMPLEMENTS
                                ;     THIS FLAG IF THE 'WORST'' FLAG IS NOT SET  (THIS IS NECESSARY FOR THE USE
                                ;     OF THE RECURSIVE 'MODE'' SUBROUTINES).
                                ;(9)  SETS THE 'EUFLAG'' IF THE BANK HAS EXTENDED UNIBUS MEMORY.
                                ;(10) SETS THE ''INTFLAG'' IF THE BANK IS INTERLEAVED.
                                ;(11) SETS THE ''INT64K'' FLAG IF THE BANK IS INTERLEAVED ON 64K WORD BOUNDS.
                                ;(12) SETS THE 'SKIPMK'' FLAG IF THIS BANK IS INTERLEAVED, AND HAS ALREADY
                                ;     BEFN TESTED.
                                ;
042700                                  PUSH    R0,R1,R2
042706                                  CLEAR   MKFLAG,KPFLAG,KFLAG,EUFLAG
042726                                  SET     ACFLAG
042734                                  CLEAR   PFLAG,RRFLAG,BMFLAG
042750                                  CLEAR   INTFLAG,INT64K,SKIPMK
042764  013701  002100                  MOV     BANK,R1
042770  006301                          ASL     R1
042772  006301                          ASL     R1                      ;R1 <- R1 * 4
042774  010137  002102                  MOV     R1,BANKINDEX
043000  032761  000100  002624          BIT     #BIT6,CONFIG(R1)        ;PROTECTED REGION OF ECC MEMORY?
043006  001403                          BEQ     1$                      ;NO - SKIP
043010                                  SET     KPFLAG
043016  012700  000002          1$:     MOV     #BIT1,R0
043022                                  IF R0 SET.IN CPUBIT AND R0 OFF.IN CONFIG(R1)
043036  005037  002114                    CLR   ACFLAG
043042                                  END ;OF IF R0
043042  005737  002114                  TST     ACFLAG                  ;ACTIVE MEMORY?
043046  001415                          BEQ     12$                     ;BRANCH IF NOT
043050  016102  002626                  MOV     CONFIG+2(R1),R2
043054  000302                          SWAB    R2
043056  042702  177770                  BIC     #^C7,R2                 ;ISOLATE MEM TYPE BITS
043062  020227  000002                  CMP     R2,#2                   ;IS THIS AN ILLEGAL MEM TYPE?
043066  003005                          BGT     12$                     ;BRANCH IF NOT
043070                                  SET     BMFLAG                  ;SET BAD BANK FLAG
043076  000137  043340                  JMP     ENEXBK                  ;JUMP OVER REST OF FLAG TESTS
043102  032761  000400  002626  12$:    BIT     #BIT8,CONFIG+2(R1)      ;IS THIS EUB?
043110  001003                          BNE     2$                      ;BRANCH IF NOT
043112                                  SET     EUFLAG                  ;YES - SET EUB FLAG
043120  032761  001000  002626  2$:     BIT     #BIT9,CONFIG+2(R1)      ;IS THERE ECC THERE?
043126  001012                          BNE     3$                      ;NO - SKIP
043130                                  SET     MKFLAG          ;YES - SET MKFLAG
043136  032761  000400  002626          BIT     #BIT8,CONFIG+2(R1)      ;IS THIS MF11S-K MEMORY
```

```
7309 043144  001403                          BEQ     3$                      ;NO - IT'S MS11-M
7310 043146                                  SET     KFLAG                   ;YES - SET KFLAG
7311 043154  032761  000200  002624  3$:     BIT     #BIT7,CONFIG(R1)        ;BANK = PROGRAM SPACE?
7312 043162  001406                          BEQ     5$                      ;NO - SKIP
7313 043164                                  SET     PFLAG,RRFLAG
7314 043200  005737  002124          5$:     TST     RLFLAG                  ;IS PROGRAM RELOCATED?
7315 043204  001402                          BEQ     6$                      ;NO - SKIP
7316 043206  005137  002122                  COM     RRFLAG                  ;YES - COMPLEMENT RELOCATION REQUIRED FLAG
7317 043212  032761  000001  002624  6$:     BIT     #BIT0,CONFIG(R1)        ;ERRORS PRESENT IN THIS BANK?
7318 043220  001403                          BEQ     8$                      ;NO - SKIP
7319 043222                                  SET     BMFLAG
7320 043230  005737  002540          8$:     TST     WORST                   ;IS THIS A WORST FIRST PASS?
7321 043234  001002                          BNE     9$                      ;YES - SKIP
7322 043236  005137  002126                  COM     BMFLAG                  ;NO - COMPLEMENT BAD MEMORY FLAG
7323 043242                          9$:     IF SELONLY IS TRUE AND #BIT14 OFF.IN CONFIG+2(R1)
7324 043260                                    SET   RRFLAG
7325 043266                                  END ;OF IF SELONLY
7326 043266  032761  010000  002626          BIT     #BIT12,CONFIG+2(R1)     ;IS THIS BANK INTERLEAVED?
7327 043274  001421                          BEQ     ENEXBK                  ;BRANCH IF IT IS NOT
7328 043276                                  SET     INTFLAG
7329 043304  032761  004000  002626          BIT     #BIT11,CONFIG+2(R1)     ;IS THIS BANK INTERLEAVED WITH 64K BOARDS?
7330 043312  001403                          BEQ     10$                     ;BRANCH IF IT IS NOT
7331 043314                                  SET     INT64K
7332 043322  032761  000040  002624  10$:    BIT     #BIT5,CONFIG(R1)        ;SHOULD THIS BANK BE TESTED?
7333 043330  001403                          BEQ     ENEXBK                  ;BRANCH IF IT SHOULD
7334 043332                                  SET     SKIPMK
7335 043340                          ENEXBK: POP     R2,R1,R0                ;RESTORE REGISTERS
7336 043346  000207                          RETURN
```

```
7339 043350                          BANKOK: SUBTST  <<SUBR  BANK OK?>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        SUBR    BANK OK?
                                     ;*****************************************************************************
7340                                         ;TEST TO INSURE THAT THE TYPE OF MEMORY IN THE PRESENT BANK
7341                                         ;IS OF THE TYPE WE ARE TESTING 'TMFLAG'.
7342                                         ;RESULT IS RETURNED IN THE CONDITION CODES (OK = (=0)).
7343 043350  013700  002132                 MOV     TMFLAG,R0
7344 043354  005100                         COM     R0
7345 043356  013701  002116                 MOV     MKFLAG,R1
7346 043362  074001                         XOR     R0,R1
7347 043364  000207                         RETURN                          ;OK = (=OK)
7348
7349 043366                          INCRPT:
7350 043366                          INCPAT: SUBTST  <<SUBR  INCREMENT PATTERN TESTING >>
                                     ;*****************************************************************************
                                     ;*SUBTEST        SUBR    INCREMENT PATTERN TESTING
                                     ;*****************************************************************************
7351                                         ;INCREMENT THE PATTERN & SET UP THE CONDITION CODES
7352                                         ;RESULT - Z BIT SET INDICATES OVERFLOW
7353 043366  005237  002110                 INC     PATTERN
7354 043372  022737  000030  002110         CMP     #30,PATTERN             ;SET UP CONDITION CODES
7355 043400  000207                         RETURN                          ;NOT EQUAL TO ZERO IS GOOD (NO OVERFLOW)
7356
7357 043402                          SETPAT:
7358 043402                          HIPAT:  SUBTST  <<SUBR  SET HIGHEST PATTERN TESTING TYPE>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        SUBR    SET HIGHEST PATTERN TESTING TYPE
                                     ;*****************************************************************************
7359 043402  012737  000027  002110         MOV     #27,PATTERN             ;SET HIGHEST PATTERN
7360 043410  000207                         RETURN
7361
7362 043412                          INCBNK: SUBTST  <<SUBR  INCREMENT BANK & TEST>>
                                     ;*****************************************************************************
                                     ;*SUBTEST        SUBR    INCREMENT BANK & TEST
                                     ;*****************************************************************************
7363                                         ;RESULTS RETURNED IN CONDITION CODES
7364 043412  005237  002100                 INC     BANK
7365 043416  023737  002526  002100         CMP     LASTBANK,BANK           ;TOO FAR?
7366 043424  000207                         RETURN
```

```
7369 043426                          BOOT:   SUBTST  <<BOOTSTRAP ROUTINE>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       BOOTSTRAP ROUTINE
                                     ;*****************************************************************************
7370                                         ;INITIALIZE ALL CSR'S
7371                                         ;UNRELOCATE IF NECESSARY
7372                                         ;FLUSH OUT ANY DBE'S
7373                                         ;TURN OFF MEMORY MANAGEMENT
7374                                         ;TURN OFF THE UNIBUS MAP
7375                                         ;BOOT RK0 OR RK1
7376 043426  104472                         ECCINIT          ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
7377 043430                                  SET4    #BOOT1           ;TRAPS TO 4 GOTO BOOT1
7378 043436                                  IF RLFLAG IS TRUE THEN $CALL UNRELOCATE
7379 043450  004737  023412                  CALL    MT0030           ;FLUSH OUT DBE'S
7380 043454  104421                         DEENERGIZE               ;TURN OFF MEMORY MANAGEMENT
7381 043456  005737  002424                  TST     NO22BIT          ;IS THIS AN 11/44?
7382 043462  001003                         BNE     BOOT1
7383 043464  042737  000040  172516         BIC     #BIT5,MMR3                    ;TURN OFF THE UNIBUS MAP
7384 043472  005001                 BOOT1:  CLR     R1
7385 043474  000005                 1$:     RESET
7386 043476  012700  177406                 MOV     #177406,R0
7387 043502  010160  000004                 MOV     R1,4(R0)
7388 043506  012710  177400                 MOV     #177400,(R0)
7389 043512  012740  000005                 MOV     #5,-(R0)
7390 043516  105710                 2$:     TSTB    (R0)
7391 043520  100376                         BPL     2$
7392 043522  062701  020000                 ADD     #BIT13,R1
7393 043526  005710                         TST     (R0)
7394 043530  100761                         BMI     1$
7395 043532  005007                         CLR     PC
```

```
7398 043534                          EXIT:   SUBTST  <<HALT PROGRAM>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       HALT PROGRAM
                                     ;*****************************************************************************
7399 043534  004737  043566                 CALL    SHUTUP
7400 043540                          EXIT2:  IF APTFLAG IS TRUE OR ACTFLAG IS TRUE
7401 043554  000777                           BR      .
7402 043556                                  ELSE
7403 043560  000000                  $EXHALT:  HALT
7404 043562  000137  003630                   JMP     START
7405 043566                                  END ;OF IF APTFLAG
7406
7407 043566                          SHUTUP: SUBTST  <<SHUTDOWN DIAGNOSTIC>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       SHUTDOWN DIAGNOSTIC
                                     ;*****************************************************************************
7408                                         ;INITIALIZE ALL CSR'S
7409                                         ;UNRELOCATE
7410                                         ;FLUSH OUT DBE'S
7411                                         ;RESTORE LOADERS
7412                                         ;TURN OFF MEMORY MANAGEMENT
7413                                         ;UNMAP THE UNIBUS MAP
7417 043566  104472                         ECCINIT         ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
7418 043570                                  IF RLFLAG IS TRUE THEN $CALL UNRELOCATE
7419 043602                                  IF QUICK IS FALSE
7420 043610  004737  023412                   CALL  MT0030          ;FLUSH OUT DBE'S
7421 043614                                  END ;OF IF QUICK
7422 043614  012700  000001                 MOV     #1,R0           ;DESTINATION BANK
7423 043620  013701  002536                 MOV     LOADHOME,R1     ;SOURCE BANK
7424 043624  004737  042350                 CALL    BANKMOV
7425 043630  104421                         DEENERGIZE              ;TURN OFF MEMORY MANAGEMENT
7426 043632  005737  002424                 TST     NO22BIT         ;DOES THIS PDP-11 HAVE 22-BIT ADDR?
7427 043636  001003                         BNE     1$              ;BRANCH IF NOT
7428 043640  042737  000040  172516         BIC     #BIT5,MMR3              ;TURN OFF UNIBUS MAP
7432 043646  000207                 1$:     RETURN
7433
7434 043650                          APTDOWN:SUBTST  <<APT SHUTDOWN SEQUENCE>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       APT SHUTDOWN SEQUENCE
                                     ;*****************************************************************************
7435 043650                                  MAP     #0                      ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
7436 043664                                  TESTAREA                ;ENTER TEST MODE
7437 043672  012737  043650  060024         MOV     #APTDOWN,FIRST+24
7438 043700  012737  000340  060026         MOV     #340,FIRST+26
7439 043706  012737  000000  123650         MOV     #0,FIRST+APTDOWN
7440 043714  104417                         KERNEL                  ;ENTER KERNEL MODE
7441 043716  000000                 APTHLT: HALT
```

```
7444 043720                                     SUBTST  <<BLOCK MOVE SUBROUTINE>>
                                        ;*****************************************************************************
                                        ;*SUBTEST        BLOCK MOVE SUBROUTINE
                                        ;*****************************************************************************
7445                                    ;BLOCK3 HAS 3 ARGUEMENTS
7446                                    ;BLOCK2 HAS 2 ARGUEMENTS
7447                                    ;BLOCK1 HAS 1 ARGUEMENTS
7448                                    ;
7449                                    ;ALL ARE CALLED BY THE BMOV MACRO
7450                                    .ENABL  LSB
7451 043720                    BLOCK1:  PUSH    R0,R1,R2
7452 043726  012702  177640            MOV     #FASTCITY,R2
7453 043732  012701  000020            MOV     #16.,R1
7454 043736  000413                    BR      3$
7455
7456 043740                    BLOCK2:  PUSH    R0,R1,R2
7457 043746  012701  000020            MOV     #16.,R1
7458 043752  000404                    BR      2$
7459
7460 043754                    BLOCK3:  PUSH    R0,R1,R2
7461 043762  012501                    MOV     (R5)+,R1
7462 043764  012502            2$:     MOV     (R5)+,R2
7463 043766  012500            3$:     MOV     (R5)+,R0
7464
7465 043770  012022            1$:     MOV     (R0)+,(R2)+
7466 043772  077102                    SOB     R1,1$
7467 043774                            POP     R2,R1,R0
7468 044002  000205                    RTS     R5
7469                                    .DSABL  LSB
```

```
                                        .SBTTL  FIELD SERVICE MODE

                                FIELDSERVICE:SUBTST     <<SUBR  FIELD SERVICE COMMAND MODE>>
                                ;*****************************************************************************
                                ;*SUBTEST        SUBR    FIELD SERVICE COMMAND MODE
                                ;*****************************************************************************
044004  104415                          SAVREG
044006                                  TYPE    MSG020          ;FIELD SERVICE COMMAND MODE

044012                                  IF RLFLAG IS TRUE OR NOFSMODE IS TRUE
044026                                    TYPE  MSG048          ;NOT AVAILABLE NOW - TRY LATER!
044032  104416                            RESREG
044034  000207                            RETURN
044036                                  END ;OF IF RLFLAG
044036  005737  002514                  TST     CACHKN
044042  001402                          BEQ     1$
044044                                  PUSH    CONTRL          ;SAVE CACHE STATUS
044050                          1$:     PUSH    CSRNO,KAMIKAZE  ;SAVE CSR & KAMIKAZE STATUS
044060  104424                          CACHOFF                 ;TURN CACHE OFF
044062                                  SET     KAMIKAZE
044070                          FS1:    TYPE    MSG026          ;COMMAND:
044074  104414                          RDDEC                   ;READ A DECIMAL NUMBER
044076                                  POP     R0              ;COMMAND --> R0
044100  020027  000024                  CMP     R0,#20.
044104  101403                          BLOS    1$
044106                                  TYPE    MSG021
044112  000766                          BR      FS1
044114                          1$:     CASE R0
044124  044200                            FSCMD0                ;EXIT FIELD SERVICE COMMANDS
044126  044302                            FSCMD1                ;READ CSR
044130  044406                            FSCMD2                ;LOAD CSR
044132  044550                            FSCMD3                ;EXAMINE MEMORY
044134  045022                            FSCMD4                ;MODIFY MEMORY
044136  045340                            FSCMD5                ;SELECT BANK & PATTERN
044140  046172                            FSCMD6                ;TYPE CONFIGURATION MAP
044142  046200                            FSCMD7                ;BATTERY BACKUP - P/F TEST
044144  046520                            FSCMD8                ;SOB-A-LONG TEST
044146  047012                            FSCMD9                ;ERROR SUMMARY
044150  047216                            FCMD10                ;REFRESH TEST
044152  047510                            FCMD11                ;SET FILL COUNT
044154  047536                            FCMD12                ;ENTER KAMIKAZE MODE
044156  047560                            FCMD13                ;EXIT KAMIKAZE MODE
044160  047600                            FCMD14                ;TURN CACHE OFF
044162  047622                            FCMD15                ;TURN CACHE ON
044164  047640                            FCMD16                ;TEST ONLY SELECTED BANKS
044166  047724                            FCMD17                ;RESUME TESTING ALL BANKS
044170                                  END ;OF CASE
044176  000734                          BR      FS1
```

```
                                FSCMD0: SUBTST  <<COMMAND 0     EXIT>>
                                ;*****************************************************************************
                                ;*SUBTEST        COMMAND 0       EXIT
                                ;*****************************************************************************
7523 044200                             TYPE    MSG103          ;LEAVING FIELD SERVICE MODE
7524 044204  062706  000002             ADD     #2,SP
7525 044210                             IF SKIPKAMI IS TRUE
7526 044216  062706  000002               ADD   #2,SP                   ;THROW AWAY OLD KAMIKAZE FLAG
7527 044222  005037  002006               CLR   SKIPKAMI
7528 044226                             ELSE
7529 044230                               POP   KAMIKAZE                ;RESTORE OLD KAMIKAZE FLAG
7530 044234                             END ;OF IF SKIPKAMI
7531 044234                             POP     CSRNO
7532 044240  005737  002514             TST     CACHKN
7533 044244  001414                     BEQ     RES0
7534 044246                             IF CACHKN EQ CACHKF           ;IF CACHE IS OFF
7535 044256  062706  000002               ADD   #2,SP                   ;THROW AWAY CACHE STATUS
7536 044262                             ELSE
7537 044264  005737  002514               TST   CACHKN
7538 044270  001402                       BEQ   RES0
7539 044272                               POP   CONTRL                  ;RESTORE CACHE STATUS
7540 044276                             END ;OF IF CACHKN
7541 044276  104416             RES0:   RESREG
7542 044300  000207                     RETURN
7543
7544 044302                     FSCMD1: SUBTST  <<FS    COMMAND 1       READ CSR>>
                                ;*****************************************************************************
                                ;*SUBTEST        FS      COMMAND 1       READ CSR
                                ;*****************************************************************************
7545 044302  004737  047766             CALL    WHICHCSR
7546 044306  010637  002266             MOV     SP,FSSTACK
7547 044312                             SET4    #RES1           ;TRAPS TO 4 GOTO RES1
7548 044320  104426                     READCSR
7549 044322                             SET     NOERROR
7550 044330  104026                     ERROR   +26             ;USE ERROR ROUTINE FOR PRINTOUT
7551 044332                             RES4                    ;RESET TRAPS TO 4 TO DEFAULT
7552 044352  000207                     RETURN
7553 044354                     RES1:   TYPE    MSG025          ;THIS CSR DOES NOT EXIST
7554 044360  013706  002266             MOV     FSSTACK,SP
7555 044364                             RES4                    ;RESET TRAPS TO 4 TO DEFAULT
7556 044404  000207                     RETURN
```

```
 7559 044406                        FSCMD2: SUBTST  <<FS    COMMAND 2       LOAD CSR>>
                                    ;*****************************************************************************
                                    ;*SUBTEST       FS      COMMAND 2       LOAD CSR
                                    ;*****************************************************************************
 7560 044406  004737  047766               CALL    WHICHCSR
 7561 044412  010637  002266               MOV     SP,FSSTACK
 7562 044416                               SET4    #RES2           ;TRAPS TO 4 GOTO RES2
 7563 044424  104426                       READCSR
 7564 044426                               TYPE    MSG027
 7565 044432                               SET     NOERROR
 7566 044440  104026                       ERROR   +26             ;USE ERROR ROUTINE FOR PRINTOUT
 7567 044442                               RES4                    ;RESET TRAPS TO 4 TO DEFAULT
 7568 044462                               TYPE    MSG023          ;FIRST CSR WORD
 7569 044466  104413                       RDOCT                   ;READ AN OCTAL NUMBER
 7570 044470                               POP     CSR             ;PUT IN IN LOC "CSR"
 7571 044474  104425                       LOADCSR
 7572 044476  104426                       READCSR
 7573 044500                               TYPE    MSG028
 7574 044504                               SET     NOERROR
 7575 044512  104026                       ERROR   +26             ;USE FOR PRINTOUT - NOT AN ERROR
 7576 044514  000207                       RETURN
 7577 044516                        RES2:  TYPE    MSG025          ;THIS CSR DOES NOT EXIST
 7578 044522  013706  002266               MOV     FSSTACK,SP
 7579 044526                               RES4                    ;RESET TRAPS TO 4 TO DEFAULT
 7580 044546  000207                       RETURN
```

```
7583 044550                                FSCMD3: SUBTST  <<FS    COMMAND 3       EXAMINE MEMORY>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       FS      COMMAND 3       EXAMINE MEMORY
                                           ;*****************************************************************************
7584 044550                                        PUSH    BANK,NOPAR,PARTHERE,4
7585 044570  012737  000002  002074               MOV     #2,NOPAR          ;INDICATE PARITY ACTION
7586 044576                                        TYPE    MSG029            ;EXAMINE MEMORY
7587 044602                                1$:     TYPE    MSG031            ;PHYSICAL ADDRESS (0-17775776)??
7588 044606  104413                               RDOCT                     ;READ OCTAL NUMBER ONTO STACK & $HIOCT
7589 044610  013737  060556  002100               MOV     $HIOCT,BANK       ;PUT MSB'S IN BANK
7590 044616                                        POP     R0                ;PUT LSB'S IN R0
7591 044620  000241                               CLC
7592 044622  006100                               ROL     R0
7593 044624  006137  002100                       ROL     BANK
7594 044630  000241                               CLC
7595 044632  006000                               ROR     R0
7596 044634  023737  002100  002526               CMP     BANK,LASTBANK     ;CHECK FOR BANK TOO HIGH
7597 044642  003357                               BGT     1$                ;BRANCH IF TRUE
7598 044644  062700  060000                       ADD     #FIRST,R0
7599 044650  032700  000001                       BIT     #BIT0,R0          ;CHECK FOR ODD ADDRESS
7600 044654  001352                               BNE     1$                ;BRANCH IF ODD ADDRESS
7601 044656  020027  157776                       CMP     R0,#LAST          ;CHECK FOR ADDRESS OVER 16K
7602 044662  101347                               BHI     1$                ;BRANCH IF OVER 16K
7603 044664  012737  044736  002264               MOV     #3$,PARTHERE      ;INCASE OF ABORTS
7604 044672                                        SET4    #4$               ;TRAPS TO 4 GOTO 4$
7605 044700                                        MAP     BANK                      ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
7606 044714                                        TESTAREA                  ;ENTER TEST MODE
7607 044722  011001                               MOV     (R0),R1
7608 044724  104417                               KERNEL                    ;ENTER KERNEL MODE
7609 044726                                        TYPOCS  R1
7610 044734  000410                               BR      EXCMD3
7611
7612 044736                                3$:     TYPE    MSG032                    ;PARITY ABORT
7613 044742  000405                               BR      EXCMD3
7614
7615 044744  062706  000004                4$:     ADD     #4,SP                     ;FIX STACK
7616 044750                                        TYPE    MSG033                    ;TIMEOUT TRAP
7617 044754  000400                               BR      EXCMD3
7618
7619 044756  104417                       EXCMD3: KERNEL                    ;ENTER KERNEL MODE
7620 044760                                        POP     4,PARTHERE,NOPAR,BANK
7621 045000                                        RES4                      ;RESET TRAPS TO 4 TO DEFAULT
7622 045020  000207                               RETURN
```

```
                                FSCMD4: SUBTST  <<FS    COMMAND 4       MODIFY MEMORY>>
                                ;*****************************************************************************
                                ;*SUBTEST        FS      COMMAND 4       MODIFY MEMORY
                                ;*****************************************************************************
                                        PUSH    BANK,NOPAR,PARTHERE,4
045022 012737 000003 002074             MOV     #3,NOPAR                ;INDICATE PARITY ACTION
045050                                  TYPE    MSG036                  ;MODIFY MEMORY
045054                          1$:     TYPE    MSG031          ;PHYSICAL ADDRESS (0-17775776)??
045060 104413                           RDOCT                   ;READ OCTAL NUMBER ONTO STACK & $HIOCT
045062 013737 060556 002100             MOV     $HIOCT,BANK     ;PUT MSB'S IN BANK
045070                                  POP     R0              ;PUT LSB'S IN R0
045072 000241                           CLC
045074 006100                           ROL     R0
045076 006137 002100                    ROL     BANK
045102 000241                           CLC
045104 006000                           ROR     R0
045106                                  IF BANK GT LASTBANK THEN GOTO 1$  ;CHECK FOR BANK TOO HIGH
045116 062700 060000                    ADD     #FIRST,R0
045122                                  IF #BIT0 SET.IN R0 THEN GOTO 1$ ;CHECK FOR ODD ADDRESS
045130                                  IF R0 HI #LAST THEN GOTO 1$     ;CHECK FOR ADDRESS OVER 16K
045136 012737 045204 002264             MOV     #3$,PARTHERE            ;INCASE OF ABORTS
045144                                  SET4    #4$             ;TRAPS TO 4 GOTO 4$
045152                                  MAP     BANK                    ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
045166 104511                           INVALIDATE
045170                                  TESTAREA                ;ENTER TEST MODE
045176 011001                           MOV     (R0),R1
                                        ;GETTING HERE MEANS WE GOT LUCKY - NO TRAPS
045200 104417                           KERNEL                  ;ENTER KERNEL MODE
045202 000410                           BR      5$

045204                          3$:     TYPE    MSG032                  ;PARITY ABORT
045210 000431                           BR      EXCMD4                  ;EXIT

045212 062706 000004            4$:     ADD     #4,SP           ;FIX STACK
045216                                  TYPE    MSG033          ;TIMEOUT TRAP
045222 000424                           BR      EXCMD4                  ;EXIT

045224                          5$:     TYPE    MSG037                  ;OLD DATA WAS
045230                                  TYPOCS  R1                      ;PRINT IT
045236                                  TYPE    MSG039                  ;INPUT NEW DATA
045242 104413                           RDOCT                           ;READ ON OCTAL NUMBER ONTO THE STACK
045244                                  POP     R1                      ;GET NEW NUMBER
045246                                  TESTAREA                ;ENTER TEST MODE
045254 010110                           MOV     R1,(R0)                 ;PUT IT IN MEMORY
045256 011001                           MOV     (R0),R1                 ;READ IT AGAIN
045260 104417                           KERNEL                  ;ENTER KERNEL MODE
045262                                  TYPE    MSG038                  ;DATA IS NOW
045266                                  TYPOCS  R1                      ;PRINT IT

045274 104417                   EXCMD4: KERNEL                  ;ENTER KERNEL MODE
045276                                  POP     4,PARTHERE,NOPAR,BANK
045316                                  RES4                    ;RESET TRAPS TO 4 TO DEFAULT
045336 000207                           RETURN
```

```
7677 045340                                FSCMD5: SUBTST  <<FS    COMMAND 5       SELECT BANK & PATTERN>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       FS      COMMAND 5       SELECT BANK & PATTERN
                                           ;*****************************************************************************
7678 045340                                        PUSH    BANK,PATTERN,TESTADD,PCBUMP,TKVEC,TKVEC+2
7679 045370  010637  002266                   MOV     SP,FSSTACK                  ;SAVE LAST GOOD STACK POINTER
7680 045374                                   TYPE    MSG040                      ;SELECT BANK & PATTERN TEST
7681 045400                                1$:TYPE    MSG030                      ;BANK(0-177)?
7682 045404  104413                           RDOCT                               ;READ AN OCTAL NUMBER ONTO THE STACK
7683 045406                                   POP     BANK                        ;PUT IT IN BANK
7684 045412                                   IF BANK GT LASTBANK THEN GOTO 1$  ;CHECK FOR BANK TOO HIGH
7685
7686 045422  013701  002100                   MOV     BANK,R1
7687 045426  006301                           ASL     R1
7688 045430  006301                           ASL     R1
7689 045432                                   IF CPUBIT OFF.IN CONFIG(R1)
7690 045442                                     TYPE  MSG041                      ;BANK NOT ACCESSABLE
7691 045446                                     GOTO 1$
7692 045450                                   END ;OF IF
7693
7694 045450                                2$:TYPE    MSG042                      ;PATTERN(0-35)?
7695 045454  104413                           RDOCT                               ;READ AN OCTAL NUMBER ONTO THE STACK
7696 045456                                   POP     PATTERN                     ;PUT IT IN PATTERN
7697 045462                                   IF PATTERN GT #35 THEN GOTO 2$  ;CHECK FOR PATTERN TO HIGH
7698 045472                                   IF PATTERN EQ #0
7699 045500                                     TYPE  MSG043                      ;PATTERN 0 DATA IS?
7700 045504  104413                             RDOCT                           ;READ AN OCTAL NUMBER ONTO THE STACK
7701 045506                                     POP   R2                        ;PUT IT IN R2
7702 045510                                   END ;OF IF
7703
7704
7705 045510                                   MAP     BANK                        ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
7706 045524  104511                           INVALIDATE
7707 045526  004737  042700                   CALL    EXBANK                      ;SET NEW MARGINS
7708 045532                                   IF RRFLAG IS TRUE
7709 045540                                     TYPE  MSG049                      ;BANK REQUIRES RELOCATION
7710 045544                                     GOTO CMD5C
7711 045546                                   END ;OF IF RRFLAG
7712 045546                                   TYPE    MSG046          ;TO ESCAPE TYPE ANY KEY!
7713 045552  012737  046102  000060          MOV     #CMD5C,TKVEC
7714 045560  012737  000340  000062          MOV     #340,TKVEC+2
7715 045566  017700  135012                   MOV     @$TKB,R0                    ;KILL ANY OLD INTERRUPT
7716 045572  042737  000200  177776          BIC     #BIT7,PSW                   ;LOWER CPU PRIORITY TO 140
7717 045600  052777  000100  134774          BIS     #BIT6,@$TKS                 ;ENABLE KEYBOARD INTERRUPTS
7718
7719
7720 045606                                   SET     HEADER,MUT
7721 045622  013701  002100           CMD5B: MOV     BANK,R1
7722 045626  006301                           ASL     R1
7723 045630  006301                           ASL     R1
7724 045632  005037  002232                   CLR     SPLTCSR
7725 045636  005037  002256                   CLR     PASFLG
7726 045642  012737  060000  002362          MOV     #FIRST,TESTADD
7727 045650  012737  060002  002364          MOV     #FIRST+2,TESTADD+2
7728 045656                                   IF #BIT12 SET.IN CONFIG+2(R1)
7729 045666  005237  002232                     INC   SPLTCSR
7730 045672                                     MAP   BANK
```

```
7731 045706 012737 120000 002364          MOV   #120000,TESTADD+2
7732 045714 005037 002232                 CLR   SPLTCSR
7733 045720                             END; OF IF #BIT12
7734 045720                             IF #SW0 SET.IN @SWR
7735 045730 104470                        ECCDIS                        ;DISABLE ERROR CORRECTION
7736 045732                             ELSE
7737 045734                               PUSH  CSRNO
7738 045740 104502                        CLRCSR                        ;CLEAR CSRS
7739 045742                               POP   CSRNO
7740 045746                             END ;OF IF
7741 045746 005237 002232               INC     SPLTCSR
7742 045752 012737 000002 002074        MOV     #2,NOPAR                ;PARITY ACTION
7743 045760 012737 000002 002276        MOV     #2,PCBUMP               ;TRAPS ADD 2 TO PC
7744 045766 013700 002110               MOV     PATTERN,R0
7745 045772 006300                      ASL     R0
7746 045774 004770 046006               CALL    @FSPAT(R0)
7747 046000 005037 002074               CLR     NOPAR
7748 046004 000706                      BR      CMD5B                   ;LOOP TILL KEYBOARD INTERRUPT
7749
7750 046006 016760                FSPAT: MT0000  ;<1 SEC          DATA PATTERN TEST
7751 046010 017036                       MT0001  ;<1 SEC          ADDRESS TEST
7752 046012 017154                       MT0002  ;<1 SEC          COMPLEMENT ADDRESS TEST
7753 046014 017312                       MT0003  ; 1 SEC          3 XOR 9 WORST CASE NOISE TEST
7754 046016 017542                       MT0004  ; 1 SEC          ROTATING ZEROS TEST
7755 046020 017662                       MT0005  ; 1 SEC          ROTATING ONES TEST
7756 046022 020014                       MT0006  ;<1 SEC          INITIAL DATA TEST
7757 046024 020050                       MT0007  ;<1 SEC          ADDRESS BIT TEST
7758 046026 020112                       MT0010  ;<1 SEC          BYTE ADDRESSING TEST
7759 046030 020146                       MT0011  ;<2 SEC          CREATE SINGLE BIT ERROR TEST
7760 046032 020214                       MT0012  ;<1 SEC          WRITE BYTE CLEARS SBE TEST
7761 046034 020310                       MT0013  ; 1 SEC          CREATE DOUBLE BIT ERROR TEST
7762 046036 020364                       MT0014  ; 1 SEC          WRITE INHIBIT DURING DATIP WITH DBE
7763 046040 020442                       MT0015  ; 1 SEC          WRITE INHIBIT OF BYTE WITH DBE
7764 046042 020510                       MT0016  ;<1 SEC          WRITE INHIBIT OF WORD WITH DBE
7765 046044 020556                       MT0017  ;<1 SEC          HOLDING 1'S & 0'S TEST
7766 046046 020600                       MT0020  ;<1 SEC          MARCHING 1'S & 0'S IN CHECK BITS
7767 046050 021630                       MT0021  ; 1 SEC          MARCHING 0'S & 1'S TEST
7768 046052 022052                       MT0022  ;10 SEC          REFRESH & SHIFTING DIAGONAL TEST
7769 046054 022104                       MT0023  ;10 SEC          SHIFTING DIAGONAL TEST
7770 046056 022150                       MT0024  ;20 SEC          FAST GALLOPING PATTERN TEST
7771 046060 022402                       MT0025  ;<1 SEC          INTERRUPT ENABLE TEST
7772 046062 022450                       MT0026  ;<1 SEC          RANDOM DATA TEST
7773 046064 022740                       MT0027  ; 1 SEC          UNIQUE BANK TEST
7774 046066 023412                       MT0030  ; 1 SEC          FLUSH OUT DBE'S TEST
7775 046070 023712                       MT0031  ; 3 SEC          SOB-A-LONG TEST
7776 046072 024102                       MT0032  ;<1 SEC          WRITE RECOVERY TEST
7777 046074 024430                       MT0033  ;35 SEC          BRANCH GOBBLE TEST
7778 046076 024616                       MT0034  ; 1 SEC          SOFT ERROR TEST
7779 046100 024762                       MT0035  ;<1 SEC          WORST CASE NOISE PARITY TEST
7780
7781 046102 013706 002266         CMD5C: MOV     FSSTACK,SP                ;RECOVER OLD STACK POINTER
7782 046106 042777 000100 134466         BIC     #BIT6,@$TKS
7783 046114                              POP     TKVEC+2,TKVEC
7784 046124 117700 134454                MOVB    @$TKB,R0                  ;GET CHARACTER TO GET RID OF FLAG
7785 046130                              POP     PCBUMP,TESTADD
7786 046140                              POP     PATTERN,BANK
7787 046150                              MAP     BANK                      ;REMAP OLD BANK
```

```
 7788 046164  004737  042700                  CALL    EXBANK
 7789 046170  000207                          RETJRN
```

```
 7791 046172                        FSCMD6: SUBTST  <<FS     COMMAND 6       TYPE CONFIGURATION MAP>>
                                    ;*****************************************************************************
                                    ;*SUBTEST       FS      COMMAND 6       TYPE CONFIGURATION MAP
                                    ;*****************************************************************************
 7792 046172  004737  035254               CALL    PCONFIG
 7793 046176  000207                        RETURN
 7794
```

```
 7797 046200                                FSCMD7: SUBTST  <<FS    COMMAND 7       BATTERY BACKUP TEST>>
                                                    ;*****************************************************************************
                                                    ;*SUBTEST        FS      COMMAND 7       BATTERY BACKUP TEST
                                                    ;*****************************************************************************
 7798 046200                                            PUSH    BANK,PATTERN,CSRNO,TKVEC,TKVEC+2
 7799 046224                                            SET     FS7FLAG
 7800 046232                                            IF #SW0 SET.IN @SWR
 7801 046242  104470                                      ECCDIS                         ;DISABLE ERROR CORRECTION
 7802 046244                                            ELSE
 7803 046246  104502                                      CLRCSR                         ;CLEAR CSRS
 7804 046250                                            END ;OF IF
 7805 046250                                            TYPE    MSG050          ;BATTERY BACKUP TEST
 7806                                                                           ;WRITING & CHECKING ADDRESS PATTERN AS BACKGROUND
 7807 046254  004737  022740                            CALL    MT0027                 ;CREATE & CHECK BACKGROUND
 7808 046260  012737  000012  002310                    MOV     #10.,TIME
 7809 046266                                            TYPE    MSG052                 ;REMOVE SYSTEM POWER FOR
 7810 046272                                            TYPDEC  TIME
 7811 046300                                            TYPE    MSG053                 ;SECONDS MAX!
 7812 046304                                            TYPE    MSG046                 ;TO ESCAPE TYPE ANY KEY
 7813 046310  012737  046352  000060                    MOV     #CMD7A,TKVEC
 7814 046316  012737  000340  000062                    MOV     #340,TKVEC+2
 7815 046324  017700  134254                            MOV     @$TKB,R0               ;KILL ANY OLD INTERRUPT
 7816 046330  042737  000200  177776                    BIC     #BIT7,PSW              ;LOWER CPU PRIORITY TO 140
 7817 046336  052777  000100  134236                    BIS     #BIT6,@$TKS            ;ENABLE KEYBOARD INTERRUPTS
 7818 046344  010637  002266                            MOV     SP,FSSTACK
 7819
 7820 046350  000001                                    WAIT
 7821
 7822 046352  013706  002266            CMD7A:  MOV     FSSTACK,SP             ;RECOVER OLD STACK POINTER
 7823 046356  042777  000100  134216              BIC     #BIT6,@$TKS
 7824 046364                                      POP     TKVEC+2,TKVEC
 7825 046374  117700  134204                      MOVB    @$TKB,R0               ;GET CHARACTER TO GET RID OF FLAG
 7826 046400                                      TYPE    MSG054                 ;NOW STARTING READ TEST OF MEMORY BANKS
 7827 046404                                      FOR BANK := #0 TO LASTBANK
 7828 046410  004737  042700                        CALL  EXBANK
 7829 046414                                        IF ACFLAG IS TRUE AND RRFLAG IS FALSE
 7830 046430  104511                                  INVALIDATE
 7831 046432                                          LET R2 := BANK
 7832 046436  012700  060000                          MOV #FIRST,R0
 7833 046442  010004                                  MOV R0,R4
 7834 046444  012701  040000                          MOV #SIZE,R1
 7835 046450  010103                                  MOV R1,R3
 7836 046452  004737  025322                          CALL        SUPD03
 7837 046456                                        END ;OF IF ACFLAG
 7838 046456                                      END ;OF FOR BANK
 7839 046472                                      TYPE    MSG047                 ;TEST COMPLETE
 7840 046476  005037  002416                      CLR     FS7FLAG
 7841 046502                                      POP     CSRNO,PATTERN,BANK
 7842 046516  000207                              RETURN
 7843
```

```
 7846 046520                        FSCMD8: SUBTST  <<FS    COMMAND 8       SOB-A-LONG TEST>>
                                    ;*****************************************************************************
                                    ;*SUBTEST       FS      COMMAND 8       SOB-A-LONG TEST
                                    ;*****************************************************************************
 7847 046520                                PUSH    BANK,PATTERN,TKVEC,TKVEC+2,NOPAR
 7848 046544  010637  002266                MOV     SP,FSSTACK                  ;SAVE LAST GOOD STACK POINTER
 7849 046550                                TYPE    MSG055                      ;SOB-A-LONG TEST
 7850
 7851 046554                                IF #SW0 SET.IN @SWR
 7852 046564  104470                          ECCDIS                            ;DISABLE ERROR CORRECTION
 7853 046566                                ELSE
 7854 046570  104502                          CLRCSR                            ;CLEAR CSRS
 7855 046572                                END ;OF IF
 7856 046572                                TYPE    MSG056                      ;BELL = EACH PASS COMPLETE
 7857
 7858 046576                                TYPE    MSG046                      ;TO ESCAPE TYPE ANY KEY!
 7859 046602  012737  046726  000060        MOV     #CMD8C,TKVEC
 7860 046610  012737  000340  000062        MOV     #340,TKVEC+2
 7861 046616  017700  133762                MOV     @$TKB,R0                    ;KILL ANY OLD INTERRUPT
 7862 046622  042737  000200  177776        BIC     #BIT7,PSW                   ;LOWER CPU PRIORITY TO 140
 7863 046630  052777  000100  133744        BIS     #BIT6,@$TKS                 ;ENABLE KEYBOARD INTERRUPTS
 7864
 7865
 7866 046636                                SET     HEADER,MUT
 7867
 7868 046652                        CMD8B:  FOR BANK := #0 TO LASTBANK
 7869 046656  004737  042700                  CALL EXBANK
 7870 046662                                  IF ACFLAG IS TRUE AND RRFLAG IS FALSE
 7871 046676  104511                            INVALIDATE
 7872 046700  004737  023712                    CALL MT0031
 7873 046704                                  END ;OF IF ACFLAG
 7874 046704                                END ;OF FOR BANK
 7875 046720                                TYPE    $BELL                       ;RING BELL
 7876 046724                                GOTO    CMD8B
 7877
 7878 046726  013706  002266        CMD8C:  MOV     FSSTACK,SP                  ;RECOVER OLD STACK POINTER
 7879 046732  042777  000100  133642        BIC     #BIT6,@$TKS
 7880 046740  117700  133640                MOVB    @$TKB,R0                    ;READ CHAR TO KILL FLAG
 7881 046744                                POP     NOPAR,TKVEC+2,TKVEC,PATTERN,BANK
 7882 046770                                MAP     BANK                        ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
 7883 047004  004737  042700                CALL    EXBANK
 7884 047010  000207                        RETURN
```

```
7887 047012                          FSCMD9: SUBTST  <<FS    COMMAND 9       ERROR SUMMARY>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       FS      COMMAND 9       ERROR SUMMARY
                                     ;*****************************************************************************
7888 047012                                  PUSH    R0,R2,R3,BANK
7889 047024  013737  061264  002404          MOV     $PASS,TEMP
7890 047032  005337  002404                  DEC     TEMP
7891 047036                                  TYPDEC  TEMP
7892 047044                                  TYPE    MSG125                  ;PASSES COMPLETED
7893 047050                                  TYPDEC  $ERTTL
7894 047056                                  TYPE    MSG079                  ;ERROR(S) DETECTED
7895 047062                                  IF $ERTTL NE #0
7896 047070  005037  002304                    CLR   SUCCESS
7897 047074                                    FOR BANK := #0 TO LASTBANK
7898 047100  013703  002100                      MOV BANK,R3
7899 047104  070327  000004                      MUL #4,R3
7900 047110                                      IFB CONFIG+2(R3) NE #0
7901 047116                                        IF SUCCESS IS FALSE
7902 047124                                          TYPE    MSG076          ;BANK  ERRORS
7903 047130                                          SET SUCCESS
7904 047136                                        END ;OF IF SUCCESS
7905 047136                                        TYPOCS    BANK,3
7906 047146  116300  002626                      MOVB      CONFIG+2(R3),R0
7907 047152  042700  177400                      BIC       #^C377,R0
7908 047156                                      TYPDEC    R0
7909 047162                                      TYPE      $CRLF
7910 047166                                    END ;OF IFB CONFIG(R3)
7911 047166                                  END ;OF FOR BANK
7912 047202                                END ;OF IF $ERTTL
7913 047202                                POP     BANK,R3,R2,R0
7914 047214  000207                        RETURN
```

```
7917 047216                                FCMD10: SUBTST  <<FS    COMMAND 10      REFRESH TEST>>
                                           ;*****************************************************************************
                                           ;*SUBTEST       FS      COMMAND 10      REFRESH TEST
                                           ;*****************************************************************************
7918 047216                                        PUSH    BANK,PATTERN,TKVEC,TKVEC+2,NOPAR
7919 047242  010637  002266                   MOV     SP,FSSTACK                  ;SAVE LAST GOOD STACK POINTER
7920 047246                                        TYPE    MSG073                  ;REFRESH TEST
7921
7922 047252                                        IF #SW0 SET.IN @SWR
7923 047262  104470                                  ECCDIS                        ;DISABLE ERROR CORRECTION
7924 047264                                        ELSE
7925 047266  104502                                  CLRCSR                        ;CLEAR CSRS
7926 047270                                        END ;OF IF
7927 047270                                        TYPE    MSG056                  ;BELL = EACH PASS COMPLETE
7928
7929 047274                                        TYPE    MSG046                  ;TO ESCAPE TYPE ANY KEY!
7930 047300  012737  047424  000060              MOV     #CMD10C,TKVEC
7931 047306  012737  000340  000062              MOV     #340,TKVEC+2
7932 047314  017700  133264                      MOV     @$TKB,R0                ;KILL ANY OLD INTERRUPT
7933 047320  042737  000200  177776              BIC     #BIT7,PSW               ;LOWER CPU PRIORITY TO 140
7934 047326  052777  000100  133246              BIS     #BIT6,@$TKS             ;ENABLE KEYBOARD INTERRUPTS
7935
7936 047334                                        SET     HEADER,MUT
7937
7938 047350                                CMD10B: FOR BANK := #0 TO LASTBANK
7939 047354  004737  042700                         CALL EXBANK
7940 047360                                          IF ACFLAG IS TRUE AND RRFLAG IS FALSE
7941 047374  104511                                      INVALIDATE
7942 047376  004737  022052                           CALL MT0022
7943 047402                                          END ;OF IF ACFLAG
7944 047402                                        END ;OF FOR BANK
7945 047416                                        TYPE    $BELL                   ;RING BELL
7946 047422                                        GOTO    CMD10B
7947
7948 047424  013706  002266              CMD10C: MOV     FSSTACK,SP              ;RECOVER OLD STACK POINTER
7949 047430  042777  000100  133144              BIC     #BIT6,@$TKS
7950 047436  117700  133142                      MOVB    @$TKB,R0                ;READ CHAR TO KILL FLAG
7951 047442                                        POP     NOPAR,TKVEC+2,TKVEC,PATTERN,BANK
7952 047466                                        MAP     BANK                    ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
7953 047502  004737  042700                      CALL    EXBANK
7954 047506  000207                              RETURN
7955
```

```
                                FCMD11: SUBTST  <<FS    COMMAND 11      SET FILL COUNT>>
                                ;*********************************************************************************
                                ;*SUBTEST        FS      COMMAND 11      SET FILL COUNT
                                ;*********************************************************************************
7959 047510                             PUSH    R0
7960 047512                             TYPE    MSG085                  ;FILL COUNT(OCTAL)?
7961 047516  104413                     RDOCT
7962 047520                             POP     R0
7963 047522  042700  177760             BIC     #^C17,R0
7964 047526  110037  002327             MOVB    R0,$FILLS
7965 047532                             POP     R0
7966 047534  000207                     RETURN
7967
                                FCMD12: SUBTST  <<FS    COMMAND 12      ENTER KAMIKAZE MODE>>
                                ;*********************************************************************************
                                ;*SUBTEST        FS      COMMAND 12      ENTER KAMIKAZE MODE
                                ;*********************************************************************************
7969 047536                             TYPE    MSG101          ;ENTERING KAMIKAZE MODE
7970 047542                             SET     KAMIKAZE,SKIPKAMI
7971 047556  000207                     RETURN
7972
                                FCMD13: SUBTST  <<FS    COMMAND 13      EXIT KAMIKAZE MODE>>
                                ;*********************************************************************************
                                ;*SUBTEST        FS      COMMAND 13      EXIT KAMIKAZE MODE
                                ;*********************************************************************************
7974 047560                             TYPE    MSG102          ;LEAVING KAMIKAZE MODE
7975 047564  005037  002004             CLR     KAMIKAZE
7976 047570                             SET     SKIPKAMI
7977 047576  000207                     RETURN
7978
                                FCMD14: SUBTST  <<FS    COMMAND 14      TURN CACHE OFF>>
                                ;*********************************************************************************
                                ;*SUBTEST        FS      COMMAND 14      TURN CACHE OFF
                                ;*********************************************************************************
7980 047600                             TYPE    MSG106                  ;CACHE IS OFF
7981 047604  104424                     CACHOFF                         ;TURN CACHE OFF
7982 047606  013737  002514  002516     MOV     CACHKN,CACHKN+2         ;SAVE OLD CACHE ON STATE
7983 047614  005037  002514             CLR     CACHKN                  ;KEEP CACHE OFF
7984 047620  000207                     RETURN
7985
                                FCMD15: SUBTST  <<FS    COMMAND 15      TURN CACHE ON>>
                                ;*********************************************************************************
                                ;*SUBTEST        FS      COMMAND 15      TURN CACHE ON
                                ;*********************************************************************************
7987 047622                             TYPE    MSG107                  ;CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)
7988 047626  013737  002516  002514     MOV     CACHKN+2,CACHKN         ;RESTORE OLD CACHE ON STATE
7989 047634  104423                     CACHON                          ;TURN CACHE ON
7990 047636  000207                     RETURN
7991
```

```
8004
8005 047640                          FCMD16: SUBTST  <<FS    COMMAND 16      TEST ONLY SELECTED BANKS>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       FS      COMMAND 16      TEST ONLY SELECTED BANKS
                                     ;*****************************************************************************
8006 047640                                  TYPE    MSG105          ;ENTER BANKS IN OCTAL - USE NUMBER OUTSIDE RANGE TO TERMINAT
8007 047644  004737  047734                  CALL    CMD17A                  ;ERASE OLD SELECTIONS
8008 047650                                  BEGIN CMD16LOOP
8009 047650                                    REPEAT
8010 047650                                      TYPE        MSG030          ;BANK(0-177)?
8011 047654  104413                              RDOCT                       ;READ AN OCTAL NUMBER ONTO THE STACK
8012 047656                                      POP R1                      ;PUT IT IN R1
8013 047660                                      IF R1 GT #177 OR R1 LT #0
8014 047672                                        LEAVE CMD16LOOP
8015 047674                                      END ;OF IF R1
8016 047674  006301                              ASL  R1
8017 047676  006301                              ASL  R1                     ;R1 <- R1 * 4
8018 047700  052761  040000  002626              BIS #BIT14,CONFIG+2(R1)
8019 047706                                    END ;OF REPEAT
8020 047710                                  END CMD16LOOP
8021 047710                                  TYPE    MSG110                  ;ONLY SELECTED BANKS WILL BE TESTED
8022 047714                                  SET     SELONLY
8023 047722  000207                        RETURN
8024
8025 047724                          FCMD17: SUBTST  <<FS    COMMAND 17      RESUME TESTING ALL BANKS>>
                                     ;*****************************************************************************
                                     ;*SUBTEST       FS      COMMAND 17      RESUME TESTING ALL BANKS
                                     ;*****************************************************************************
8026 047724                                  TYPE    MSG111                  ;ALL BANKS WILL BE TESTED
8027 047730  005037  002000                  CLR     SELONLY
8028
8029                                         ;ENTRY POINT FROM CMD16
8030 047734  013702  002526          CMD17A: MOV     LASTBANK,R2
8031 047740  006302                          ASL     R2
8032 047742  006302                          ASL     R2
8033 047744                                  FOR R1 := #0 TO R2 BY #4
8034 047746  042761  040000  002626            BIC   #BIT14,CONFIG+2(R1)
8035 047754                                  END ;OF FOR R1
8036 047764  000207                        RETURN
```

```
 8039 047766                        WHICHCSR:SUBTST <<SUBR  DETERMINE CORRECT CSR>>
                                    ;*****************************************************************************
                                    ;*SUBTEST       SUBR    DETERMINE CORRECT CSR
                                    ;*****************************************************************************
 8040 047766  013700  002216               MOV     TOTCSRS,R0      ;GET CSR'S FLAG
 8041 047772  022700  100000               CMP     #BIT15,R0       ;CSR 0?
 8042 047776  001003                       BNE     1$              ;NO - SKIP
 8043 050000  005037  002146               CLR     CSRNO           ;YES - SET IT UP
 8044 050004  000207                       RETURN
 8045
 8046 050006                        1$:     TYPE    MSG022          ;WHICH CSR(0-F)
 8047 050012  104412                       RDLIN                   ;GET CHARACTER
 8048 050014                               POP     R0              ;PUT IN R0
 8049 050016  011000                       MOV     (R0),R0         ;PUT CHAR IN R0
 8050 050020  020027  000106               CMP     R0,#106         ;CHECK LIMIT
 8051 050024  101370                       BHI     1$              ;IF BAD LOOP TILL HE TYPES IT RIGHT
 8052 050026  022700  000101               CMP     #'A,R0
 8053 050032  103002                       BHIS    2$
 8054 050034  162700  000007               SUB     #7,R0
 8055 050040  162700  000060        2$:     SUB     #60,R0
 8056 050044  006300                       ASL     R0
 8057 050046  010037  002146               MOV     R0,CSRNO
 8058 050052  000207                       RETURN
```

```
                                        .SBTTL  ERROR DATA (SUPERVISOR) SETUP STUFF
050054                          $PER25: LET ADDRESS := R1 - #2
050066                                  IF ABORTFLAG IS FALSE
050074                                    TESTAREA              ;ENTER TEST MODE
050102                                    LET BAD := -2(R1)
050110  104417                            KERNEL                ;ENTER KERNEL MODE
050112                                  END ;OF IF ABORTFLAG
050112                                  IF 177654 EQ #0
050120                                    LET GOOD := R2
050124                                  ELSE
050126                                    LET GOOD := R3
050132                                  END ;OF IF
050132  000137  053230                  JMP     PERRAW

050136                          PERRA3: SUBTST  <<DATA WAS 3 WORDS>>
                                ;*****************************************************************************
                                ;*SUBTEST        DATA WAS 3 WORDS
                                ;*****************************************************************************
050136                                  IF BADPC EQ #0 THEN $CALL BADSTACK
050150                                  PUSH    R0
050152  005037  002144                  CLR     CSR             ;MAKE SURE CSR BIT HOLDER IS CLEAR
050156  104505                          CHK1DIS                 ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
050160                                  TESTAREA
050166  005711                          TST     (R1)            ;READ LOCATION TO READ CHECKBITS INTO CSR
050170  104417                          KERNEL
050172  104426                          READCSR                 ;GET CSR CONTENTS
050174  013700  002144                  MOV     CSR,R0          ;SAVE CSR CONTENTS IN R0
050200  104503                          CLR1CSR                 ;RETURN CSR TO NORMAL MODE
050202  072027  177773                  ASH     #-5,R0          ;MOVE CHECK BITS TO BOTTOM OF WORD
050206  042700  177600                  BIC     #^C177,R0       ;CLEAR OFF EXTRANEOUS GARBAGE
050212                                  LET ADDRESS := R1        ;SAVE VIRTUAL ADDRESS FOR PRINTOUT
050216  005037  002042                  CLR     GOOD            ;FIRST TEST WORD WRITTEN SHOULD ALWAYS BE ZERO
050222                                  TESTAREA                ;ENTER TEST MODE
050230  011137  002050                  MOV     (R1),BAD        ;GET BAD DATA FROM MUT - FIRST WORD
050234  011437  002052                  MOV     (R4),BAD2       ;AND SECOND WORD
050240  104417                          KERNEL                  ;ENTER KERNEL MODE
050242  110037  002054                  MOVB    R0,BAD3         ;MOVE BAD CHECKBITS FOR PRINTOUT
050246  105037  002055                  CLRB    BAD3+1          ;CLEAR OFF THE OTHER UNUSED BITS
050252  004737  053464                  CALL    PERBNK          ;MARK BANK AS BAD IN CONFIG TABLE
050256  104033                          ERROR   +33
050260                                  POP     R0              ;RESTORE R0
050262                                  IF #SW0 SET.IN @SWR
050272  104506                            ENASBE                ;TRAP ON SINGLE BIT ERRORS
050274                                  ELSE
050276  104472                            ECCINIT               ;TRAP ON UNCORRECTABLE ERRORS
050300                                  END; OF IF #SW0
050300  000002                          RTI
```

```
 8653 050302                        $PER30: LET GOOD := R1
 8654 050306                                LET ADDRESS := (SP) - 16
 8655 050320                                IF ABORTFLAG IS FALSE
 8656 050326                                  TESTAREA                ;ENTER TEST MODE
 8657 050334                                  LET BAD := @ADDRESS
 8658 050342  104417                          KERNEL                  ;ENTER KERNEL MODE
 8659 050344                                END ;OF IF ABORTFLAG
 8660 050344  000137  053230                JMP PERRAW
 8661
 8662 050350                        GETDATA:SUBTST  <<GET DATA FROM ABORTED AREA IF POSSIBLE>>
                                    ;*****************************************************************************
                                    ;*SUBTEST       GET DATA FROM ABORTED AREA IF POSSIBLE
                                    ;*****************************************************************************
 8663 050350                                PUSH    R0,4,114
 8664 050362  010637  050446                MOV     SP,GETDA1
 8665 050366  012737  050426  000004        MOV     #1$,4
 8666 050374  012737  050426  000114        MOV     #1$,114
 8667 050402  013700  002032                MOV     ADDRESS,R0
 8668 050406                                TESTAREA
 8669 050414  011037  002050                MOV     (R0),BAD
 8670 050420  104417                        KERNEL
 8671 050422  005037  002140                CLR     ABORTFLAG
 8672 050426  013706  050446        1$:     MOV     GETDA1,SP               ;RESTORE KNOWN GOOD STACK POINTER
 8673 050432                                POP     114,4,R0
 8674 050444  000207                        RETURN
 8675 050446  000000                GETDA1: 0
```

```
                                        .SBTTL  POWER FAIL AUTO RESTART
                                        .SBTTL  ROUTINE POWER DOWN AND UP
                                ;*****************************************************************************
                                ;POWER DOWN ROUTINE
050450                          $PWRDN:
                                        ;SAVE CACHE STATUS
050450  005737  002514                  TST     CACHKN
050454  001403                          BEQ     5$
050456                                  PUSH    CONTRL
050462  104423                          CACHON                          ;TURN CACHE ON
050464  012737  051416  000024  5$:     MOV     #$ILLUP,PWRVEC ;;SET FOR FAST UP
050472  012737  000340  000026          MOV     #340,PWRVEC+2   ;;PRIO:7
050500                                  PUSH    R0,R1,R2,R3,R4,R5,CSRNO
                                        ;SAVE USER PAR'S & PDR7
050520  012700  177700                  MOV     #177700,R0
050524  012701  000021                  MOV     #17.,R1
050530                          1$:     PUSH    -(R0)
050532  077102                          SOB     R1,1$
                                        ;SAVE SUPERVISOR PAR'S
050534  005737  002426                  TST     NOSUPER
050540  001013                          BNE     PD1
050542  012700  172300                  MOV     #172300,R0
050546  012701  000020                  MOV     #16.,R1
050552                          2$:     PUSH    -(R0)
050554  077102                          SOB     R1,2$
050556                                  IF RLFLAG IS TRUE THEN $CALL WOOPS
                                        ;COPY KERNEL MAP TO USER & SUPERVISOR
050570  012700  172300          PD1:    MOV     #KIPDR0,R0
050574  012701  177600                  MOV     #UIPDR0,R1
050600  012702  172200                  MOV     #SIPDR0,R2
050604  012703  000040                  MOV     #32.,R3
050610  011021                  3$:     MOV     (R0),(R1)+
050612  012022                          MOV     (R0)+,(R2)+
050614  077303                          SOB     R3,3$
```

```
 8720                                        ;SAVE USER & SUPERVISOR STACK POINTERS
 8721 050616                                 USER
 8722 050624  010600                         MOV     USP,R0
 8723 050626  104417                         KERNEL                  ;ENTER KERNEL MODE
 8724 050630                                 PUSH    R0
 8725 050632  005737  002426                 TST     NOSUPER
 8726 050636  001006                         BNE     7$
 8727 050640                                 SUPERVISOR              ;ENTER SUPERVISOR MODE
 8728 050646  010600                         MOV     SSP,R0
 8729 050650  104417                         KERNEL                  ;ENTER KERNEL MODE
 8730 050652                                 PUSH    R0
 8731                                        ;SAVE ECC REGISTERS
 8732 050654  013701  002216          7$:    MOV     TOTCSRS,R1      ;GET CSR'S
 8733 050660                                 BEGIN   LCSRSAVE
 8734 050660                                   FOR CSRNO := #0 TO #36 BY #2
 8735 050664  006301                             ASL         R1
 8736 050666                                     ON.ERROR
 8737 050670  104426                               READCSR
 8738 050672                                       PUSH      CSR
 8739 050676                                     END ;OF ON.ERROR
 8740 050676                                   IF R1 EQ #0 THEN LEAVE LCSRSAVE
 8741 050702                                   END ;OF FOR CSRNO
 8742 050720                                 END LCSRSAVE
 8743                                        ;SAVE MMR0,1,2,3
 8744 050720                                 PUSH    MMR0,MMR1,MMR2
 8745 050734  005737  002426                 TST     NOSUPER
 8746 050740  001002                         BNE     8$
 8747 050742                                 PUSH    MMR3
 8748                                        ;SAVE KERNEL PAR'S
 8749 050746  012700  172400          8$:    MOV     #172400,R0
 8750 050752  012701  000020                 MOV     #16.,R1
 8751 050756                          4$:    PUSH    -(R0)
 8752 050760  077102                         SOB     R1,4$
 8753                                        ;SAVE UNIBUS MAP REGISTERS
 8754 050762  005737  002424                 TST     NO22BIT
 8755 050766  001004                         BNE     9$
 8756 050770                                 PUSH    MAPH0,MAPL0
 8757                                        ;SAVE POSSIBLE SOFTWARE SWITCH REGISTER
 8758 051000                          9$:    PUSH    @SWR
 8759                                        ;SAVE STACK POINTER
 8760 051004  010637  051422                 MOV     SP,$SAVR6       ;;SAVE SP
 8761                                        ;NOW SET UP REAL VECTOR
 8762 051010  012737  051022  000024         MOV     #$PWRUP,PWRVEC  ;;SET UP VECTOR
 8763 051016  000000                  $DOWN: HALT
 8764 051020  000776                         BR      $DOWN           ;;HANG UP
```

```
                                        ;*****************************************************************************
                                        ;POWER UP ROUTINE
051022                                  $PWRUP:
051022  012737  051416  000024                  MOV     #$ILLUP,PWRVEC ;;SET FOR FAST DOWN
                                                ;RESTORE STACK POINTER
051030  013706  051422                          MOV     $SAVR6,SP       ;;GET SP
051034  005037  051422                          CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
051040  005237  051422          1$:             INC     $SAVR6          ;;WAIT FOR THE INC
051044  001375                                  BNE     1$              ;;OF A WORD
                                                ;RESTORE POSSIBLE SOFTWARE SWITCH REGISTER
051046                                          POP     @SWR
                                                ;RESTORE UNIBUS MAP
051052  005737  002424                          TST     NO22BIT
051056  001006                                  BNE     10$
051060                                          POP     MAPLO,MAPHO
051070  004737  042304                          CALL    LOWMAP          ;SETUP LOWER 16K OF UNIBUS MAP
                                                ;RESTORE KERNEL PAR'S & PDR'S
051074  012700  172340          10$:            MOV     #172340,R0
051100  012702  172300                          MOV     #KIPDR0,R2
051104  012701  000020                          MOV     #16.,R1
051110                          6$:             POP     (R0)+
051112  012722  077406                          MOV     #77406,(R2)+
051116  077104                                  SOB     R1,6$
                                                ;RESTORE MMR3,2,1,0
051120  005737  002426                          TST     NOSUPER
051124  001002                                  BNE     11$
051126                                          POP     MMR3
051132                          11$:            POP     MMR2,MMR1,MMR0
                                                ;RESTORE ECC REGISTERS
051146  013701  002216                          MOV     TOTCSRS,R1      ;GET CSR'S
051152  042701  177400                          BIC     #177400,R1
051156                                          BEGIN LCSRRESTORE
051156                                            FOR CSRNO := #36 DOWNTO #0 BY #2
051164  006201                                      ASR         R1
051166                                            ON.ERROR
051170                                              POP       CSR
051174  104425                                      LOADCSR
051176                                            END ;OF ON.ERROR
051176                                          IF R1 EQ #0 THEN LEAVE LCSRRESTORE
051202                                          END ;OF FOR CSRNO
051220                                          END LCSRRESTORE
                                                ;COPY KERNEL MAP TO USER & SUPERVISOR
051220  012700  172300                          MOV     #KIPDR0,R0
051224  012701  177600                          MOV     #UIPDR0,R1
051230  012702  172200                          MOV     #SIPDR0,R2
051234  012703  000040                          MOV     #32.,R3
051240  011021                  3$:             MOV     (R0),(R1)+
051242  012022                                  MOV     (R0)+,(R2)+
051244  077303                                  SOB     R3,3$
```

```
8820                                        ;RESTORE SUPERVISOR & USER STACK POINTERS
8821 051246  005737  002426                 TST     NOSUPER
8822 051252  001006                         BNE     13$
8823 051254                                 POP     R0
8824 051256                                 SUPERVISOR              ;ENTER SUPERVISOR MODE
8825 051264  010006                         MOV     R0,SSP
8826 051266  104417                         KERNEL                  ;ENTER KERNEL MODE
8827 051270                         13$:    POP     R0
8828 051272                                 USER
8829 051300  010006                         MOV     R0,USP
8830 051302  104417                         KERNEL                  ;ENTER KERNEL MODE
8831                                        ;RESTORE SUPERVISOR PAR'S
8832 051304  012700  172240                 MOV     #172240,R0
8833 051310  012701  000020                 MOV     #16.,R1
8834 051314                         7$:     POP     (R0)+
8835 051316  077102                         SOB     R1,7$
8836                                        ;RESTORE USER PAR'S & PDR7
8837 051320  012700  177636                 MOV     #177636,R0
8838 051324  012701  000021                 MOV     #17.,R1
8839 051330                         8$:     POP     (R0)+
8840 051332  077102                         SOB     R1,8$
8841                                        ;RESTORE POSSIBLE SOFTWARE DISPLAY REGISTER
8842 051334  013777  002010  131236         MOV     $PATMAR,@DISPLAY
8843 051342  013737  002010  000174         MOV     $PATMAR,DISPREG
8844 051350                                 POP     CSRNO,R5,R4,R3,R2,R1,R0
8845 051370  012737  050450  000024         MOV     #$PWRDN,PWRVEC  ;;SET UP THE POWER DOWN VECTOR
8846 051376                                 TYPE    MSG051          ;REPORT THE POWER FAILURE
8847                                        ;RESTORE CACHE STATUS
8848 051402  005737  002514                 TST     CACHKN
8849 051406  001402                         BEQ     9$
8850 051410                                 POP     CONTRL
8851 051414  000002                 9$:     RTI
8852 051416  000000                 $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
8853 051420  000776                         BR      $ILLUP          ;; BEFORE THE POWER DOWN WAS COMPLETE
8854 051422  000000                 $SAVR6: 0                       ;;PUT THE SP HERE
8855                                        .EVEN
```

```
 8867 051424                                WOOPS:  SUBTST  <<POWER FAIL WHILE RELOCATED>>
                                            ;*****************************************************************************
                                            ;*SUBTEST       POWER FAIL WHILE RELOCATED
                                            ;*****************************************************************************
 8868 051424                                        PUSH    BANK
 8869 051430  005037  002100                        CLR     BANK
 8870 051434                                        MAP     BANK                    ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
 8871 051450                                        SUPERVISOR                      ;ENTER SUPERVISOR MODE
 8872 051456  013737  060024  052022                MOV     FIRST+PWRVEC,WOOPSAV
 8873 051464  013737  060026  052024                MOV     FIRST+PWRVEC+2,WOOPSAV+2
 8874 051472                                        BMOV    FIRST+WOOPUP,WOOPSAV+4,WOOPEND-WOOPUP/2+12.
 8875 051504  012737  051610  060024                MOV     #WOOPUP,FIRST+PWRVEC
 8876 051512  012737  000340  060026                MOV     #340,FIRST+PWRVEC+2
 8877 051520                                        BMOV    WOOPUP,FIRST+WOOPUP,WOOPEND-WOOPUP/2
 8878 051532  012700  172340                        MOV     #KIPAR0,R0
 8879 051536  012701  131772                        MOV     #FIRST+WOOPEND,R1
 8880 051542  012702  000010                        MOV     #8.,R2
 8881 051546  012021                        1$:     MOV     (R0)+,(R1)+
 8882 051550  077202                                SOB     R2,1$
 8883 051552  005737  002426                        TST     NOSUPER
 8884 051556  001002                                BNE     2$
 8885 051560  013721  172516                        MOV     MMR3,(R1)+
 8886 051564  013721  177576                2$:     MOV     MMR2,(R1)+
 8887 051570  013721  177574                        MOV     MMR1,(R1)+
 8888 051574  013721  177572                        MOV     MMR0,(R1)+
 8889 051600  104417                                KERNEL                          ;ENTER KERNEL MODE
 8890 051602                                        POP     BANK
 8891 051606  000207                                RETURN
```

```
 8894 051610                            WOOPUP: SUBTST  <<POWER UP FROM BANK 0 TO RELOCATION>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       POWER UP FROM BANK 0 TO RELOCATION
                                        ;*****************************************************************************
 8895 051610  012700  051772                    MOV     #WOOPEND,R0
 8896 051614  012701  172340                    MOV     #KIPAR0,R1
 8897 051620  012703  172300                    MOV     #KIPDR0,R3
 8898 051624  012702  000010                    MOV     #8.,R2
 8899 051630  012021                    1$:     MOV     (R0)+,(R1)+
 8900 051632  012723  077406                    MOV     #77406,(R3)+
 8901 051636  077204                            SOB     R2,1$
 8902 051640  005737  002426                    TST     NOSUPER
 8903 051644  001002                            BNE     3$
 8904 051646  012037  172516                    MOV     (R0)+,MMR3
 8905 051652  012037  177576            3$:     MOV     (R0)+,MMR2
 8906 051656  012037  177574                    MOV     (R0)+,MMR1
 8907 051662  012037  177572                    MOV     (R0)+,MMR0
 8908 051666  013706  051422                    MOV     $SAVR6,SP
 8909 051672                                    PUSH    BANK
 8910 051676  005037  002100                    CLR     BANK
 8911 051702                                    MAP     BANK                    ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
 8912 051716                                    SUPERVISOR              ;ENTER SUPERVISOR MODE
 8913 051724  013737  052022  060024           MOV     WOOPSAV,FIRST+PWRVEC
 8914 051732  013737  052024  060026           MOV     WOOPSAV+2,FIRST+PWRVEC+2
 8915                                           ;SIMULATE THE FOLLOWING BLOCK MOV BUT WITH NO STACK ACCESSES
 8916                                           ;BMOV   WOOPSAV+4,FIRST+WOOPUP,WOOPEND-WOOPUP/2+12.
 8917 051740  012700  052026                    MOV     #WOOPSAV+4,R0
 8918 051744  012701  000105                    MOV     #WOOPEND-WOOPUP/2+12.,R1
 8919 051750  012702  131610                    MOV     #FIRST+WOOPUP,R2
 8920 051754  012022                    2$:     MOV     (R0)+,(R2)+
 8921 051756  077102                            SOB     R1,2$
 8922
 8923 051760  104417                            KERNEL                          ;ENTER KERNEL MODE
 8924 051762                                    POP     BANK
 8925 051766  000137  051022                    JMP     $PWRUP
 8926 051772  000014                WOOPEND:.REPT   12.
 8929 052022  000107                WOOPSAV:.REPT   WOOPEND-WOOPUP/2+12.*2
```

```
                                        .SBTTL  IO SUBROUTINES
                                        .SBTTL  ROUTINE TYPE
                                ;*******************************************************************************
                                ;*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
                                ;*
052240  105737  002330          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
052244  100407                          BMI     6$              ;BR IF NO
052246  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
052250  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
052254  112046                  4$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
052256  001005                          BNE     7$              ;;BR IF IT ISN'T THE TERMINATOR
052260  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
052262  012600                  5$:     MOV     (SP)+,R0        ;;RESTORE R0
052264  062716  000002          6$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
052270  000002                          RTI                     ;;RETURN
052272  122716  000011          7$:     CMPB    #HT,(SP)        ;BRANCH IF NOT <HT>
052276  001002                          BNE     11$
052300  112716  000040                  MOVB    #' ,(SP)        ;REPLACE TAB WITH SPACE
052304  122716  000200          11$:    CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
052310  001006                          BNE     8$
052312  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
052314                                  TYPE                    ;;TYPE A CR AND LF
052316  002620                          $CRLF
052320  105037  052452                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
052324  000753                          BR      4$              ;;GET NEXT CHARACTER
052326  004737  052364          8$:     CALL    $TYPEC          ;;GO TYPE THIS CHARACTER
052332  123726  002612          9$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
052336  001346                          BNE     4$              ;;IF NO GO GET NEXT CHAR.
052340  013746  002326                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
052344  105366  000001          10$:    DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
052350  002770                          BLT     9$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
052352  004737  052364                  CALL    $TYPEC          ;;GO TYPE A NULL
052356  105337  052452                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
052362  000770                          BR      10$             ;;LOOP
052364                          $TYPEC: PUSH    R1
052366  116601  000004                  MOVB    4(SP),R1
052372  005737  002514                  TST     CACHKN
052376  001402                          BEQ     2$
052400                                  PUSH    CONTRL
052404                          2$:     PUSH    R0
052406  104424                          CACHOFF                 ;TURN CACHE OFF
052410  105777  130172          3$:       TSTB  @$TPS           ;;WAIT UNTIL PRINTER IS READY
```

```
 9015 052414  100375                          BPL   3$
 9016 052416  110177  130166                  MOVB  R1,@$TPB        ;;LOAD CHAR TO BE TYPED INTO DATA REG.
 9020 052422  122766  000015  000002          CMPB  #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
 9021 052430  001003                          BNE   1$              ;;BRANCH IF NO
 9022 052432  105037  052452                  CLRB  $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
 9023 052436  000406                          BR    $TYPEX          ;;EXIT
 9024 052440  122766  000012  000002  1$:     CMPB  #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
 9025 052446  001402                          BEQ   $TYPEX          ;;BRANCH IF YES
 9026 052450  105227                          INCB  (PC)+           ;;COUNT THE CHARACTER
 9027 052452  000000                $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
 9028 052454                        $TYPEX: POP   R0
 9029 052456  005737  002514                  TST   CACHKN          ;IS THERE A CACHE?
 9030 052462  001402                          BEQ   2$              ;BRANCH IF NOT
 9031 052464                                  POP   CONTRL          ;POP CACHE STATUS
 9032 052470                        2$:     POP   R1
 9033 052472  000207                          RETURN
 9034 052474                        SUPLIMIT:;....!.!!.!!.THIS IS THE LIMIT ON SUPERVISOR MAPPED TO MUT SPACE
```

```
                        .SBTTL  ERROR DATA SETUP
;
;USE THIS       IF THIS CONDITION DISCRIBES THE ERROR
;
;PERR01         TRAP
;               BAD DATA IN R0 UNLESS ABORTED
;                       THEN BAD DATA IS POINTED TO BY -(R4)
;               GOOD DATA IN R5
;
;PERR02         TRAP
;               BAD DATA IN R1 UNLESS ABORTED
;                       THEN BAD DATA IS POINTED TO BY -(R4)
;               GOOD DATA IN R2
;
;PERR03         TRAP
;               BAD DATA IS POINTED TO BY -(R1)
;               GOOD DATA IN R4
;
;PERR04         TRAP
;               BAD DATA IN R4 UNLESS ABORTED
;                       THEN BAD DATA IS POINTED TO BY -2(R0)
;               GOOD DATA IN R2
;
;PERR05         JSR     PC
;               BAD DATA IS POINTED TO BY -(R0)
;               GOOD DATA IN R2
;               RETURN AFTER SETTING UP GOOD,BAD,ADDRESS
;
;PERR06         JSR     PC
;               BAD DATA IS POINTED TO BY -(R0)
;               GOOD DATA IS ZERO
;               RETURN AFTER SETTING UP GOOD,BAD,ADDRESS
;
;PERR07         TRAP
;               BAD DATA IN R2 UNLESS ABORTED
;                       THEN BAD DATA IS POINTED TO BY (R1)
;               GOOD DATA IN DATBUF
;
;PERR10         TRAP
;               BAD DATA IN R2 UNLESS ABORTED
;                       THEN BAD DATA IS POINTED TO BY 2(R1)
;               GOOD DATA IN DATBUF+2
;
;PERR11         TRAP
;               BYTE TEST
;               BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
;                       THEN BAD DATA IS POINTED TO BY (R1)
;               GOOD DATA IS A ZERO BYTE
;
;PERR12         TRAP
;               BYTE TEST
;               BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
;                       THEN BAD DATA IS POINTED TO BY (R1)
;               GOOD DATA IS A BYTE OF ONES
;
;PERR13         TRAP
;               BAD DATA IN R0 UNLESS ABORTED
```

```
                        ;                       THEN BAD DATA IS POINTED TO BY (R1)
                        ;               GOOD DATA IS ZERO
                        ;
                        ; PERR14       TRAP
                        ;               BAD DATA IN R0 UNLESS ABORTED
                        ;                       THEN BAD DATA IS POINTED TO BY (R1)
                        ;               GOOD DATA IS ONES
                        ;
                        ; PERR15       TRAP
                        ;               BAD DATA IN R0 UNLESS ABORTED
                        ;                       THEN BAD DATA IS POINTED TO BY (R1)
                        ;               GOOD DATA IN TSTDAT
                        ;
                        ; PERR16       TRAP
                        ;               BAD DATA IN R0 UNLESS ABORTED
                        ;                       THEN BAD DATA IS POINTED TO BY (R1)
                        ;               GOOD DATA IN TSTDAT+2
                        ;
                        ; PERR17       TRAP
                        ;               BAD DATA IN R0 UNLESS ABORTED
                        ;                       THEN BAD DATA IS POINTED TO BY (R1)
                        ;               GOOD DATA IN R2
                        ;
                        ; PERR20       TRAP
                        ;               BAD DATA IN R0 UNLESS ABORTED
                        ;                       THEN BAD DATA IS POINTED TO BY (R1)
                        ;               GOOD DATA IN R3
                        ;
                        ; PERR21       TRAP
                        ;               7 BIT BYTE TEST
                        ;               BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
                        ;                       THEN BAD DATA IS POINTED TO BY (R1)
                        ;               GOOD DATA IS A 7 BIT BYTE ON ONES
                        ;
                        ; PERR22       TRAP
                        ;               BAD DATA IN R2 UNLESS ABORTED
                        ;                       THEN BAD DATA IS POINTED TO BY (R1)
                        ;               GOOD DATA IN R0
                        ;
                        ; PERR23       TRAP
                        ;               BAD DATA IN R0 UNLESS ABORTED
                        ;                       THEN BAD DATA IS POINTED TO BY (R1)
                        ;               GOOD DATA IN R4
                        ;
                        ; PERR24       TRAP
                        ;               BAD DATA IN R0 UNLESS ABORTED
                        ;                       THEN BAD DATA IS POINTED TO BY (R2)
                        ;               GOOD DATA IN R3
                        ;
                        ; PERR25       TRAP
                        ;               BAD DATA POINTED TO BY -(R1)
                        ;               GOOD DATA IN R2 UNLESS LOC V177654 IS SET
                        ;                       THEN GOOD DATA IS IN R3
                        ;
                        ; PERR26       TRAP
                        ;               BAD DATA IS DOUBLE WORD POINTED TO BY R1 AND IN LOW 7 BITS OF R0
                        ;               GOOD DATA IS 000000,,100000,,100
```

```
9826
9827                    PERR27      TRAP
9828                                BAD DATA IS DOUBLE WORD POINTED TO BY R1 AND IN LOW 7 BITS OF R0
9829                                GOOD DATA IS 000000,,000000,,077
9830
9831                    PERR30      TRAP
9832                                BAD DATA IS POINTED TO BY -16(SP)
9833                                GOOD DATA IS IN R1
9834
9835                    PERR31      TRAP
9836                                SPECIAL ECC FAILURE HANDLER
9837
9838                    PERR32      TRAP
9839                                SPECIAL ECC FAILURE HANDLER
9840
9841                    PERR33      TRAP
9842                                SPECIAL ECC FAILURE HANDLER
9843
9844                    PERR34      TRAP
9845                                SPECIAL ECC FAILURE HANDLER
9846
9847                    PERR35      TRAP
9848                                SPECIAL BRANCH GOBBLE FAILURE HANDLER
9849
9850                        CALLING SEQUENCE FOR TRAP TYPES
9851                        BEQ     2$              ;NO - ERROR,BRANCH FOR CARD
9852                        PERRXX                  ;TRAP TO ERROR ROUTINE
9853                    2$: NEXT    INSTRUCTION     ;CONTINUE TESTING
```

```
 9856 052474  010437  002032          $PER01: MOV     R4,ADDRESS
 9857 052500  162737  000002  002032          SUB     #2,ADDRESS
 9858 052506  010037  002050                  MOV     R0,BAD
 9859 052512  010537  002042                  MOV     R5,GOOD
 9860 052516  000137  053230                  JMP PERRAW
 9861
 9862 052522  010437  002032          $PER02: MOV     R4,ADDRESS
 9863 052526  162737  000002  002032          SUB     #2,ADDRESS
 9864 052534  010137  002050                  MOV     R1,BAD
 9865 052540  010237  002042                  MOV     R2,GOOD
 9866 052544  000137  053230                  JMP PERRAW
 9867
 9868 052550  010137  002032          $PER03: MOV     R1,ADDRESS
 9869 052554  162737  000002  002032          SUB     #2,ADDRESS
 9870 052562  010437  002042                  MOV     R4,GOOD
 9871 052566  016137  177776  002050          MOV     -2(R1),BAD
 9872 052574  000137  053230                  JMP PERRAW
 9873
 9874 052600  010037  002032          $PER04: MOV     R0,ADDRESS
 9875 052604  162737  000002  002032          SUB     #2,ADDRESS
 9876 052612  010437  002050                  MOV     R4,BAD
 9877 052616  010237  002042                  MOV     R2,GOOD
 9878 052622  000137  053230                  JMP     PERRAW
 9879
 9880 052626  010237  002042          PERR05: MOV     R2,GOOD
 9881 052632  014037  002050          PERA05: MOV     -(R0),BAD
 9882 052636  010037  002032                  MOV     R0,ADDRESS
 9883 052642  062700  000002                  ADD     #2,R0           ;RESTORE R0
 9884 052646  004737  036626                  CALL    BADSTACK
 9885 052652  000207                          RETURN
 9886
 9887 052654  005037  002042          PERR06: CLR     GOOD
 9888 052660  000764                          BR      PERA05
 9889
 9890 052662  010137  002032          $PER07: MOV     R1,ADDRESS
 9891 052666  010237  002050                  MOV     R2,BAD
 9892 052672  013737  002234  002042          MOV     DATBUF,GOOD
 9893 052700  000137  053230                  JMP     PERRAW
 9894
 9895 052704                          $PER10: LET ADDRESS := R1 + #2
 9896 052716                                  LET BAD := R2
 9897 052722                                  LET GOOD := DATBUF+2
 9898 052730  000137  053230                  JMP     PERRAW
 9899
 9900 052734                          $PER11: LET ADDRESS := R1
 9901 052740                                  LET BAD := R0
 9902 052744                                  LET GOOD := #0
 9903 052750  000137  053302                  JMP     PERRAB
 9904
 9905 052754                          $PER12: LET ADDRESS := R1
 9906 052760                                  LET BAD := R0
 9907 052764                                  LET GOOD := #377
 9908 052772  000137  053302                  JMP     PERRAB
```

```
 9911 052776                        $PER13: LET ADDRESS := R1
 9912 053002                                LET BAD := R0
 9913 053006                                LET GOOD := #0
 9914 053012  000137  053230               JMP     PERRAW
 9915
 9916 053016                        $PER14: LET ADDRESS := R1
 9917 053022                                LET BAD := R0
 9918 053026                                LET GOOD := ONES
 9919 053034  000137  053230               JMP     PERRAW
 9920
 9921 053040                        $PER15: LET ADDRESS := R1
 9922 053044                                LET BAD := R0
 9923 053050                                LET GOOD := TSTDAT
 9924 053056  000137  053230               JMP     PERRAW
 9925
 9926 053062                        $PER16: LET ADDRESS := R1
 9927 053066                                LET BAD := R0
 9928 053072                                LET GOOD := TSTDAT+2
 9929 053100  000453                        BR      PERRAW
 9930
 9931 053102                        $PER17: LET ADDRESS := R1
 9932 053106                                LET BAD := R0
 9933 053112                                LET GOOD := R2
 9934 053116  000444                        BR      PERRAW
 9935
 9936 053120                        $PER20: LET ADDRESS := R1
 9937 053124                                LET BAD := R0
 9938 053130                                LET GOOD := R3
 9939 053134  000435                        BR      PERRAW
 9940
 9941 053136                        $PER21: LET ADDRESS := R1
 9942 053142                                LET BAD := R0
 9943 053146                                LET GOOD := #177
 9944 053154  000477                        BR      PERRA7
 9945
 9946 053156                        $PER22: LET ADDRESS := R1
 9947 053162                                LET BAD := R2
 9948 053166                                LET GOOD := R0
 9949 053172  000416                        BR      PERRAW
 9950
 9951 053174                        $PER23: LET ADDRESS := R1
 9952 053200                                LET BAD := R0
 9953 053204                                LET GOOD := R4
 9954 053210  000407                        BR      PERRAW
 9955
 9956 053212                        $PER24: LET ADDRESS := R2
 9957 053216                                LET BAD := R0
 9958 053222                                LET GOOD := R3
 9959 053226  000400                        BR      PERRAW
```

```
 9961 053230                            PERRAW: SUBTST  <<DATA WAS A WORD>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       DATA WAS A WORD
                                        ;*****************************************************************************
 9962 053230  004737  053464                       CALL    PERBNK
 9963 053234                                       IF ABORTFLAG IS TRUE THEN $CALL GETDATA
 9964 053246                                       IF BADPC EQ #0 THEN $CALL BADSTACK
 9965 053260  004737  053440                       CALL    PERXOR
 9966 053264                                       IF ABORTFLAG IS FALSE
 9967 053272  104011                                 ERROR +11
 9968 053274                                       ELSE
 9969 053276  104012                                 ERROR +12
 9970 053300                                       END ;OF IF ABORTFLAG
 9971 053300  000002                               RTI
 9972
 9973 053302                            PERRAB: SUBTST  <<DATA WAS A BYTE>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       DATA WAS A BYTE
                                        ;*****************************************************************************
 9974 053302  004737  053464                       CALL    PERBNK
 9975 053306                                       IF ABORTFLAG IS TRUE THEN $CALL GETDATA
 9976 053320                                       IF BADPC EQ #0 THEN $CALL BADSTACK
 9977 053332  004737  053440                       CALL    PERXOR
 9978 053336                                       IF ABORTFLAG IS FALSE
 9979 053344  104014                                 ERROR +14
 9980 053346                                       ELSE
 9981 053350  104015                                 ERROR +15
 9982 053352                                       END ;OF IF ABORTFLAG
 9983 053352  000002                               RTI
```

```
 9986 053354                                PERRA7: SUBTST  <<DATA WAS A 7 BIT BYTE>>
                                            ;*****************************************************************************
                                            ;*SUBTEST       DATA WAS A 7 BIT BYTE
                                            ;*****************************************************************************
 9987 053354                                        IF BADPC EQ #0 THEN $CALL BADSTACK
 9988 053366  004737  053440                        CALL    PERXOR
 9989 053372  004737  053464                        CALL    PERBNK
 9990 053376  104022                                ERROR   +22
 9991 053400  000002                                RTI
 9992
 9993 053402                                $PER26: LET GOOD2 := #100000
 9994 053410                                        LET GOOD3 := #100
 9995 053416  000137  050136                        JMP PERRA3
 9996
 9997 053422  005037  002044                $PER27: CLR     GOOD2
 9998 053426                                        LET GOOD3 := #077
 9999 053434  000137  050136                        JMP PERRA3
10000
10001 053440                                PERXOR: SUBTST  <<DETERMINE XOR OF GOOD & BAD>>
                                            ;*****************************************************************************
                                            ;*SUBTEST       DETERMINE XOR OF GOOD & BAD
                                            ;*****************************************************************************
10002 053440                                        PUSH    R0
10003 053442  013700  002042                        MOV     GOOD,R0
10004 053446  013737  002050  002056                MOV     BAD,BADXOR
10005 053454  074037  002056                        XOR     R0,BADXOR
10006 053460                                        POP     R0
10007 053462  000207                                RETURN
```

```
10010 053464                                PERBNK: SUBTST<<LOG ERROR ON BAD BANK>>
                                            ;*****************************************************************************
                                            ;*SUBTEST       LOG ERROR ON BAD BANK
                                            ;*****************************************************************************
10011                                               ;WHILE WE'RE HERE LET'S MARK THE BAD BANK IN THE CONFIGURATION TABLE
10012 053464                                        PUSH    R0,R1
10013 053470  013701  002100                        MOV     BANK,R1
10014 053474  006301                                ASL     R1
10015 053476  006301                                ASL     R1
10016 053500  052761  000001  002624                BIS     #BIT0,CONFIG(R1)
10017 053506  105261  002626                        INCB    CONFIG+2(R1)            ;BUMP BANK COUNTER
10018 053512  001002                                BNE     12$                     ;NO OVERFLOW - SKIP
10019 053514  105361  002626                        DECB    CONFIG+2(R1)            ;SET BACK TO 255.
10020 053520  126137  002626  002524  12$:        CMPB    CONFIG+2(R1),FRRMAX     ;IS IT PAST MAX?
10021 053526  101403                                BLOS    11$                     ;NO - SKIP
10022 053530                                        SET     TOOMANY                 ;YES
10023 053536                                11$:    POP     R1,R0
10024 053542  000207                                RETURN
10025
10026 053544  010037  002050                PERECC: MOV     R0,BAD
10027 053550                                        IF ADDRESS EQ TESTADD
10028 053560  013737  002240  002042                  MOV TSTDAT,GOOD
10029 053566                                        ELSE
10030 053570  013737  002242  002042                  MOV   TSTDAT+2,GOOD
10031 053576                                        END ;OF IF (R1)
10032 053576  004737  053440                        CALL    PERXOR
10033 053602                                        SET     HEADER
10034 053610  000207                                RETURN
10035
10036 053612                                $PER31: IF BADPC EQ #0 THEN $CALL BADSTACK
10037 053624  004737  053544                        CALL    PERECC
10038 053630                                        IF REALPAT EQ #11
10039 053640  104037                                  ERROR +37
10040 053642                                        END ;OF IF REALPAT
10041 053642                                        IF REALPAT EQ #14
10042 053652  104042                                  ERROR +42
10043 053654                                        END ;OF IF REALPAT
10044 053654                                        IF REALPAT EQ #15
10045 053664  104043                                  ERROR +43
10046 053666                                        END ;OF IF REALPAT
10047 053666                                        IF REALPAT EQ #16
10048 053676  104044                                  ERROR +44
10049 053700                                        END ;OF IF REALPAT
10050 053700                                        SET HEADER
10051 053706  000002                                RTI
```

```
10054 053710                                $PER32: IF BADPC EQ #0 THEN $CALL BADSTACK
10055 053722  010137  002032                        MOV     R1,ADDRESS
10056 053726  010037  002050                        MOV     R0,BAD
10057 053732  010237  002042                        MOV     R2,GOOD
10058 053736                                        SET     HEADER
10059 053744  104040                                ERROR   +40
10060 053746                                        SET     HEADER
10061 053754  000002                                RTI
10062
10063 053756                                $PER33: IF BADPC EQ #0 THEN $CALL BADSTACK
10064 053770  010137  002032                        MOV     R1,ADDRESS'
10065 053774  010037  002050                        MOV     R0,BAD
10066 054000  105037  002051                        CLRB    BAD+1
10067 054004  012737  000377  002042                MOV     #377,GOOD
10068 054012  004737  053440                        CALL    PERXOR
10069 054016                                        SET     HEADER
10070 054024  104041                                ERROR   +41
10071 054026                                        SET     HEADER
10072 054034  000002                                RTI
10073
10074 054036                                $PER34:IF BADPC EQ #0 THEN $CALL BADS~ACK
10075 054050                                        IF #BIT15!BIT4 OFF.IN CSR
10076 054060  104016                                  ERROR +16                     ;NO SBE OR DBE
10077 054062                                        ELSE
10078 054064  104001                                  ERROR +1                      ;EXPECTED SBE SO DBE MUST HAVE GOTTEN SET
10079 054066                                        END ;OF IF #BIT15.BIT4
10080 054066  000002                                RTI
10081
10082                                               ;DURING BRANCH GOBBLE THE CONDITION CODES WERE WRONG
10083 054070  004737  053464          $PER35: CALL    PERBNK
10084 054074  004737  036626                  CALL    BADSTACK
10085 054100  013737  002030  002050          MOV     BADPSW,BAD
10086 054106  012737  000012  002042          MOV     #12,GOOD
10087 054114  104047                          ERROR   +47
10088 054116  062706  000004                  ADD     #4,SP                         ;FIX STACK FROM TRAP
10089 054122  000207                          RETURN                                ;ABORTING TEST
10090
10091 054124  010037  002042          $PER36: MOV     R0,GOOD
10092 054130  010137  002050                  MOV     R1,BAD
10093 054134                                  SET     HEADER
10094 054142  104023                          ERROR   +23
10095 054144                                  SET     HEADER
10096 054152  000002                          RTI
```

```
                                        .SBTTL  ROUTINE SCOPE HANDLER
                                ;*******************************************************************************
                                ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                ;*AND LOAD THE DISPLAY DATA INTO THE DISPLAY REGISTER
                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                ;*SW14=1         LOOP ON TEST
                                ;*SW9=1 LOOP ON ERROR
                                ;*CALL
                                ;*      SCOPE               ;;SCOPE=IOT
054154  005237  061266          $SCOPE: INC     $DEVCT          ;TELL APT WE ARE ALIVE
054160                                  IF RESULT IS LT
054162  005037  061266                    CLR   $DEVCT
054166  105237  061270                    INCB  $UNIT
054172                                  END ;OF IF RESULT
054172  104410                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
054174                                  IF STOPOK IS TRUE AND #SW8 SET.IN @SWR
054212  005037  002370                    CLR   STOPOK
054216  000137  043534                    JMP   EXIT
054222                                  END ;OF IF STOPOK
054222                                  IF NOSCOPE IS TRUE
054230  000002                            RTI
054232                                  END ;OF IF NOSCOPE
054232                          1$:     IF #SW14 SET.IN @SWR THEN GOTO $OVER
                                ;#####START OF CODE FOR THE XOR TESTER#####
054242  000424                  $XTSTR: BR      2$              ;;IF RUNNING ON THE ''XOR'' TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A ''NOP'' (NOP=240)
054244  013746  000004                  MOV     ERRVEC,-(SP)    ;;SAVE THE CONTENTS OF THE ERROR VECTOR
054250  012737  054270  000004          MOV     #1$,ERRVEC      ;;SET FOR TIMEOUT
054256  005737  177060                  TST     177060  ;;TIME OUT ON XOR?
054262  012637  000004                  MOV     (SP)+,ERRVEC    ;;RESTORE THE ERROR VECTOR
054266  000427                          BR      $SVLAD          ;;GO TO THE NEXT TEST
054270  062706  000004          1$:     ADD     #4,SP           ;FIX STACK FROM TRAP
054274  005737  002424                  TST     NO22BIT         ;IS THIS AN 11/44?
054300  001002                          BNE     6$              ;BRANCH IF NOT
054302  005037  177766                  CLR     CPUERR          ;RESET CPU ERROR REGISTER
054306  012637  000004          6$:     MOV     (SP)+,ERRVEC    ;;RESTORE THE ERROR VECTOR
054312  000407                          BR      4$              ;;LOOP ON THE PRESENT TEST
054314                          2$:;#####END OF CODE FOR THE XOR TESTER#####
054314  105737  002012          3$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
054320  001412                          BEQ     $SVLAD          ;;BR IF NO
054322  032777  001000  126246          BIT     #SW9,@SWR       ;;LOOP ON ERROR?
054330  001404                          BEQ     5$              ;;BR IF NO
054332  013737  002564  002562  4$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
054340  000410                          BR      $OVER
054342  105037  002012          5$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
054346  011637  002562          $SVLAD: MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
054352  011637  002564                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
054356  005037  002332                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
054362  004737  054374          $OVER:  CALL    GETDIS
054366  013716  002562                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
054372  000002                          RTI                     ;;FIXES PS
```

```
 10159 054374                           GETDIS: SUBTST  <<SUBR  DISPLAY>>
                                        ;*****************************************************************************
                                        ;*SUBTEST        SUBR    DISPLAY
                                        ;*****************************************************************************
 10160 054374  113737  002100  002011          MOVB    BANK,$BANK
 10161 054402  113737  002260  002010          MOVB    REALPAT,$PATMAR
 10162 054410                                  PUSH    R0
 10163 054412  005737  002124                  TST     RLFLAG                  ;ARE WE RELOCATED?
 10164 054416  001403                          BEQ     1$                      ;NO - SKIP
 10165 054420  052737  100000  002010          BIS     #BIT15,$PATMAR          ;YES - SET MSB
 10166 054426                           1$:
 10170 054426  013777  002010   26144         MOV     $PATMAR,@DISPLAY
 10171 054434  013737  002010  000174         MOV     $PATMAR,DISPREG         ;SOFTWARE DISPLAY REGISTER
 10172 054442                                  POP     R0
 10173 054444  000207                          RETURN
```

```
10176                                   .SBTTL  ROUTINE ERROR HANDLER
10177                                 ;*************************************************************************************
10178                                 ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
10179                                 ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
10180                                 ;*AND GO TO $ERRTYP ON ERROR
10181                                 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
10182                                 ;*SW15=1        HALT ON ERROR
10183                                 ;*SW13=1        INHIBIT ERROR TYPEOUTS
10184                                 ;*SW10=1        BELL ON ERROR
10185                                 ;*SW9=1 LOOP ON ERROR
10186                                 ;*CALL
10187                                 ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
10188
10189                                   .ENABL  LSB
10190 054446                    $ERROR: IF NOERROR IS FALSE
10191 054454  104410                      CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
10192 054456  105237  002012      1$:       INCB  $ERFLG            ;;SET THE ERROR FLAG
10193 054462  001775                      BEQ   1$                ;;DON'T LET THE FLAG GO TO ZERO
10194 054464  004737  054374              CALL  GETDIS            ;SETUP DISPLAY STUFF
10195 054470  013737  002010  061262      MOV   $PATMAR,$TESTN    ;FOR APT
10196 054476  032777  002000  126072      BIT   #SW10,@SWR        ;;BELL ON ERROR?
10197 054504  001404                      BEQ   2$                ;;NO - SKIP
10198 054506                              TYPE  $BELL             ;;RING BELL
10199 054512                              TYPE  MSG014            ;CONTROL Z
10200 054516  005237  002570      2$:       INC   $ERTTL            ;;COUNT THE NUMBER OF ERRORS
10201 054522                              IF RESULT IS MI
10202 054524  012737  077777  002570          MOV #77777,$ERTTL
10203 054532                              END ;OF IF RESULT
10204 054532                            END ;OF IF NOERROR
10205 054532  011637  002016            MOV     (SP),ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
10206 054536  162737  000002  002016    SUB     #2,ERRPC
10207 054544  010637  002022            MOV     SP,ERRSP
10208 054550  016637  000002  002026    MOV     2(SP),ERRPSW
10209 054556  117737  125234  002013    MOVB    @ERRPC,$ITEMB   ;;STRIP AND SAVE THE ERROR ITEM CODE
10210 054564                            IF NOERROR IS FALSE
10211 054572                              IF BADPC NE #0
10212 054600  013737  002020  002016        MOV BADPC,ERRPC
10213 054606  162737  000002  002016        SUB #2,ERRPC
10214 054614  013737  002024  002022        MOV BADSP,ERRSP
10215 054622  013737  002030  002026        MOV BADPSW,ERRPSW
10216 054630  005037  002020                CLR BADPC
10217 054634                              END ;IF
10218 054634  013737  002016  061260      MOV   ERRPC,$FATAL    ;FOR APT
10219 054642                              IF #SW13 SET.IN @SWR
10220 054652  000412                        BR  3$
10221 054654                              END ;OF IF #SW13
10227 054654                              IF #SW5 SET.IN @SWR AND TOOMANY IS TRUE
10228 054672                                GOTO 3$
10229 054674                              END ;OF IF #SW5
10230 054674                            END ;OF IF NOERROR
10231 054674  004737  055110            CALL    $ERRTYP         ;;GO TO USER ERROR ROUTINE
```

```
10233 054700                                3$:     IF NOERROR IS FALSE
10234 054706  005777  125664                         TST   @SWR              ;;HALT ON ERROR
10235 054712  100002                                BPL   7$                ;;SKIP IF CONTINUE
10236 054714  000000                        $HALT:  HALT                    ;;HALT ON ERROR!
10237 054716  104410                                CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
10238 054720                                7$:     IF NOSCOPE IS FALSE AND #SW9 SET.IN @SWR
10239 054736  013716  002564                           MOV $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
10240 054742                                         END ;OF IF NOSCOPE
10241 054742  005737  002332                         TST   $ESCAPE           ;;CHECK FOR AN ESCAPE ADDRESS
10242 054746  001402                                BEQ   9$                ;;BR IF NONE
10243 054750  013716  002332                        MOV   $ESCAPE,(SP)      ;;FUDGE RETURN ADDRESS FOR ESCAPE
10244 054754                                9$:     IF DETFLAG IS FALSE
10245 054762  005737  002424                           TST N022BIT
10246 054766  001002                                   BNE 11$
10247 054770  005037  177766                           CLR CPUERR
10248 054774                                      11$: IF ACTFLAG IS TRUE OR APTFLAG IS TRUE OR FATAL$ IS TRUE
10249 055016  012737  000001  061256                     MOV       #1,$MSGTY       ;FOR APT
10250 055024  000137  043534                           JMP       EXIT
10251 055030                                         END ;OF IF ACTFLAG
10252 055030                                         IF XXDPCHAIN IS TRUE AND $ERTTL HI #20
10253 055046                                           TYPE      MSG066          ;ERROR COUNT EXCEEDED 20 - ABORTING FOR XXDP CHAIN
10254 055052  013700  000042                           MOV       42,R0
10255 055056  005037  000042                           CLR       42
10256 055062  000137  013306                           JMP       $Z;?24
10257 055066                                         END ;OF IF XXDPCHAIN
10258 055066                                       END ;OF IF DETFLAG
10259 055066                                      ELSE
10260 055070                                        SET   HEADER
10261 055076                                      END ;OF IF NOERROR
10262 055076                                10$:    CLEAR   TOOMANY,NOERROR
10263 055106  000002                                RTI                     ;;RETURN
10264                                               .DSABL  LSB
```

```
                                .SBTTL  ROUTINE ERROR MESSAGE TYPEOUT

                        ;*******************************************************************************
                        ;*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.

055110  104415          $ERRTYP:SAVREG
055112                          TYPE    $CRLF           ;;''CARRIAGE RETURN'' & 'LINE FEED''
055116  005000                  CLR     R0              ;;PICKUP THE ITEM INDEX
055120  153700  002013          BISB    $ITEMB,R0
055124  001004                  BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                        ;;TYPE THE PC OF THE ERROR
055126                          TYPOCT  ERRPC,<ERROR ADDRESS>
055134  000503                  BR      11$             ;;GET OUT
055136  005300          1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
055140  006300                  ASL     R0              ;;      WORK FOR THE ERROR TABLE
055142  006300                  ASL     R0
055144  006300                  ASL     R0
055146  062700  065512          ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
055152  012037  055210          MOV     (R0)+,3$        ;;PICKUP 'ERROR MESSAGE'' POINTER
055156  001417                  BEQ     4$              ;;SKIP TYPEOUT IF NO POINTER
055160  005737  002400          TST     NOERROR         ;IS THIS REALLY AN ERROR?
055164  001003                  BNE     12$             ;YES - SKIP
055166  005737  002552          TST     HEADER          ;TYPE HEADER?
055172  100011                  BPL     4$              ;NO - SKIP
055174  005737  002062  12$:    TST     FATAL$          ;WAS IT A FATAL ERROR?
055200  001402                  BEQ     2$              ;NO - SKIP
055202                          TYPE    MSG067          ;FATAL
055206                  2$:     TYPE                    ;;TYPE THE 'ERROR MESSAGE''
055210  000000          3$:     .WORD   0               ;;''ERROR MESSAGE'' POINTER GOES HERE
055212                          TYPE    $CRLF           ;;''CARRIAGE RETURN'' & 'LINE FEED''
055216  012037  055242  4$:     MOV     (R0)+,5$        ;;PICKUP 'DATA HEADER'' POINTER
055222  001412                  BEQ     6$              ;;SKIP TYPEOUT IF 0
055224  005737  002400          TST     NOERROR         ;IS THIS REALLY AN ERROR?
055230  001003                  BNE     13$             ;YES - SKIP
055232  005737  002552          TST     HEADER          ;TYPE HEADER?
055236  100004                  BPL     6$              ;NO - SKIP
055240                  13$:    TYPE                    ;;TYPE THE 'DATA HEADER''
055242  000000          5$:     .WORD   0               ;;''DATA HEADER'' POINTER GOES HERE
055244                          TYPE    $CRLF           ;;''CARRIAGE RETURN'' & 'LINE FEED''
055250  012001          6$:     MOV     (R0)+,R1        ;;PICKUP 'DATA TABLE'' POINTER
055252  001427                  BEQ     10$             ;;BR IF NO DATA TO BE TYPED
055254  012002                  MOV     (R0)+,R2        ;;PICKUP 'DATA FORMAT'' POINTER
```

```
 10313 055256  112203                  7$:     MOVB    (R2)+,R3
 10314 055260  006303                          ASL     R3              ;MAKE IT A WORD ADDRESS
 10315 055262  004773  055270                    CALL    @8$(R3)
 10316 055266  000412                          BR      9$
 10317 055270  055404                  8$:     TAG70$
 10318 055272  055414                          TAG71$
 10319 055274  055424                          TAG72$
 10320 055276  055474                          TAG73$
 10321 055300  055534                          TAG74$
 10322 055302  055546                          TAG75$
 10323 055304  055560                          TAG76$
 10324 055306  055624                          TAG77$
 10325 055310  055632                          TAG78$
 10326 055312  055712                          TAG79$
 10331 055314  062701  000002            9$:     ADD     #2,R1           ;UPDATE DATA TABLE POINTER
 10332 055320  005711                          TST     (R1)            ;;IS THERE ANOTHER NUMBER?
 10333 055322  001403                          BEQ     10$             ;;BR IF NO
 10334 055324                                  TYPE    MSG018          ;TYPE 2 SPACES
 10335 055330  000752                          BR      7$              ;;LOOP
 10336
 10337 055332  005737  002106            10$:    TST     MUT             ;IS THERE A MEMORY UNDER TEST
 10338 055336  001402                          BEQ     11$             ;NO - SKIP
 10339 055340  005237  002552                    INC     HEADER          ;YES - BUMP HEADER FLAG
 10340 055344  104416                  11$:    RESREG
 10341 055346                                  IF #SW7 SET.IN @SWR AND DETFLAG IS FALSE AND NOERROR IS FALSE
 10342 055372  004737  055734                      CALL  DETAIL
 10343 055376                                  END ;OF IF #SW7
 10344 055376                                  TYPE    MSG104          ;CONTROL 7
 10345 055402  000207                          RETURN
```

```
10348                                   ;*****************************************************************************
10349                                   ;*** OCTAL ***
10350                                   ;*****************************************************************************
10351 055404                            TAG70$: TYPOCT  @(R1)           ;;TYPE AN OCTAL NUMBER
10352 055412  000207                            RETURN
10353
10354                                   ;*****************************************************************************
10355                                   ;*** DECIMAL ***
10356                                   ;*****************************************************************************
10357 055414                            TAG71$: TYPDEC  @(R1)           ;;TYPE A DECIMAL NUMBER
10358 055422  000207                            RETURN
10359
10360                                   ;*****************************************************************************
10361                                   ;*** INTERLEAVE ***
10362                                   ;*****************************************************************************
10363 055424                            TAG72$: PUSH    R1,R5
10364 055430  013701  002100                    MOV     BANK,R1
10365 055434  070127  000004                    MUL     #4,R1
10366 055440                                    SET     NOTAB                   ;INDICATE NO TABLE TO BE PRINTED - NOW
10367 055446                                    TYPE    MSG014
10368 055452  004737  035714                    CALL    TCFIG1
10369 055456  005037  002342                    CLR     NOTAB
10370 055462                                    POP     R5,R1
10371 055466                                    TYPE    MSG014                  ;1 SPACE
10372 055472  000207                            RETURN
10373
10374                                   ;*****************************************************************************
10375                                   ;*** CSR ***
10376                                   ;*****************************************************************************
10377 055474                            TAG73$: PUSH    R1,R5
10378 055500  013701  002100                    MOV     BANK,R1
10379 055504  070127  000004                    MUL     #4,R1
10380 055510                                    SET     NOTAB
10381 055516  004737  036234                    CALL    TCFIG3
10382 055522  005037  002342                    CLR     NOTAB
10383 055526                                    POP     R5,R1
10384 055532  000207                            RETURN
10385
10386                                   ;*****************************************************************************
10387                                   ;*** PATTERN ***
10388                                   ;*****************************************************************************
10389 055534                            TAG74$: TYPOCS  REALPAT,<TYPE (0-77)>,2,Z
10390 055544  000207                            RETURN
10391
10392                                   ;*****************************************************************************
10393                                   ;*** BANK ***
10394                                   ;*****************************************************************************
10395 055546                            TAG75$: TYPOCS  BANK,<TYPE (0-167)>,3
10396 055556  000207                            RETURN
```

```
10398                                   ;*****************************************************************************
10399                                   ;*** MTYPE ***
10400                                   ;*****************************************************************************
10401 055560                            TAG76$: PUSH    R1,R5
10402 055564  013701  002100                    MOV     BANK,R1
10403 055570  070127  000004                    MUL     #4,R1
10404 055574                                    SET     NOTAB
10405 055602                                    TYPE    MSG019
10406 055606  004737  036046                    CALL    TCFIG2
10407 055612  005037  002342                    CLR     NOTAB
10408 055616                                    POP     R5,R1
10409 055622  000207                            RETURN
10410
10411                                   ;*****************************************************************************
10412                                   ;*** UNKNOWN DATA ***
10413                                   ;*****************************************************************************
10414 055624                            TAG77$: TYPE    MSG061
10415 055630  000207                            RETURN
10416
10417                                   ;*****************************************************************************
10418                                   ;*** PHYSICAL ADDRESS ***
10419                                   ;*****************************************************************************
10420 055632  013737  002032  002036  TAG78$: MOV     ADDRESS,PHYADD
10421 055640  162737  060000  002036          SUB     #FIRST,PHYADD
10422 055646  013737  002100  002040          MOV     BANK,PHYADD+2
10423 055654  006237  002040                  ASR     PHYADD+2
10424 055660  103003                          BCC     1$
10425 055662  052737  100000  002036          BIS     #BIT15,PHYADD
10426 055670  012746  002036          1$:     MOV     #PHYADD,-(SP)           ;POINTFR TO DOUBLE WORD ON STACK
10427 055674  004737  061136                  CALL    $DB2O                   ;CALL DOUBLE PRECISION CONVERSION ROUTINE
10428 055700  062706  000002                  ADD     #2,SP                   ;FIX STACK
10429 055704                                    TYPE    $OCT8
10430 055710  000207                            RETURN
10431
10432                                   ;*****************************************************************************
10433                                   ;*** OCTAL BYTE ***
10434                                   ;*****************************************************************************
10435 055712                            TAG79$: TYPE    MSG018          ;2 SPACES
10436 055716                                    TYPOCS  @(R1),<TYPE BYTE>,3,Z
10437 055726                                    TYPE    MSG014          ;SPACE
10438 055732  000207                            RETURN
```

```
10482 055734                              DETAIL: SUBTST  <<SUBR  DETAILED ERROR REPORT>>
                                          ;*****************************************************************************
                                          ;*SUBTEST        SUBR    DETAILED ERROR REPORT
                                          ;*****************************************************************************
10483 055734  005237  002212                        INC     DETFLAG
10484 055740  022737  000003  002212                CMP     #3,DETFLAG
10485 055746  101472                                BLOS    4$
10486 055750  022737  000002  002212                CMP     #2,DETFLAG
10487 055756  001435                                BEQ     2$
10488 055760                                        PUSH    HEADER,MUT
10489 055770                                        SET     HEADER
10490 055776  005037  002106                        CLR     MUT
10491 056002  010037  002172                        MOV     R0,DETR0
10492 056006  012700  002174                        MOV     #DETR1,R0
10493 056012  010120                                MOV     R1,(R0)+
10494 056014  010220                                MOV     R2,(R0)+
10495 056016  010320                                MOV     R3,(R0)+
10496 056020  010420                                MOV     R4,(R0)+
10497 056022  010520                                MOV     R5,(R0)+
10498 056024  013720  002022                        MOV     ERRSP,(R0)+
10499 056030  013720  002026                        MOV     ERRPSW,(R0)+
10500 056034  013700  002172                        MOV     DETR0,R0
10501 056040                                        SET     NOERROR
10502 056046  104013                                ERROR   +13
10503 056050  000422                                BR      1$
10504 056052                              2$:       PUSH    HEADER,MUT
10505 056062                                        SET     HEADER
10506 056070  005037  002106                        CLR     MUT
10507 056074                                        SET     NOERROR
10508 056102  104031                                ERROR   +31
10509 056104  005737  002424                        TST     NO22BIT
10510 056110  001002                                BNE     1$
10511 056112  005037  177766                        CLR     CPUERR
10512 056116                              1$:       POP     MUT,HEADER
10513                                               ;WARNING RECURSIVE
10514 056126  004737  055734                        CALL    DETAIL
10515 056132  000207                                RETURN
```

```
10518
10519 056134  004737  057572          4$:     CALL    CONTT                   ;DISPLAY 'DISPLAY'' INFO
10520
10521                                         ;TYPE CONTENTS OF ALL CSR'S
10522 056140                                  PUSH    CSR,CSRNO,R1
10523 056152                                  TYPE    MSG058
10524 056156                                  TYPE    $CRLF
10525 056162  013701  002216                 MOV     TOTCSRS,R1
10526 056166                                  BEGIN DUMPCSRLOOP
10527 056166                                    FOR CSRNO := #0 TO #36 BY #2
10528 056172  006301                              ASL       R1
10529 056174                                      ON.ERROR
10530 056176  104426                                READCSR
10531 056200                                        TYPOCT    CSR
10532 056206                                        TYPE      MSG018          ;2 SPACES
10533 056212                                      END ;OF ON.ERROR
10534 056212                                      IF R1 EQ #0 THEN LEAVE DUMPCSRLOOP
10535 056216                                    END ;OF FOR CSRNO
10536 056234                                  END DUMPCSRLOOP
10537 056234                                  POP     R1,CSRNO,CSR
10538
10539                                         ;TYPE STACKS
10540 056246                                  PUSH    R0,R1
10541 056252                                  TYPE    MSG088          ;KERNEL STACK
10542 056256  013701  002534                 MOV     KSTACK,R1
10543 056262  162701  000002                 SUB     #2,R1
10544 056266                                  FOR     R0 := SP TO R1 BY #2
10545 056270                                    TYPE  $CRLF
10546 056274                                    TYPOCT        R0
10547 056300                                    TYPE          MSG018          ;2 SPACES
10548 056304                                    TYPOCT        (R0)
10549 056310                                  END ;OF FOR R0
10550                                         ;SET PREVIOUS MODE TO SUPERVISOR
10551 056320  005737  002426                 TST     NOSUPER
10552 056324  001036                         BNE     DET1
10553 056326  042737  030000  177776         BIC     #BIT13!BIT12,PSW
10554 056334  052737  010000  177776         BIS     #BIT12,PSW
10555 056342  006506                         MFPI    SSP
10556 056344                                  POP     R1,R0
10557 056350                                  TYPE    MSG089          ;SUPERVISOR STACK
10558 056354                                  IF R0 LT #SUPSTK
10559 056362                                    FOR R0 := R0 TO #SUPSTK-2 BY #2
10560 056362                                      TYPE        $CRLF
10561 056366                                      TYPOCT      R0
10562 056372                                      TYPE        MSG018          ;2 SPACES
10563 056376                                      TYPOCT      (R0)
10564 056402                                    END ;OF FOR R0
10565 056414                                  ELSE
10566 056416                                    TYPE  MSG091          ;IS EMPTY
10567 056422                                  END ;OF IF R0
10568                                         ;SET PREVIOUS MODE TO USER
10569 056422  052737  030000  177776  DET1:   BIS     #BIT13!BIT12,PSW
10570 056430  006506                         MFPI    USP
10571 056432                                  POP     R0
10572 056434                                  TYPE    MSG090          ;USER STACK
10573 056440                                  IF R0 LT #USESTK
10574 056446                                    FOR R0 := R0 TO #USESTK-2 BY #2
```

```
10575 056446                                    TYPE        $CRLF
10576 056452                                    TYPOCT      R0
10577 056456                                    TYPE        MSG018          ;2 SPACES
10578 056462                                    TYPOCT      (R0)
10579 056466                                  END ;OF FOR R0
10580 056500                                ELSE
10581 056502                                  TYPE  MSG091          ;IS EMPTY
10582 056506                                END ;OF IF R0
10583 056506                                TYPE    $CRLF
10584 056512  005037  002212                  CLR     DETFLAG
10585 056516                                POP     R0
10586 056520  000207                        RETURN
```

```
                        .SBTTL  ROUTINE 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

056522  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
056526  116637  000001  056745          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
056534  112637  056747                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
056540  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
056544  000406                          BR      $TYPON
056546  112737  000001  056745  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
056554  112737  000006  056747          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
056562  112737  000005  056744  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
056570  010346                          MOV     R3,-(SP)        ;;SAVE R3
056572  010446                          MOV     R4,-(SP)        ;;SAVE R4
056574  010546                          MOV     R5,-(SP)        ;;SAVE R5
056576  113704  056747                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
056602  005404                          NEG     R4
056604  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
056610  110437  056746                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
056614  113704  056745                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
056620  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
056624  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
056626  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO 'C'
056630  000404                          BR      3$              ;;GO DO MSB
056632  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
056634  006105                          ROL     R5
056636  006105                          ROL     R5
056640  010503                          MOV     R5,R3
056642  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
056644  105337  056746                  DECB    $OMODE          ;;TYPE THIS DIGIT?
056650  100016                          BPL     6$              ;;BR IF NO
056652  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
056656  001002                          BNE     4$              ;;TEST FOR 0
056660  005704                          TST     R4              ;;SUPPRESS THIS 0?
056662  001403                          BEQ     5$              ;;BR IF YES
056664  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
```

```
10681 056666  052703  000060              BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
10682 056672  052703  000040        5$:   BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
10683 056676  110337  056742              MOVB    R3,8$           ;;SAVE FOR TYPING
10684 056702                              TYPE    8$              ;;GO TYPE THIS DIGIT
10685 056706  105337  056744        6$:   DECB    $OCNT           ;;COUNT BY 1
10686 056712  003347                      BGT     2$              ;;BR IF MORE TO DO
10687 056714  002402                      BLT     7$              ;;BR IF DONE
10688 056716  005204                      INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
10689 056720  000744                      BR      2$              ;;GO DO THE LAST DIGIT
10690 056722  012605                7$:   MOV     (SP)+,R5        ;;RESTORE R5
10691 056724  012604                      MOV     (SP)+,R4        ;;RESTORE R4
10692 056726  012603                      MOV     (SP)+,R3        ;;RESTORE R3
10693 056730  016666  000002  000004      MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
10694 056736  012616                      MOV     (SP)+,(SP)
10695 056740  000002                      RTI                     ;;RETURN
10696 056742     000                8$:   .BYTE   0               ;;STORAGE FOR ASCII DIGIT
10697 056743     000                      .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
10698 056744     000                $OCNT: .BYTE   0              ;;OCTAL DIGIT COUNTER
10699 056745     000                $OFILL: .BYTE  0              ;;ZERO FILL SWITCH
10700 056746  000000              $OMODE: .WORD   0             ;;NUMBER OF DIGITS TO TYPE
```

```
10702                                           .SBTTL  ROUTINE CONVERT BINARY TO DECIMAL AND TYPE
10703                                   ;*******************************************************************************
10704                                   ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
10705                                   ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
10706                                   ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
10707                                   ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
10708                                   ;*REPLACED WITH SPACES.
10709                                   ;*CALL:
10710                                   ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
10711                                   ;*      TYPDS                   ;;GO TO THE ROUTINE
10712 056750                           $TYPDS: PUSH    R0,R1,R2,R3,R5
10713 056762  012746  020200                   MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
10714 056766  016605  000020                   MOV     20(SP),R5       ;;GET THE INPUT NUMBER
10715 056772  100004                           BPL     1$              ;;BR IF INPUT IS POS.
10716 056774  005405                           NEG     R5              ;;MAKE THE BINARY NUMBER POS.
10717 056776  112766  000055  000001           MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
10718 057004  005000                   1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
10719 057006  012703  057164                   MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
10720 057012  112723  000040                   MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
10721 057016  005002                   2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
10722 057020  016001  057154                   MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
10723 057024  160105                   3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
10724 057026  002402                           BLT     4$              ;;BR IF DONE
10725 057030  005202                           INC     R2              ;;INCREASE THE BCD DIGIT BY 1
10726 057032  000774                           BR      3$
10727 057034  060105                   4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
10728 057036  005702                           TST     R2              ;;CHECK IF BCD DIGIT=0
10729 057040  001002                           BNE     5$              ;;FALL THROUGH IF 0
10730 057042  105716                           TSTB    (SP)            ;;STILL DOING LEADING 0'S?
10731 057044  100407                           BMI     7$              ;;BR IF YES
10732 057046  106316                   5$:     ASLB    (SP)            ;;MSD?
10733 057050  103003                           BCC     6$              ;;BR IF NO
10734 057052  116663  000001  177777           MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
10735 057060  052702  000060           6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
10736 057064  052702  000040           7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
10737 057070  110223                           MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
10738 057072  005720                           TST     (R0)+           ;;JUST INCREMENTING
10739 057074  020027  000010                   CMP     R0,#10          ;;CHECK THE TABLE INDEX
10740 057100  002746                           BLT     2$              ;;GO DO THE NEXT DIGIT
10741 057102  003002                           BGT     8$              ;;GO TO EXIT
10742 057104  010502                           MOV     R5,R2           ;;GET THE LSD
10743 057106  000764                           BR      6$              ;;GO CHANGE TO ASCII
10744 057110  105726                   8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
10745 057112  100003                           BPL     9$              ;;BR IF NO
10746 057114  116663  177777  177776           MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
10747 057122  105013                   9$:     CLRB    (R3)            ;;SET THE TERMINATOR
10748 057124                                   POP     R5,R3,R2,R1,R0
10749 057136                                   TYPE    $DBLK           ;;NOW TYPE THE NUMBER
10750 057142  016666  000002  000004           MOV     2(SP),4(SP)     ;;ADJUST THE STACK
10751 057150  012616                           MOV     (SP)+,(SP)
10752 057152  000002                           RTI                     ;;RETURN TO USER
10753 057154  023420                   $DTBL:  10000.
10754 057156  001750                           1000.
10755 057160  000144                           100.
10756 057162  000012                           10.
10757 057164  000000  000000  000000   $DBLK:  .WORD   0,0,0,0
      057172  000000
```

```
10759                                           .SBTTL  ROUTINE TTY INPUT
10760                                   ;*******************************************************************************
10761                                   ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
10762                                   ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
10763                                   ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
10764                                   ;*WHEN OPERATING IN TTY FLAG MODE.
10765                                           .ENABLE LSB
10766 057174                           $CKSWR:
10772 057174  105777  123402                     TSTB    @$TKS           ;;CHAR THERE?
10773 057200  100135                             BPL     12$             ;;IF NO, DON'T WAIT AROUND
10774 057202  117746  123376                     MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
10775 057206  042716  177600                     BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
10776 057212  022716  000006           CONTS1:   CMP     #6,(SP)         ;IS IT CONTROL F?
10777 057216  001002                             BNE     1$              ;NO SKIP
10778 057220  004737  044004                     CALL    FIELDSERVICE
10779 057224  022716  000024           1$:       CMP     #24,(SP)        ;IS IT CONTROL T?
10780 057230  001002                             BNE     16$             ;NO - SKIP
10781 057232  004737  057572                     CALL    CONTT           ;YES - CALL CONTROL T ROUTINE
10782 057236  022716  000003           16$:      CMP     #3,(SP)         ;IS IT CONTROL C?
10783 057242  001461                             BEQ     5$              ;YES EXIT *****NOTE***** STACK IS SCREWED UP!
10784 057244  022716  000004                     CMP     #4,(SP)         ;IS IT CONTROL D?
10785 057250  001002                             BNE     2$              ;NO - SKIP
10786 057252  000137  062114           CONTP:    JMP     @#0.ODT         ;YES - GO TO ODT
10787 057256  022716  000023           2$:       CMP     #23,(SP)        ;IS IT CONTROL S?
10788 057262  001002                             BNE     17$             ;NO - SKIP
10789 057264  004737  057646                     CALL    CONTS           ;YES - CALL CONTROL S ROUTINE
10790 057270  022716  000013           17$:      CMP     #13,(SP)        ;IS IT CONTROL K?
10791 057274  001005                             BNE     6$              ;NO - SKIP
10792 057276                                     TYPE    $CNTLK          ;TYPE A ^K
10793 057302  013706  002142                     MOV     CTLKVEC,SP      ;RESET KSP TO AFTER PATTERN EXEC ROUTINE
10794 057306  000207                             RETURN                  ;RETURN TO PATTERN EXEC ROUTINE
10795 057310  022737  000176  002576  6$:       CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
10796 057316  001067                             BNE     CKEND           ;;BRANCH IF NO
10797 057320  022726  000007                     CMP     #7,(SP)+        ;;IS IT A CONTROL G?
10798 057324  001064                             BNE     CKEND           ;;NO, RETURN TO USER
10799 057326  005737  002060                     TST     $AUTO           ;ARE WE RUNNING IN AUTO-MODE?
10800 057332  001061                             BNE     CKEND           ;BRANCH IF YES
10801 057334                                     TYPE    $CNTLG          ;;ECHO THE CONTROL-G (^G)
10802 057340                           $GTSWR:   TYPE    $MSWR           ;;TYPE CURRENT CONTENTS
10803 057344                                     TYPOCT  @SWR            ;;OF THE SWR
10804 057352                                     TYPE    $MNEW           ;;PROMPT FOR NEW SWR
10805 057356  005046                     3$:     CLR     -(SP)           ;;CLEAR COUNTER
10806 057360  005046                             CLR     -(SP)           ;;THE NEW SWR
10807 057362  105777  123214           4$:       TSTB    @$TKS           ;CHAR THE E?
10808 057366  100375                             BPL     4$              ;;IF NOT TRY AGAIN
10809 057370  117746  123210                     MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
10810 057374  042716  177600                     BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
10811 057400  021627  000003                     CMP     (SP),#3         ;;IS IT A CONTROL-C?
10812 057404  001006                             BNE     7$              ;;BRANCH IF NOT
10813 057406                           5$:       TYPE    $CNTLC          ;;YES, ECHO CONTROL-C (^C)
10814 057412  062706  000006                     ADD     #6,SP           ;;CLEAN UP STACK
10815 057416  000137  043426                     JMP     BOOT            ;;CONTROL-C RESTART
10816 057422  021627  000025           7$:       CMP     (SP),#25        ;;IS IT A CONTROL-U?
10817 057426  001005                             BNE     9$              ;;BRANCH IF NOT
10818 057430                                     TYPE    $CNTLU          ;;YES, ECHO CONTROL-U (^U)
10819 057434  062706  000006           8$:       ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
10820 057440  000746                             BR      3$              ;;LET'S TRY IT AGAIN
```

```
10821 057442  021627  000015          9$:     CMP     (SP),#15        ;;IS IT A <CR>?
10822 057446  001016                          BNE     13$             ;;BRANCH IF NO
10823 057450  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
10824 057454  001403                          BEQ     10$             ;;BRANCH IF YES
10825 057456  016677  000002  123112          MOV     2(SP),@SWR      ;;SAVE NEW SWR
10826 057464  062706  000006          10$:    ADD     #6,SP           ;;CLEAR UP STACK
10827 057470                                  TYPE    $CRLF           ;;ECHO <CR> AND <LF>
10828 057474  000002                  12$:    RTI                     ;;RETURN
10829 057476  062706  000002          CKEND:  ADD     #2,SP           ;FIX STACK
10830 057502  000002                          RTI                     ;RETURN
10831 057504  004737  052364          13$:    CALL    $TYPEC          ;;ECHO CHAR
10832 057510  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
10833 057514  002420                          BLT     15$             ;;BRANCH IF YES
10834 057516  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
10835 057522  003015                          BGT     15$             ;;BRANCH IF YES
10836 057524  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
10837 057530  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
10838 057534  001403                          BEQ     14$             ;;BRANCH IF YES
10839 057536  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
10840 057540  006316                          ASL     (SP)            ;;    CHAR OVER TO MAKE
10841 057542  006316                          ASL     (SP)            ;;    ROOM FOR NEW ONE.
10842 057544  005266  000002          14$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
10843 057550  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
10844 057554  000702                          BR      4$              ;;GET THE NEXT ONE
10845 057556                          15$:    TYPE    $QUES           ;;TYPE ?<CR><LF>
10846 057562  000724                          BR      8$              ;;SIMULATE CONTROL-J
10847 057564     136     113     015  $CNTLK: .ASCIZ   /^K/<15><12>    ;CONTROL K ASCII STRING
      057567     012     000
10848                                         .EVEN
10849                                         .DSABL  LSB
```

```
10852 057572                            CONTT:  SUBTST  <<CONTROL T>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       CONTROL T
                                        ;*****************************************************************************
10853 057572                                    PUSH    R0
10854 057574                                    TYPE    $CRLF
10864 057600                                      IF RLFLAG IS TRUE
10865 057606                                        TYPE        MSG092          ;RELOCATED
10866 057612                                      END ;OF IF RLFLAG
10867 057612                                      TYPE  MSG093                  ;BANK=
10868 057616                                      TYPOCS        BANK,,3         ;TYPE 3 DIGITS
10869 057626                                      TYPE  MSG095                  ;PAT=
10870 057632                                      TYPOCS        REALPAT,,2      ;TYPE 2 DIGITS
10874 057642                                    POP     R0
10875 057644  000207                            RETURN
10876
10877 057646                            CONTS:  SUBTST  <<CONTROL S & CONTROL Q>>
                                        ;*****************************************************************************
                                        ;*SUBTEST       CONTROL S & CONTROL Q
                                        ;*****************************************************************************
10878 057646                                    POP     R0              ;GET RID OF RETURN ADDRESS FROM STACK
10879 057650  105777  122726              CONTS2: TSTB    @$TKS           ;W^IT FOR CHARACTER
10880 057654  100375                            BPL     CONTS2
10881 057656  117716  122722                    MOVB    @$TKB,(SP)      ;REPLACE OVER OLD CHARACTER ON STACK
10882 057662  042716  177600                    BIC     #^C177,(SP)     ;STRIP AIL BUT ASCII
10883 057666                                    IF (SP) EQ #21          ;IF IT IS A CONTROL Q
10884 057674  000137  057212                      JMP   CONTS1
10885 057700                                    ELSE
10886 057702  000762                              BR    CONTS2
10887 057704                                    END ;OF IF (SP)
```

```
10889                                   ;*************************************************************************
10890                                   ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
10891                                   ;*CALL:
10892                                   ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
10893                                   ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
10894                                   ;*                              ;;WITH PARITY BIT STRIPPED OFF
10895                                   ;
10896
10897 057704  011646                   $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
10898 057706  016666  000004  000002           MOV     4(SP),2(SP)     ;;SAVE THE PS
10899 057714  105777  122662           1$:     TSTB    @$TKS           ;;WAIT FOR
10900 057720  100375                           BPL     1$              ;;A CHARACTER
10901 057722  117766  122656  000004           MOVB    @$TKB,4(SP)     ;;READ THE TTY
10902 057730  042766  177600  000004           BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
10903 057736  026627  000004  000023           CMP     4(SP),#23       ;;IS IT A CONTROL-S?
10904 057744  001013                           BNE     3$              ;;BRANCH IF NO
10905 057746  105777  122630           2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
10906 057752  100375                           BPL     2$              ;;LOOP UNTIL ITS THERE
10907 057754  117746  122624                   MOVB    @$TKB,-(SP)     ;;GET CHARACTER
10908 057760  042716  177600                   BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
10909 057764  022627  000021                   CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
10910 057770  001366                           BNE     2$              ;;IF NOT DISCARD IT
10911 057772  000750                           BR      1$              ;;YES, RESUME
10912 057774  026627  000004  000140   3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
10913 060002  002407                           BLT     4$              ;;BRANCH IF YES
10914 060004  026627  000004  000175           CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
10915 060012  003003                           BGT     4$              ;;BRANCH IF YES
10916 060014  042766  000040  000004           BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
10917 060022  000002                   4$:     RTI                     ;;GO BACK TO USER
10918                                   ;*************************************************************************
10919                                   ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
10920                                   ;*CALL:
10921                                   ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
10922                                   ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
10923                                   ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
10924 060024  010346                   $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
10925 060026  005046                           CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
10926 060030  012703  060322           1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
10927 060034  022703  060346           2$:     CMP     #$TTYIN+20.,R3  ;;BUFFER FULL?
10928 060040  101477                           BLOS    8$              ;;BR IF YES
10929 060042  104411                           RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
10930 060044  112613                           MOVB    (SP)+,(R3)      ;;GET CHARACTER
10931 060046  122713  000003                   CMPB    #3,(R3)         ;;IS IT A CONTROL-C?
10932 060052  001016                           BNE     3$              ;;BRANCH IF NO
10933 060054                                   TYPE    $CNTLC          ;;TYPE A CONTROL-C (^C)
10934 060060  005726                           TST     (SP)+           ;;CLEAN RUBOUT KEY OFF OF THE STACK
10935 060062  012603                           MOV     (SP)+,R3        ;;RESTORE R3
10936 060064  032777  000400  122504           BIT     #BIT8,@SWR      ;;IS THERE A HALT FLAG SET IN THE SWR?
10937 060072  001404                           BEQ     11$             ;;BRANCH IF NOT TO BOOT ROUTINE
10938 060074  005037  002370                   CLR     STOPOK          ;;GET READY TO HALT PROGRAM
10939 060100  000137  043534                   JMP     EXIT            ;;GO HALT PROGRAM
10940 060104  000137  043426           11$:    JMP     BOOT            ;;GOTO CONTROL-C RESTART
10941 060110  122713  000177           3$:     CMPB    #177,(R3)       ;;IS IT A RUBOUT
10942 060114  001022                           BNE     5$              ;;BR IF NO
10943 060116  005716                           TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
10944 060120  001007                           BNE     4$              ;;BR IF NO
10945 060122  112737  000134  060320           MOVB    #'\,10$         ;;TYPE A BACK SLASH
```

```
10946 060130                                    TYPE    10$
10947 060134  012716  177777                    MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
10948 060140  005303                    4$:     DEC     R3              ;;BACKUP BY ONE
10949 060142  020327  060322                    CMP     R3,#$TTYIN      ;;STACK EMPTY?
10950 060146  103434                            BLO     8$              ;;BR IF YES
10951 060150  111337  060320                    MOVB    (R3),10$                ;;SETUP TO TYPEOUT THE DELETED CHAR.
10952 060154                                    TYPE    10$             ;;GO TYPE
10953 060160  000725                            BR      2$              ;;GO READ ANOTHER CHAR.
10954 060162  005716                    5$:     TST     (SP)            ;;RUBOUT KEY SET?
10955 060164  001406                            BEQ     6$              ;;BR IF NO
10956 060166  112737  000134  060320            MOVB    #'\,10$         ;;TYPE A BACK SLASH
10957 060174                                    TYPE    10$
10958 060200  005016                            CLR     (SP)            ;;CLEAR THE RUBOUT KEY
10959 060202  122713  000025            6$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
10960 060206  001003                            BNE     7$              ;;BR IF NO
10961 060210                                    TYPE    $CNTLU          ;;TYPE A CONTROL 'U'
10962 060214  000705                            BR      1$              ;;GO START OVER
10963 060216  122713  000022            7$:     CMPB    #22,(R3)        ;;IS CHARACTER A '^R'?
10964 060222  001011                            BNE     9$              ;;BRANCH IF NO
10965 060224  105013                            CLRB    (R3)            ;;CLEAR THE CHARACTER
10966 060226                                    TYPE    $CRLF           ;;TYPE A 'CR' & 'LF'
10967 060232                                    TYPE    $TTYIN          ;;TYPE THE INPUT STRING
10968 060236  000676                            BR      2$              ;;GO PICKUP ANOTHER CHACTER
10969 060240                            8$:     TYPE    $QUES           ;;TYPE A '?'
10970 060244  000671                            BR      1$              ;;CLEAR THE BUFFER AND LOOP
10971 060246  111337  060320            9$:     MOVB    (R3),10$                ;;ECHO THE CHARACTER
10972 060252                                    TYPE    10$
10973 060256  122723  000015                    CMPB    #15,(R3)+       ;;CHECK FOR RETURN
10974 060262  001264                            BNE     2$              ;;LOOP IF NOT RETURN
10975 060264  105063  177777                    CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
10976 060270                                    TYPE    $LF             ;;TYPE A LINE FEED
10977 060274  005726                            TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
10978 060276  012603                            MOV     (SP)+,R3        ;;RESTORE R3
10979 060300  011646                            MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
10980 060302  016666  000004  000002            MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
10981 060310  012766  060322  000004            MOV     #$TTYIN,4(SP)
10982 060316  000002                            RTI                     ;;RETURN
10983 060320     000                    10$:    .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
10984 060321     000                            .BYTE   0               ;;TERMINATOR
10985 060322  000024                    $TTYIN: .REPT   20.             ;;RESERVE SIZE BYTES FOR TTY INPUT
10988 060346     136     103     015    $CNTLC: .ASCIZ  /^C/<15><12>    ;;CONTROL 'C'
      060351     012     000
10989 060353     136     125     015    $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL 'U'
      060356     012     000
10990 060360     136     107     015    $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL 'G'
      060363     012     000
10991 060365     015     012     123    $MSWR:  .ASCIZ  <15><12>/SWR = /
      060370     127     122     040
      060373     075     040     000
10992 060376     040     040     116    $MNEW:  .ASCIZ  /  NEW = /
      060401     105     127     040
      060404     075     040     000
10993                                   .EVEN
```

```
                                        .SBTTL  ROUTINE 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
060410  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
060412  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
060420                                  PUSH    R0,R1,R2
060426  104412                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
060430  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
060432  010037  060536                  MOV     R0,5$           ;;AND SAVE IT
060436  005001                          CLR     R1              ;;CLEAR DATA WORD
060440  005002                          CLR     R2
060442  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
060444  001420                          BEQ     3$              ;;IF ZERO GET OUT
060446  122716  000060                  CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
060452  003026                          BGT     4$              ;;IS AN OCTAL DIGIT
060454  122716  000067                  CMPB    #'7,(SP)
060460  002423                          BLT     4$
060462  006301                          ASL     R1              ;;*2
060464  006102                          ROL     R2
060466  006301                          ASL     R1              ;;*4
060470  006102                          ROL     R2
060472  006301                          ASL     R1              ;;*8
060474  006102                          ROL     R2
060476  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
060502  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
060504  000756                          BR      2$              ;;LOOP
060506  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
060510  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
060514  010237  060556                  MOV     R2,$HIOCT
060520                                  POP     R2,R1,R0
060526  000002                          RTI                     ;;RETURN
060530  005726                  4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
060532  105010                          CLRB    (R0)            ;;SET A TERMINATOR
060534                                  TYPE                    ;;TYPE UP THRU THE BAD CHAR.
060536  000000                  5$:     .WORD   0
060540                                  TYPE    MSG062          ;INPUT MUST BE A
060544                                  TYPE    MSG063          ;N OCTAL
060550                                  TYPE    MSG064          ;NUMBER
060554  000724                          BR      1$              ;;TRY AGAIN
060556  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
                                        .SBTTL  ROUTINE 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
```

```
11052                                   ;*POSITIVE 32767 TO NEGATIVE 32768.
11053                                   ;*CALL:
11054                                   ;*      RDDEC                   ;;READ A DECIMAL NUMBER
11055                                   ;*      RETURN HERE             ;;NUMBER IS ON TOP OF THE STACK
11056                                   ;
11057
11058 060560  011646                   $RDDEC: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR
11059 060562  016666  000004  000002           MOV     4(SP),2(SP)     ;;THE INPUT NUMBER
11060 060570                                   PUSH    R0,R1,R2
11061 060576  104412                   1$:     RDLIN                   ;;READ AN ASCIZ LINE
11062 060600  012600                           MOV     (SP)+,R0        ;;ADDRESS OF 1ST CHAR.
11063 060602  010037  060726                   MOV     R0,6$           ;;SAVE INCASE OF BAD INPUT
11064 060606  005046                           CLR     -(SP)           ;;CLEAR DATA WORD
11065 060610  005002                           CLR     R2              ;;SIGN SET POSITIVE
11066 060612  122710  000055                   CMPB    #'-,(R0)        ;;SEE IF A MINUS SIGN WAS TYPED
11067 060616  001001                           BNE     2$              ;;BR IF NO MINUS SIGN
11068 060620  112002                           MOVB    (R0)+,R2        ;;SAVE FOR LATER USE
11069 060622  112001                   2$:     MOVB    (R0)+,R1        ;;PICKUP THIS CHARACTER
11070 060624  001424                           BEQ     3$              ;;GET OUT IF ZERO
11071 060626  122701  000060                   CMPB    #'0,R1          ;;MAKE SURE THIS CHARACTER
11072 060632  003032                           BGT     5$              ;;IS A DIGIT BETWEEN 0 & 9
11073 060634  122701  000071                   CMPB    #'9,R1
11074 060640  002427                           BLT     5$
11075 060642  032716  170000                   BIT     #^C7777,(SP)    ;;DON'T LET NUMBER GET TO BIG
11076 060646  001024                           BNE     5$              ;;BR IF NUMBER WOULD OVERFLOW
11077 060650  006316                           ASL     (SP)            ;;*2
11078 060652  011646                           MOV     (SP),-(SP)      ;;SAVE FOR LATER
11079 060654  006316                           ASL     (SP)            ;;*4
11080 060656  006316                           ASL     (SP)            ;;*8.
11081 060660  062616                           ADD     (SP)+,(SP)      ;;*10.
11082 060662  102416                           BVS     5$              ;;OVERFLOW ISN'T ALLOWED
11083 060664  162701  000060                   SUB     #'0,R1          ;;STRIP AWAY THE ASCII JUNK
11084 060670  060116                           ADD     R1,(SP)         ;;ADD IN THIS DIGIT
11085 060672  102412                           BVS     5$              ;;OVERFLOW ISN'T ALLOWED
11086 060674  000752                           BR      2$              ;;LOOP
11087 060676  005702                   3$:     TST     R2              ;;CHECK IF NUMBER IS NEG
11088 060700  001401                           BEQ     4$              ;;BR IF NO
11089 060702  005416                           NEG     (SP)            ;;YES--NEGATE THE NUMBER
11090 060704  012666  000012           4$:     MOV     (SP)+,12(SP)    ;;SAVE THE RESULT
11091 060710                                   POP     R2,R1,R0
11092 060716  000002                           RTI                     ;;RETURN
11093
11094 060720  005726                   5$:     TST     (SP)+           ;;CLEAN PARTIAL NUMBER FROM STACK
11095 060722  105010                           CLRB    (R0)            ;;SET A TERMINATOR
11096 060724                                   TYPE                    ;;TYPE THE INPUT UP TO BAD CHAR.
11097 060726  000000                   6$:     .WORD   0               ;;POINTER GOES HERE
11098 060730                                   TYPE    MSG062          ;INPUT MUSST BE A
11099 060734                                   TYPE    MSG065          ;DECIMAL
11100 060740                                   TYPE    MSG064          ;NUMBER
11101 060744  000714                           BR      1$              ;;TRY AGAIN
```

```
11103                                           .SBTTL  ROUTINE SAVE AND RESTORE R0-R5
11104
11105                                   ;*******************************************************************************
11106                                   ;*SAVE R0-R5
11107                                   ;*CALL:
11108                                   ;*      SAVREG
11109                                   ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
11110                                   ;*
11111                                   ;*TOP---(+16)
11112                                   ;* +2---(+18)
11113                                   ;* +4---R5
11114                                   ;* +6---R4
11115                                   ;* +8---R3
11116                                   ;*+10---R2
11117                                   ;*+12---R1
11118                                   ;*+14---R0
11119
11120 060746                            $SAVREG:
11121 060746                                    PUSH    R0,R1,R2,R3,R4,R5
11122 060762  016646  000022                    MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
11123 060766  016646  000022                    MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
11124 060772  016646  000022                    MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
11125 060776  016646  000022                    MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
11126 061002  000002                            RTI
11127
11128                                   ;*RESTORE R0-R5
11129                                   ;*CALL:
11130                                   ;*      RESREG
11131 061004                            $RESREG:
11132 061004  012666  000022                    MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
11133 061010  012666  000022                    MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
11134 061014  012666  000022                    MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
11135 061020  012666  000022                    MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
11136 061024                                    POP     R5,R4,R3,R2,R1,R0
11137 061040  000002                            RTI
```

```
                                        .SBTTL  ROUTINE RANDOM NUMBER GENERATOR

                                ;*******************************************************************************
                                ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
                                ;*WITH A RANGE OF 0 TO 2**(+33)-1.
                                ;*CALL:
                                ;*      CALL    $RAND           ;;CALL THE ROUTINE
                                ;*      RETURN                  ;;RETURN HERE THE RANDOM
                                ;*                              ;;NUMBER WILL BE IN
                                ;*                              ;;$HINUM,$LONUM

061042                          $RAND:  PUSH    R0,R1,R2
061050  013700  002544                  MOV     SEEDLO,R0       ;SET R0 WITH LOW
061054  013701  002542                  MOV     SEEDHI,R1       ;SET R1 WITH HIGH
061060  012702  000007                  MOV     #7,R2           ;SET SHIFT COUNT
061064  006300                  1$:     ASL     R0              ;;SHIFT R0 LEFT AND
061066  006101                          ROL     R1              ;;ROTATE CARRY INTO R1 AND
061070  077203                          SOB     R2,1$
061072  063700  002544                  ADD     SEEDLO,R0       ;ADD NUMBER TO MAKE X 129
061076  005501                          ADC     R1              ;;PROPOGATE CARRY
061100  063701  002542                  ADD     SEEDHI,R1       ;ADD NUMBER TO MAKE X 129
061104  062700  001057                  ADD     #1057,R0        ;;ADD LOW CONSTANT
061110  005501                          ADC     R1              ;;PROPOGATE CARRY
061112  062701  047401                  ADD     #47401,R1       ;;ADD HIGH CONSTANT
061116  010037  002544                  MOV     R0,SEEDLO       ;SAVE R0
061122  010137  002542                  MOV     R1,SEEDHI       ;SAVE R1
061126                                  POP     R2,R1,R0
061134  000207                          RETURN
```

```
                                        .SBTTL  ROUTINE DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT
                                ;*******************************************************************************
                                ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
                                ;*UNSIGNED OCTAL ASCIZ NUMBER.
                                ;*CALL
                                ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
                                ;*      CALL    $DB2O           ;;CALL THE ROUTINE
                                ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK

061136  104415          $DB2O:  SAVREG                  ;;SAVE ALL REGISTERS
061140  016601  000002          MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
061144  012705  061255          MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
061150  012704  000014          MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
061154  012703  177770          MOV     #^C7,R3         ;;MASK
061160  012100                  MOV     (R1)+,R0        ;;LOWER WORD
061162  012101                  MOV     (R1)+,R1        ;;HIGH WORD
061164  005002                  CLR     R2              ;;TERMINATOR
061166  110245          1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
061170  010002                  MOV     R0,R2           ;;GET THIS DIGIT
061172  005304                  DEC     R4              ;;COUNT THIS CHARACTER
061174  003007                  BGT     3$              ;;BR IF NOT THE LAST DIGIT
061176  001405                  BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
061200  005205                  INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
061202  010566  000002          MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
061206  104416                  RESREG                  ;;RESTORE ALL REGISTERS
061210  000207                  RETURN                  ;;RETURN TO USER
061212  006203          2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
061214  006001          3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
061216  006000                  ROR     R0              ;;      THE NEXT OCTAL DIGIT
061220  006001                  ROR     R1
061222  006000                  ROR     R0
061224  006001                  ROR     R1
061226  006000                  ROR     R0
061230  040302                  BIC     R3,R2           ;;MASK OUT ALL JUNK
061232  062702  000060          ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
061236  000753                  BR      1$              ;;GO PUT IT IN THE DATA TABLE
061240  000016          $OCTVL: .REPT   14.             ;;RESERVE DATA TABLE
061244          $OCT8=$OCTVL+4                          ;POINTER TO 11 DIGIT NUMBER
```

```
                                        .SBTTL  TABLES
                                        .SBTTL  APT MAILBOX-ETABLE
061256                          $MAIL:
061256  000000                  $MSGTY: .WORD   0       ;;MESSAGE TYPE CODE
061260  000000                  $FATAL: .WORD   0       ;;FATAL ERROR NUMBER (ERROR PC)
061262  000000                  $TESTN: .WORD   0       ;;TEST PATTERN NUMBER
061264  000000                  $PASS:  .WORD   0       ;;PASS COUNT
061266  000000                  $DEVCT: .WORD   0       ;;DEVICE COUNT
061270  000000                  $UNIT:  .WORD   0       ;;I/O UNIT NUMBER
061272  000000                  $MSGAD: .WORD   0       ;;MESSAGE ADDRESS
061274  000000                  $MSGLG: .WORD   0       ;;MESSAGE LENGTH
061276                          $ETABLE:                ;;APT ENVIRONMENT TABLE
061276      000                 $ENV:   .BYTE   0       ;;ENVIRONMENT BYTE      ;SET TO A 1 FOR APT AUTO MODE
                                ;NOTE:  IF BIT #7 IS SET IN $ENVM THE TABLE BELOW (BEGINNING AT $MAMS1 AND
                                ;       ENDING AT $MADR4) MUST BE FILLED IN TO INDICATE THE PROPER AMOUNT OF
                                ;       EACH TYPE OF MEMORY.
061277      000                 $ENVM:  .BYTE   0       ;ENVIRONMENT MODE
                                                        ;BIT7(200)=USE APT SIZE INFO ;BIT5(40)=NO CONSOLE
061300  000101                  $SWREG: .WORD   101     ;;APT SWITCH REGISTER
061302  000000                  $USWR:  .WORD   0       ;USED TO LIMIT THE NUMBER OF PASSES
061304  000000                  $CPUOP: .WORD   0       ;;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
061306      001                 $MAMS1: .BYTE   1       ;;HIGH ADDRESS,M.S. BYTE ;DEFAULT = 64K
061307      004                 $MTYP1: .BYTE   4       ;;MEM. TYPE,BLK#1
                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
                                ;*                              900 NSEC CORE=001
                                ;*                              300 NSEC BIPOLAR=002
                                ;*                              PARITY MOS=003
                                ;*                              ERROR CORRECTING MOS=004
061310  177776                  $MADR1: .WORD   177776  ;;HIGH ADDRESS,BLK#1
                                ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE'' ABOVE
061312      000                 $MAMS2: .BYTE   0       ;;HIGH ADDRESS,M.S. BYTE
061313      000                 $MTYP2: .BYTE   0       ;;MEM.TYPE,BLK#2
061314  000000                  $MADR2: .WORD   0       ;;MEM.LAST ADDRESS,BLK#2
061316      000                 $MAMS3: .BYTE   0       ;;HIGH ADDRESS,M.S.BYTE
061317      000                 $MTYP3: .BYTE   0       ;;MEM.TYPE,BLK#3
061320  000000                  $MADR3: .WORD   0       ;;MEM.LAST ADDRESS,BLK#3
061322      000                 $MAMS4: .BYTE   0       ;;HIGH ADDRESS,M.S.BYTE
061323      000                 $MTYP4: .BYTE   0       ;;MEM.TYPE,BLK#4
061324  000000                  $MADR4: .WORD   0       ;;MEM.LAST ADDRESS,BLK#4
061326  000000                  $VECT1: .WORD   0       ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
061330  000000                  $VECT2: .WORD   0       ;;INTERRUPT VECTOR#2BUS PRIORITY#2
061332  000000                  $BASE:  .WORD   0       ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
061334  000000                  $DEVM:  .WORD   0       ;;DEVICE MAP

061336  000000                  $CDW1:  .WORD   0
061340  000000                  $CDW2:  .WORD   0
```

```
 11265                                  ;THE FOLLOWING LOCATIONS SPECIFY WHICH PATTERNS
 11266                                  ;ARE TO BE RUN FOR PARTICULAR MEMORIES
 11267                                  ;
 11268                                  ;REFERENCE THE TABLE LISTED BELOW TO RELATE BITS TO PATTERNS.
 11269                                  ;BIT0 SET WILL RUN THE FIRST ENTRY IN THE TABLE, BIT0 SET
 11270                                  ;IN THE SECOND WORD WILL RUN THE 17TH ENTRY IN THE TABLE ...
 11271                                  ;
 11272                                  ;NOTE** NULL TESTS DO NOT TAKE ANY TIME
 11273                                  ;                                        ;FIELD SERVICE VALUE
 11274 061342  177777                   $DDW0:  .WORD   177777  ;ECC CSR TESTS          177777          TABLE = MKCSRT:
 11275 061344  177777                   $DDW1:  .WORD   177777  ;ECC CSR TESTS          177777          TABLE = MKCSRT:
 11276 061346  177777                   $DDW2:  .WORD   177777  ;ECC PATTERNS           103777          TABLE = MKPAT:
 11277 061350  177777                   $DDW3:  .WORD   177777  ;ECC PATTERNS           177777          TABLE = MKPAT:
 11278 061352  177777                   $DDW4:  .WORD   177777  ;PARITY PATTERNS        003777          TABLE = MJPAT:
 11279 061354  177777                   $DDW5:  .WORD   177777  ;PARITY PATTERNS        177774          TABLE = MJPAT:
 11283 061356                           $ETEND:
 11284                                  ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
 11285                                  ;INTERFACE SPEC.
 11286
 11287 061356                           $APTHD:
 11288 061356  000000                   $HIBTS: .WORD   0               ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
 11289 061360  061256                   $MBADR: .WORD   $MAIL           ;;ADDRESS OF APT MAILBOX (BITS 0-15)
 11290 061362  000043                   $TSTM:  .WORD   35.             ;;RUN TIM OF LONGEST TEST
 11291 061364  001274                   $PASTM: .WORD   700.            ;;RUN TIME IN SECS. OF 1ST PASS ON 128K (QUICK VERIFY)
 11292 061366  000000                   $UNITM: .WORD   0.              ;;EXTRA RUN TIME OF A PASS FOR EACH ADDITIONAL 128K (QV)
 11293 061370  000040                           .WORD   $ETEND-$MAIL/2  ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
                                        .SBTTL  ROUTINE 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.

061372  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
061374  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
061400  005740                          TST     -(R0)           ;;BACKUP BY 2
061402  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
061404  006300                          ASL     R0              ;;POSITION FOR INDEXING
061406  016000  061434                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
061412  000200                          RTS     R0              ;;GO TO ROUTINE

                                ;;THIS IS USE TO HANDLE THE ''GETPRI'' MACRO

061414  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
061416  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
061424  000002                          RTI                     ;;RESTORE THE PSW

061426                          $NOTRAP:TYPE    MSG006          ;UNDEFINED TRAP INSTRUCTION
061432  000000                  $HALT2: HALT
```

```
11322                                           .SBTTL  TRAP TABLE
11323
11324                                   ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
11325                                   ;*BY THE 'TRAP'' INSTRUCTION.
11326
11327                                   ;       ROUTINE
11328                                   ;       -------
11329 061434  061414                   $TRPAD: .WORD   $TRAP2
11330 061436  052240                           $TYPE   ;CALL=TYPEIT    TRAP+1(104401)  TTY TYPEOUT ROUTINE
11331 061440  056546                           $TYPOC  ;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
11332 061442  056522                           $TYPOS  ;CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
11333 061444  061426                           $NOTRAP;$TYPON  ;CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
11334 061446  056750                           $TYPDS  ;CALL=TYPDS     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
11335 061450  061426                           $NOTRAP;$TYPBN  ;CALL=TYPBN     TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER
11336
11337 061452  057340                           $GTSWR  ;CALL=GTSWR     TRAP+7(104407)  GET SOFT-SWR SETTING
11338 061454  057174                           $CKSWR  ;CALL=CKSWR     TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
11339
11340 061456  057704                           $RDCHR  ;CALL=RDCHR     TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
11341 061460  060024                           $RDLIN  ;CALL=RDLIN     TRAP+12(104412) TTY TYPEIN STRING ROUTINE
11342 061462  060410                           $RDOCT  ;CALL=RDOCT     TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
11343 061464  060560                           $RDDEC  ;CALL=RDDEC     TRAP+14(104414) READ A DECIMAL NUMBER FROM TTY
11344
11345 061466  060746                           $SAVREG ;CALL=SAVREG    TRAP+15(104415) SAVE R0-R5 ROUTINE
11346 061470  061004                           $RESREG ;CALL=RESREG    TRAP+16(104406) RESTORE R0-R5 ROUTIN.
11347
11348 061472  036656                           $KERNEL  ;CALL=KERNEL   TRAP+17(104417) ENTER KERNEL MODE
11349 061474  036666                           $ENERGIZE;CALL=ENERGIZETRAP+20(104420) TURN ON MEMORY MANAGEMENT & TRAPS
11350 061476  036676                           $DEENFRGI;CALL=DEENERGITRAP+21(104421) TURN OFF MEMORY MENAGEMENT & TRAPS
11351
11352 061500  041062                           $KMAP    ;CALL=KMAP     TRAP+22(104422) MAP KERNEL 1 TO 1
11353
11354 061502  036706                           $CACHN  ;CALL=CACHON    TRAP+23(104423) TURN CACHE ON
11355 061504  036732                           $CACHF  ;CALL=CACHOFF   TRAP+24(104424) TURN CACHE OFF
11356
11357 061506  036750                           $LOADC  ;CALL=LOADCSR   TRAP+25(104425) LOAD CORRECT CSR
11358 061510  037044                           $READC  ;CALL=READCSR   TRAP+26(104426) READ CORRECT CSR
11359
11360 061512  052474                           $PER01  ;CALL=PERR01    TRAP+27(104427) PROGRAM DETECTED ERROR
11361 061514  052522                           $PER02  ;CALL=PERR02    TRAP+30(104430) PROGRAM DETECTED ERROR
11362 061516  052550                           $PER03  ;CALL=PERR03    TRAP+31(104431) PROGRAM DETECTED ERROR
11363 061520  052600                           $PER04  ;CALL=PERR04    TRAP+32(104432) PROGRAM DETECTED ERROR
11364 061522  052662                           $PER07  ;CALL=PERR07    TRAP+33(104433) PROGRAM DETECTED ERROR
11365 061524  052704                           $PER10  ;CALL=PERR10    TRAP+34(104434) PROGRAM DETECTED ERROR
11366 061526  052734                           $PER11  ;CALL=PERR11    TRAP+35(104435) PROGRAM DETECTED ERROR
11367 061530  052754                           $PER12  ;CALL=PERR12    TRAP+36(104436) PROGRAM DETECTED ERROR
11368 061532  052776                           $PER13  ;CALL=PERR13    TRAP+37(104437) PROGRAM DETECTED ERROR
11369 061534  053016                           $PER14  ;CALL=PERR14    TRAP+40(104440) PROGRAM DETECTED ERROR
11370 061536  053040                           $PER15  ;CALL=PERR15    TRAP+41(104441) PROGRAM DETECTED ERROR
11371 061540  053062                           $PER16  ;CALL=PERR16    TRAP+42(104442) PROGRAM DETECTED ERROR
11372 061542  053102                           $PER17  ;CALL=PERR17    TRAP+43(104443) PROGRAM DETECTED ERROR
11373 061544  053120                           $PER20  ;CALL=PERR20    TRAP+44(104444) PROGRAM DETECTED ERROR
11374 061546  053136                           $PER21  ;CALL=PERR21    TRAP+45(104445) PROGRAM DETECTED ERROR
11375 061550  053156                           $PER22  ;CALL=PERR22    TRAP+46(104446) PROGRAM DETECTED ERROR
11376 061552  053174                           $PER23  ;CALL=PERR23    TRAP+47(104447) PROGRAM DETECTED ERROR
11377 061554  053212                           $PER24  ;CALL=PERR24    TRAP+50(104450) PROGRAM DETECTED ERROR
11378 061556  050054                           $PER25  ;CALL=PERR25    TRAP+51(104451) PROGRAM DETECTED ERROR
```

```
 11379 061560  053402                $PER26  ;CALL=PERR26     TRAP+52(104452) PROGRAM DETECTED ERROR
 11380 061562  053422                $PER27  ;CALL=PERR27     TRAP+53(104453) PROGRAM DETECTED ERROR
 11381 061564  050302                $PER30  ;CALL=PERR30     TRAP+54(104454) PROGRAM DETECTED ERROR
 11382 061566  053612                $PER31  ;CALL=PERR31     TRAP+55(104455) PROGRAM DETECTED ERROR
 11383 061570  053710                $PER32  ;CALL=PERR32     TRAP+56(104456) PROGRAM DETECTED ERROR
 11384 061572  053756                $PER33  ;CALL=PERR33     TRAP+57(104457) PROGRAM DETECTED ERROR
 11385 061574  054036                $PER34  ;CALL=PERR34     TRAP+60(104460) PROGRAM DETECTED ERROR
 11386 061576  054070                $PER35  ;CALL=PERR35     TRAP+61(104461) PROGRAM DETECTED ERROR
 11387 061600  054124                $PER36  ;CALL=PERR36     TRAP+62(104462) PROGRAM DETECTED ERROR
 11388 061602  061426                $NOTRAP ;CALL=PERR37     TRAP+63(104463) PROGRAM DETECTED ERROR
 11389 061604  061426                $NOTRAP ;CALL=PERR40     TRAP+64(104464) PROGRAM DETECTED ERROR
 11390 061606  061426                $NOTRAP ;CALL=PERR41     TRAP+65(104465) PROGRAM DETECTED ERROR
 11391 061610  061426                $NOTRAP ;CALL=PERR42     TRAP+66(104466) PROGRAM DETECTED ERROR
 11392 061612  061426                $NOTRAP ;CALL=PERR43     TRAP+67(104467) PROGRAM DETECTED ERROR
 11393
 11394 061614  037266                $ECCDIS ;CALL=ECCDIS     TRAP+70(104470) DISABLE ECC ON ALL CSR'S
 11395 061616  037302                $ECC1DIS;CALL=ECC1DIS    TRAP+71(104471) DISABLE ECC ON 1 SELECTED CSR
 11396 061620  037314                $ECCINIT;CALL=ECCINIT    TRAP+72(104472) INITIALIZE ALL MK11 CSR'S
 11397 061622  037330                $ECC1INIT;CALL=ECC1INIT  TRAP+73(104473) INITIALIZE 1 SELECTED MK11 CSR
 11398 061624  037370                $CBCSR  ;CALL=CBCSR      TRAP+74(104474) WRITE GENERATED CHECKBITS IN ALL CSR'S
 11399 061626  037412                $CB1CSR ;CALL=CB1CSR     TRAP+75(104475) WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
 11400 061630  037432                $WASSBE ;CALL=WASSBE     TRAP+76(104476) WAS THERE A SBE ON ANY CSR?
 11401 061632  037546                $WAS1SBE;CALL=WAS1SBE    TRAP+77(104477) WAS THERE A SBE ON 1 SELECTED CSR?
 11402 061634  037576                $WASDBE ;CALL=WASDBE     TRAP+100(104500)WAS THERE A DBE ON ANY CSR?
 11403 061636  037712                $WAS1DBE;CALL=WAS1DBE    TRAP+101(104501)WAS THERE A DBE ON 1 SELECTED CSR?
 11404 061640  037742                $CLRCSR ;CALL=CLRCSR     TRAP+102(104502)CLEAR ALL CSR'S
 11405 061642  037754                $CLR1CSR;CALL=CLR1CSR    TRAP+103(104503)CLEAR 1 SELECTED CSR
 11406 061644  037764                $CHKDIS ;CALL=CHKDIS     TRAP+104(104504)DISABLE ECC & WRITE CKBITS FROM ALL CSR'S
 11407 061646  040000                $CHK1DIS;CALL=CHK1DIS    TRAP+105(104505)DISABLE ECC & WRITE CKBITS FROM 1 CSR
 11408 061650  037342                $ENASBE ;CALL=ENASBE     TRAP+106(104506)ENABLE TRAPS ON SBE'S FROM ALL CSR'S
 11409 061652  037356                $ENA1SBE;CALL=ENA1SBE    TRAP+107(104507)ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
 11410 061654  037064                $TSTRD  ;CALL=TSTREAD    TRAP+110(104510)TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES
 11411 061656  040060                $INVALID;CALL=INVALID    TRAP+111(104511)INVALIDATE BACKGROUND PATTERN ON BANK
 11412 061660  040110                $ERRGEN ;CALL=ERRGEN     TRAP+114(104512)TEST ERROR ADDRESS
 11413 061662  061426                $NOTRAP
 11414 061664  061426                $NOTRAP
 11415 061666  061426                $NOTRAP
 11416 061670  061426                $NOTRAP
 11417 061672  061426                $NOTRAP
 11418 061674  061426                $NOTRAP
 11419 061676  061426                $NOTRAP
```

```
                                        .SBTTL ODT (CZMSD) DATA AREA
11423   177776          ST      =       177776          ;STATUS REGISTER
11424   000016          O.BKP   =       16              ;NUMBER OF BREAKPOINTS-1 MULT. BY 2
11425   000014          O.TVEC  =       14              ;TRT VECTOR LOCATION
11426   000340          O.STM   =       340             ;PRIORITY MASK - STATUS REGISTER
11427   000020          O.TBT   =       20              ;T-BIT MASK - STATUS REGISTER
11428   000003          TRT     =       000003          ;TRT INSTRUCTION
11429   177562          O.RDB   =       177562          ;R DATA BUFFER
11430   177560          O.RCSR  =       177560          ;R C/SR
11431   177566          O.TDB   =       177566          ;T DATA BUFFER
11432   177564          O.TCSR  =       177564          ;T C/SR
```

```
11435 061700  000000                O.CAD:  0                       ; CURRENT ADDRESS
11436 061702  000000                O.DOT:  0                       ; ORIGIN ADDRESS
11437 061704  000000                O.XXX:  .WORD   0               ;TEMPORARY STORAGE
11438 061706  000000                O.BW:   .WORD   0               ; =0 - ALL CLOSED
11439                                                               ; =1 - BYTE OPEN,
11440                                                               ; =2 - WORD OPEN
11441 061710     000                O.SEQ:  .BYTE   0               ;CHANGE SEQUENCE INDICATOR
11442 061711     000                O.S:    .BYTE   0               ;SINGLE INSTRUCTION FLAG
11443                                                               ;0 IF NOT ACTIVE
11444                                                               ;-1 IF ACTIVE
11445                                                               ;NO BREAK POINTS MAY BE SET WHILE IN
11446                                                               ;SINGLE INSTRUCTION MODE
11447 061712     000                O.T:    .BYTE   0               ;  T-BIT FLAG
11448 061713     000                O.P:    .BYTE   0               ;PROCEED FLAG = -2 IF MANUAL ENTRY
11449                                                               ;               -1 IF NO PROCEED ALLOWED
11450                                                               ;               0-7 IF PROCEED ALLOWED
11451 061714     000                O.CSR1: .BYTE   0               ;SAVE CELL - R C/SR
11452 061715     000                O.CSR2: .BYTE   0               ;SAVE CELL - T C/SR
11453 061716  000000                O.FIL:  0                       ;FILL COUNTER
11454 061720     000                O.CMFD: .BYTE   0               ;COMMA FOUND SWITCH, =0 NO COMMA FOUND
11455                                                               ;                    =1 COMMA FOUND
11456 061721     000                O.SMFD: .BYTE   0               ;SEMICOLON FOUND SWITCH
11457                                                               ;=0 NO SEMICOLON FOUND
11458                                                               ;=1 SEMICOLON FOUND
11459 061722     000                O.SCRN: .BYTE   0               ;FLAG; 1=PASS SPACES ON FROM TTY
11460                                                               ;ALSO, IF =1, <LF> IS ECHOED
11461 061723     000                O.MINS: .BYTE   0               ;MINUS SIGN TYPED  (SWITCH)
11462                                                               ;0=NO MINUS TYPED;  1=MINUS SIGN TYPED
11463 061724  000000                O.BIAS: .WORD   0               ;CURRENT RELOCATION BIAS
```

```
11466                                   ;THE ORDER OF THE FOLLOWING ENTRIES IS CRITICAL
11467           000026                          .REPT   26
11470 062002    000000                  O.UR0:  0       ;USER R0
11471 062004    000000                          0       ;      R1
11472 062006    000000                          0       ;      R2
11473 062010    000000                          0       ;      R3
11474 062012    000000                          0       ;      R4
11475 062014    000000                          0       ;      R5
11476 062016    000000                  O.USP:  0       ;USER SP
11477 062020    000000                  O.UPC:  0       ;USER PC
11478 062022    000000                  O.UST:  0       ;USER ST
11479 062024    000007                  O.PRI:  7       ;ODT PRIORITY
11480                                   ; BREAK POINT LISTS, ADR1 = ADDRESS OF BREAKPOINT,CT = COUNT,
11481                                   ;   UIN = CONTENTS
11482
11483 062026    000000                  O.ADR1: .WORD 0
11484 062030    000000                          .WORD 0
11485 062032    000000                          .WORD 0
11486 062034    000000                          .WORD 0
11487 062036    000000                          .WORD 0
11488 062040    000000                          .WORD 0
11489 062042    000000                          .WORD 0
11490 062044    000000                          .WORD 0
11491 062046    000000                          .WORD 0
11492 062050    000000                  O.CT:   .WORD 0
11493 062052    000000                          .WORD 0
11494 062054    000000                          .WORD 0
11495 062056    000000                          .WORD 0
11496 062060    000000                          .WORD 0
11497 062062    000000                          .WORD 0
11498 062064    000000                          .WORD 0
11499 062066    000000                          .WORD 0
11500 062070    000000                          .WORD 0
11501 062072    000000                  O.UIN:  .WORD 0
11502 062074    000000                          .WORD 0
11503 062076    000000                          .WORD 0
11504 062100    000000                          .WORD 0
11505 062102    000000                          .WORD 0
11506 062104    000000                          .WORD 0
11507 062106    000000                          .WORD 0
11508 062110    000000                          .WORD 0
11509 062112    000000                          .WORD 0
```

```
11515                                   ; INITIALIZE ODT
11516                                   ;  USE 0.ODT FOR A NORMAL ENTRY
11517                                   ;  USE 0.ODT+2 TO RESTART ODT - WIPING OUT ALL BREAKPOINTS
11518                                   ;  USE 0.ODT+4 TO RE-ENTER (I.E. - FAKE A BREAKPOINT)
11519
11520 062114  000421                  0.ODT:  BR      2$              ;NORMAL ENTRY
11521 062116  000440                          BR      3$              ;RESTART
11522 062120  004737  065312          1$:     JSR     PC,0.RRST       ;RE-ENTER  --  SAVE STATUS
11523 062124  013746  000016                  MOV     0.TVEC+2,-(SP)  ;SET UP LOCAL STATUS
11524 062130  004737  065322                  JSR     PC,0.WST
11525 062134  010746                          MOV     PC,-(SP)        ;FAKE THE PC
11526 062136  012637  062020                  MOV     (SP)+,0.UPC
11527 062142  112737  177777  061713          MOVB    #-1,0.P         ;DISALLOW PROCEED
11528 062150  105037  061711                  CLRB    0.S
11529 062154  000137  064200                  JMP     0.BK1
11530
11531                                           ;RUN AT CURRENT STATUS
11532 062160  004537  064620          2$:     JSR     5,0.SVTT        ;SAVE TTY STATUS
11533 062164  012737  062120  057254          MOV     #1$,@#CONTP+2
11534 062172  004037  064500                  JSR     0,0.SVR         ;SAVE REGISTERS (MAINLY SP)
11535 062176  012706  062002                  MOV     #0.UR0,SP       ;SET UP STACK
11536 062202  012704  065420                  MOV     #0.ID,R4        ;TYPE ID
11537 062206  012703  065433                  MOV     #0.IDND,R3
11538 062212  004537  065246                  JSR     5,0.TYPE
11539 062216  000414                          BR      4$
11540 062220  004037  064500          3$:     JSR     0,0.SVR         ;SAVE REGISTERS
11541 062224  004537  064722                  JSR     5,0.REM         ;REMOVE ALL BREAKPOINTS
11542 062230  113704  062024                  MOVB    0.PRI,R4        ;GET ODT PRIORITY
11543 062234  106004                          RORB    R4              ;SHIFT
11544 062236  106004                          RORB    R4              ;  INTO
11545 062240  106004                          RORB    R4              ;   POSITION
11546 062242  010446                          MOV     R4,-(SP)        ;STORE IN STATUS
11547 062244  004737  065322                  JSR     PC,0.WST
11548 062250  105037  061711          4$:     CLRB    0.S             ;DISABLE SINGLE INSTRUCTION FOR NOW
11549 062254  112737  177777  061713          MOVB    #-1,0.P         ;DISALLOW PROCEED
11550 062262  012737  000340  000016          MOV     #0.STM,0.TVEC+2 ;STATUS WORD TO TRT VECTOR+2
11551 062270  012737  064162  000014          MOV     #0.BRK,0.TVEC   ;PC TO TRT VECTOR
11552 062276  000137  063550                  JMP     0.RALL          ;CLEAR BRK PT TABLES
11553                                   ;
11554                                   ;0.CTLC
11555                                   ;       ^C PROCESSING. RETURN TO KEYBOARD MONITOR
11556                                   ;
11557                                   ;
11558 062302  013706  062016          0.CTLC: MOV     0.USP,SP        ;RESTORE USER STACK
11559 062306  012704  062326                  MOV     #1$,R4
11560 062312  012703  062327                  MOV     #2$,R3
11561 062316  004537  065246                  JSR     R5,0.TYPE       ;ECHO '^C'
11562 062322  000137  043426                  JMP     BOOT
11563 062326     136                  1$:     .BYTE   '^
11564 062327     103                  2$:     .BYTE   'C
```

```
11567                                   ;CONTROL F PROCESSING
11568 062330  013706  062016            O.CTLF: MOV     O.USP,SP        ;RESTORE USER STACK
11569 062334  012704  062360                    MOV     #1$,R4
11570 062340  012703  062361                    MOV     #2$,R3
11571 062344  004537  065246                    JSR     R5,O.TYPE       ;ECHO ^F
11572 062350  004737  044004                    CALL    FIELDSERVICE
11573 062354  000137  062114                    JMP     O.ODT
11574 062360     136                    1$:     .BYTE   '^
11575 062361     106                    2$:     .BYTE   'F
11576
11577                                   ;CONTROL E PROCESSING
11578 062362  012704  062406            O.CTLE: MOV     #1$,R4
11579 062366  012703  062407                    MOV     #2$,R3
11580 062372  004537  065246                    JSR     R5,O.TYPE       ;ECHO ^E
11581 062376  012705  057256                    MOV     #CONTP+4,R5
11582 062402  000137  063744                    JMP     O.GOGO
11583 062406     136                    1$:     .BYTE   '^
11584 062407     105                    2$:     .BYTE   'E
11585
11586                                   ; SPECIAL NAME HANDLER
11587                                   ;  DEPENDS UPON THE EXPLICIT ORDER OF THE TWO TABLES O.TL AND O.URO
11588
11589 062410  004537  065126            O.REGT: JSR     5,O.GET         ;SPECIAL NAME, GET ONE MORE CHARACTER
11590 062414  012704  065466                    MOV     #O.TL,R4        ;TABLE START ADDRESS
11591 062420  120024                    1$:     CMPB    R0,(R4)+        ;IS THIS THE CORRECT CHARACTER?
11592 062422  001414                            BEQ     3$              ;JUMP IF YES
11593 062424  022704  065506                    CMP     #O.TL+O.LG,R4   ;IS THE SEARCH DONE?
11594 062430  101373                            BHI     1$              ;BRANCH IF NOT
11595 062432  042700  177770                    BIC     #177770,R0      ;MASK OFF OCTAL
11596 062436  010004                            MOV     R0,R4
11597 062440  006304                    2$:     ASL     R4
11598 062442  062704  062002                    ADD     #O.URO,R4       ;GENERATE ADDRESS
11599 062446  005202                            INC     R2              ;SET FOUND FLAG
11600 062450  000137  062674                    JMP     O.SCAN          ;GO FIND NEXT CHARACTER
11601 062454  162704  065457            3$:     SUB     #O.TL-7,R4      ;GO FIND NEXT CHARACTER
11602 062460  000767                            BR      2$
```

```
11605                                   ; 'BACKARROW'' HANDLER - OPEN INDEXED ON THE PC (BACK ARROW)
11606                                           .ENABL  LSB
11607 062462  004537  062536          O.ORPC: JSR     5,2$            ;TEST WORD MODE AND CLOSE
11608 062466  061202                          ADD     @R2,R2          ;COMPUTE
11609 062470  005202                          INC     R2
11610 062472  005202                          INC     R2              ;   NEW ADDRESS
11611 062474  010237  061700          1$:     MOV     R2,O.CAD        ;UPDATE CAD
11612 062500  000137  063346                  JMP     O.OP2A          ;GO FINISH UP
11613 062504  004537  062536          O.ORAB: JSR     5,2$            ;TEST WORD MODE AND CLOSE
11614 062510  011202                          MOV     @R2,R2          ;GET ABSOLUTE ADDRESS
11615 062512  000770                          BR      1$
11616 062514  004537  062536          O.ORRB: JSR     5,2$            ;TEST AND CLOSE
11617 062520  011201                          MOV     @R2,R1          ;COMPUTE NEW ADDRESS
11618 062522  110101                          MOVB    R1,R1           ;EXTEND THE SIGN
11619 062524  006301                          ASL     R1              ;R2=2(@R2)
11620 062526  005201                          INC     R1              ;   +2
11621 062530  005201                          INC     R1
11622 062532  060102                          ADD     R1,R2           ;   +PC
11623 062534  000757                          BR      1$
11624 062536  004737  065334          2$:     JSR     PC,O.CLSE       ;CLOSE CURRENT CELL
11625 062542  022737  000002  061706          CMP     #2,O.BW         ;ONLY WORD MODE ALLOWED
11626 062550  001003                          BNE     3$              ;BRANCH IF ERROR
11627 062552  013702  061700                  MOV     O.CAD,R2        ;CURRENT ADDRESS IN R2
11628 062556  000205                          RTS     R5
11629 062560  005726                  3$:     TST     (SP)+
11630 062562  000137  062630          O.TCL2: JMP     O.ERR           ;POP A WORD AND SHOW THE ERROR
11631                                           .DSABL  LSB
11632
11633                                   ;       PROCESS S - SINGLE INSTRUCTION MODE
11634 062566  105737  061721          O.SNGL: TSTB    O.SMFD          ;DONT REACT IF ; NOT TYPED
11635 062572  001773                          BEQ     O.TCL2
11636 062574  005702                          TST     R2              ;SEE IF TURN ON OR TURN OFF
11637 062576  001004                          BNE     1$              ;BRANCH IF TURNING IT ON
11638 062600  105037  061711                  CLRB    O.S             ;CLEAR THE FLAG
11639 062604  000137  062640                  JMP     O.DCD           ;CONTINUE THE SCAN
11640 062610  112737  177777  061711  1$:     MOVB    #-1,O.S         ;SET THE FLAG
11641 062616  000137  062640                  JMP     O.DCD
```

```
11644                                   ;
11645                                   ;MINUS PROCESSING
11646                                   ;
11647 062622  105237  061723          O.MIN:  INCB    O.MINS          ;SET MINUS FOUND SWITCH ON
11648 062626  000420                          BR      O.DCD1
11649
11650                                   ; COMMAND DECODER - ODT11X
11651
11652                                   ;  ALL REGISTERS MAY BE USED (R0-R5),
11653
11654 062630  012700  000077          O.ERR:  MOV     #'?,R0          ;  ? TO BE TYPED
11655 062634  004537  065050                  JSR     5,O.FTYP        ; OUTPUT ?
11656 062640  005037  061706          O.DCD:  CLR     O.BW            ;CLOSE ALL
11657 062644  004537  065374                  JSR     5,O.CRLS        ;TYPE  <CR><LF>*
11658 062650  105037  061721          O.DCD3: CLRB    O.SMFD          ;SEMICOLON FOUND FLAG
11659 062654  105037  061720                  CLRB    O.CMFD          ;COMMA FOUND FLAG
11660 062660  105037  061723                  CLRB    O.MINS          ;MINUS SIGN FOUND FLAG
11661 062664  005003                          CLR     R3              ;R3 IS A SAVE REGISTER FOR R2
11662 062666  005005                          CLR     R5              ;R5 IS A SAVE REGISTER FOR R4
11663 062670  005004          O.DCD1: CLR     R4              ; R4 CONTAINS THE CONVERTED OCTAL
11664 062672  005002                          CLR     R2              ; R2 IS THE NUMBER FOUND FLAG
11665 062674  004537  065126          O.SCAN: JSR     5,O.GET         ;GET A CHAR, RETURN IN R0
11666 062700  022700  000060                  CMP     #'0,R0          ;COMPARE WITH ASCII 0
11667 062704  101013                          BHI     5$              ;CHECK LEGALITY IF NON-NUMERIC
11668 062706  022700  000067                  CMP     #'7,R0          ;COMPARE WITH ASCII 7
11669 062712  103410                          BLO     5$              ;CHECK LEGALITY IF NOT OCTAL
11670 062714  042700  177770                  BIC     #177770,R0      ;CONVERT TO BCD
11671 062720  006304                          ASL     R4              ; MAKE ROOM
11672 062722  006304                          ASL     R4              ;  IN
11673 062724  006304                          ASL     R4              ;    R4
11674 062726  060004                          ADD     R0,R4           ;PACK THREE BITS IN R4
11675 062730  005202                          INC     R2              ;R2 HAS NUMERIC FLAG
11676 062732  000760                          BR      O.SCAN          ;   AND TRY AGAIN
11677 062734  005001                  5$:     CLR     R1              ;CLEAR INDEX
11678 062736  120061  065442          1$:     CMPB    R0,O.LGCH(R1)   ;DO THE CODES MATCH?
11679 062742  001405                          BEQ     2$              ;JUMP IF YES
11680 062744  005201                          INC     R1              ; SET INDEX FOR NEXT SEARCH
11681 062746  020127  000024                  CMP     R1,#O.CLGT      ;IS THE SEARCH DONE?
11682 062752  103326                          BHIS    O.ERR           ;    OOPS!
11683 062754  000770                          BR      1$              ;RE-LOOP
11684 062756  105737  061723          2$:     TSTB    O.MINS          ;IF MINUS WAS NOT TYPED
11685 062762  001401                          BEQ     4$              ;DO NOT NEGATE K
11686 062764  005404                          NEG     R4              ;OTHERWISE, TAKE 2'S COMPLEMENT.
11687 062766  105737  061720          4$:     TSTB    O.CMFD          ;IF A COMMA NOT TYPED,
11688 062772  001402                          BEQ     3$              ;SKIP NEXT INSTRUCTION.
11689 062774  063704  061724                  ADD     O.BIAS,R4       ;OTHERWISE, ADD RELOC. BIAS TO (R4)
11690 063000  105037  061723          3$:     CLRB    O.MINS          ;REINITIALIZE MINUS-TYPED SWITCH FOR NXT SCAN
11691 063004  006301                          ASL     R1              ;MULTIPLY BY TWO
11692 063006  000171  063012                  JMP     @6$(R1)         ;GO TO PROPER ROUTINE
```

```
  11695 063012  063062                6$:     O.SEMI  ;  ; SEMICOLON
  11696 063014  063100                        O.WRD   ;  /    OPEN WORD
  11697 063016  063112                        O.BYT   ;  \    OPEN BYTE -BACK SLASH
  11698 063020  063260                        O.CRET  ;  CARRIAGE RETURN    CLOSE
  11699 063022  062410                        O.REGT  ;  $  REGISTER OPS
  11700 063024  063716                        O.GO    ;  G  GO TO ADDRESS K
  11701 063026  063300                        O.OP1   ;  <LF>  MODIFY, CLOSE, OPEN NEXT
  11702 063030  062462                        O.ORPC  ;  'BACKARROW''  OPEN RELATED, INDEX - PC (BACK ARROW)
  11703 063032  063274                        O.OLD   ;  <  RETURN TO OLD SEQUENCE AND OPEN
  11704 063034  063422                        O.BACK  ;  ^  OPEN PREVIOUS (UP ARROW)
  11705 063036  063606                        O.OFST  ;  O  OFFSET
  11706 063040  063444                        O.BKPT  ;  B  BREAKPOINTS
  11707 063042  064052                        O.PROC  ;  P  PROCEED
  11708 063044  062504                        O.ORAB  ;  @  OPEN RELATED, ABSOLUTE
  11709 063046  062514                        O.ORRB  ;  >  OPEN RELATED, REL. BRANCH
  11710 063050  062566                        O.SNGL  ;  S  SINGLE INSTRUCTION MODE
  11711 063052  062622                        O.MIN   ;  - MINUS, NEGATES NUMBER TYPED IN
  11712 063054  062302                        O.CTLC  ;  ^C EXIT TO MONITOR
  11713 063056  062330                        O.CTLF  ;  ^F EXIT TO FIELD SERVICE MODE
  11714 063060  062362                        O.CTLE  ;  ^E PROCEED FROM ^D
```

```
11717
11718                                   ; SEMI-COLON PROCESSOR
11719
11720 063062  010203                    0.SEMI: MOV     R2,R3           ;A SEMI-COLON HAS BEEN RECEIVED
11721 063064  010405                            MOV     R4,R5           ;NUMERIC FLAG TO R3, CONTENTS TO R5
11722 063066  105237  061721                    INCB    0.SMFD          ;SET SEMICOLON FOUND FLAG
11723 063072  105037  061720                    CLRB    0.CMFD          ;RESET COMMA FOUND FLAG
11724 063076  000674                            BR      0.DCD1          ;GO BACK FOR MORE
11725
11726                                   ; PROCESS / AND \ - OPEN WORD OR BYTE
11727
11728                                   ;INPUT - IF R2 IS NON-ZERO A NEW REXP HAS BEEN TYPED IN
11729                                   ;INPUT - -ADDRESS OF WORD TO BE OPENED IS IN R4
11730                                           .ENABL  LSB
11731 063100  012737  000002  061706    0.WRD:  MOV     #2,0.BW         ;OPEN WORD
11732 063106  000404                            BR      11$
11733 063110  006104                    1$:     ROL     R4              ;GET THE ADDRESS BACK
11734 063112  012737  000001  061706    0.BYT:  MOV     #1,0.BW         ;OPEN BYTE
11735 063120  005702                    11$:    TST     R2              ;GET VALUE IF R2 IS NON-ZERO
11736 063122  001011                            BNE     14$             ;BRANCH IF NUMBER INPUT
11737 063124  105737  061720                    TSTB    0.CMFD          ;TEST FOR ","AND";"
11738 063130  001402                            BEQ     12$
11739 063132  000137  062630            13$:    JMP     0.ERR           ;ERROR IF PRESENT WITHOUT NUMBER.
11740 063136  105737  061721            12$:    TSTB    0.SMFD
11741 063142  001373                            BNE     13$
11742 063144  000404                            BR      0.WRD1          ;NO NUMBER - REOPEN PREVIOUS LOCATION
11743 063146  010437  061702            14$:    MOV     R4,0.DOT        ;PUT VALUE IN DOT
11744 063152  010437  061700                    MOV     R4,0.CAD        ;    ALSO IN CAD
11745 063156  022737  000001  061706    0.WRD1: CMP     #1,0.BW         ;CHECK BYTE MODE
11746 063164  001407                            BEQ     22$             ;JUMP IF BYTE
11747 063166  013704  061700                    MOV     0.CAD,R4
11748 063172  006204                            ASR     R4              ;MOVE ONE BIT TO CARRY
11749 063174  103745                            BCS     1$              ;JUMP IF ODD ADDRESS
11750 063176  017700  176476                    MOV     @0.CAD,R0       ;GET CONTENTS OF WORD
11751 063202  000402                            BR      23$
11752 063204  117700  176470            22$:    MOVB    @0.CAD,R0       ;GET CONTENTS OF BYTE
11753 063210  004537  064754            23$:    JSR     5,0.CADV        ;GO GET AND TYPE OUT @CAD
11754 063214  022737  000001  061706            CMP     #1,0.BW         ;CHECK IF BYTE MODE.
11755 063222  001212                            BNE     0.DCD3          ;IF NOT WE'RE DONE. ELSE:
11756 063224  112700  000075                    MOVB    #'=,R0          ;TYP "=" AND THEN THE ASCII BYTE
11757 063230  004537  065050                    JSR     5,0.FTYP
11758 063234  117700  176440                    MOVB    @0.CAD,R0
11759 063240  004537  065050                    JSR     5,0.FTYP
11760 063244  112700  000040                    MOVB    #' ,R0
11761 063250  004537  065050                    JSR     5,0.FTYP
11762 063254  000137  062650                    JMP     0.DCD3          ;GO BACK TO DECODER
11763                                           .DSABL  LSB
```

```
11766                                   ; PROCESS CARRIAGE RETURN
11767 063260  004737  065334            O.CRET: JSR     PC,O.CLSE       ;CLOSE LOCATION
11768 063264  000137  062640            O.DCDA: JMP     O.DCD           ;RETURN TO DECODER
11769
11770 063270  000137  062630            O.ERR3: JMP     O.ERR           ; INTERMEDIATE HELP
11771
11772                                   ; PROCESS <LF>, OPEN NEXT WORD
11773
11774 063274  105237  061710            O.OLD:  INCB    O.SEQ           ;SET FLAG TO LATER RESTORE CAD
11775 063300  005737  061706            O.OP1:  TST     O.BW            ;<LF> RECEIVED
11776 063304  001771                    O.ERR2: BEQ     O.ERR3          ;ERROR IF NOTHING IS OPEN
11777 063306  004737  065334                    JSR     PC,O.CLSE       ;CLOSE PRESENT CELL
11778 063312  105737  061710                    TSTB    O.SEQ           ;SHOULD CAD BE RESTORED?
11779 063316  001405                            BEQ     1$              ;BRANCH IF NOT
11780 063320  013737  061702  061700            MOV     O.DOT,O.CAD     ;RESTORE PREVIOUS SEQUENCE
11781 063326  105037  061710                    CLRB    O.SEQ           ;RESET FLAG; NO LONGER NEEDED
11782 063332  063737  061706  061700  1$:       ADD     O.BW,O.CAD      ;GENERATE NEW ADDRESS
11783 063340  013737  061700  061702  O.OP2:   MOV     O.CAD,O.DOT     ;INITIALIZE DOT
11784 063346  004537  065366            O.OP2A: JSR     5,O.CRLF        ;<CR><LF>
 1785 063352  013746  061706                    MOV     O.BW,-(SP)      ;SAVE BW
11786 063356  012737  000002  061706            MOV     #2,O.BW         ;SET TO TYPE FULL WORD ADDRESS
11787 063364  013700  061700                    MOV     O.CAD,R0        ;NUMBER TO TYPE
11788 063370  004537  065412                    JSR     5,O.RORA        ; CHECK FORMAT
11789 063374  011637  061706                    MOV     @SP,O.BW        ;RESTORE BW
11790 063400  012700  027534                    MOV     #27534,R0       ;PUT "/\" IN R0
11791 063404  022726  000001                    CMP     #1,(SP)+        ;IS IT BYTE MODE?
11792 063410  001401                            BEQ     1$              ;JUMP IF YES
11793 063412  000300                            SWAB    R0              ;TYPE A /
11794 063414  004537  065050            1$:     JSR     5,O.FTYP        ;TYPE THE LOW BYTE OF R0
11795 063420  000656                            BR      O.WRD1          ;GO PROCESS IT
11796
```

11796

```
11799
11800                                   ; PROCESS ^, OPEN PREVIOUS WORD
11801
11802 063422  005737  061706          O.BACK: TST     O.BW            ;  ^ RECEIVED
11803 063426  001726                          BEQ     O.ERR2          ;ERROR IF NOTHING OPEN
11804 063430  004737  065334                  JSR     PC,O.CLSE
11805 063434  163737  061706  061700        SUB     O.BW,O.CAD      ;GENERATE NEW ADDRESS
11806 063442  000736                        BR      O.OP2           ;GO DO THE REST
11807
11808                                   ; B HANDLER - SET AND REMOVE BREAKPOINTS
11809
11810 063444  012700  065510          O.BKPT: MOV     #O.TRTC,R0
11811 063450  006304                          ASL     R4              ;MULTIPLY NUMBER BY TWO
11812 063452  005703                          TST     R3
11813 063454  001423                          BEQ     3$              ;IF R3 IS ZERO GO REMOVE BREAKPOINT
11814 063456  006205                          ASR     R5              ;GET ONE BIT TO CARRY
11815 063460  103514                          BCS     O.ERR1          ;BADNESS IF ODD ADDRESS
11816 063462  006305                          ASL     R5              ;RESTORE ONE BIT
11817 063464  062704  062026                  ADD     #O.ADR1,R4
11818 063470  005702                          TST     R2
11819 063472  001007                          BNE     1$              ;JUMP IF SPECIFIC CELL
11820 063474  020014                  2$:     CMP     R0,@R4          ;IS THIS CELL FREE?
11821 063476  001405                          BEQ     1$              ;JUMP IF YES
11822 063500  020427  062044                  CMP     R4,#O.BKP+O.ADR1;ARE WE AT THE END OF OUR ROPE
11823 063504  103102                          BHIS    O.ERR1          ;YES, THERE IS NOTHING FREE
11824 063506  005724                          TST     (R4)+           ;INCREMENT BY TWO
11825 063510  000771                          BR      2$
11826 063512  020427  062044          1$:     CMP     R4,#O.BKP+O.ADR1
11827 063516  101075                          BHI     O.ERR1          ;ERROR IF TOO LARGE
11828 063520  010514                          MOV     R5,@R4          ;SET BREAKPOINT
11829 063522  000660                          BR      O.DCDA          ;RETURN
11830 063524  005702                  3$:     TST     R2
11831 063526  001410                          BEQ     O.RALL          ;GO REMOVE ALL
11832 063530  020427  000016                  CMP     R4,#O.BKP
11833 063534  101066                          BHI     O.ERR1          ;JUMP IF NUMBER TOO LARGE
11834 063536  010064  062026                  MOV     R0,O.ADR1(R4)   ;CLEAR BREAKPOINT
11835 063542  005064  062050                  CLR     O.CT(R4)        ;CLEAR COUNT ALSO
11836 063546  000646                          BR      O.DCDA
11837 063550  005004                  O.RALL: CLR     R4
11838 063552  012700  065510                  MOV     #O.TRTC,R0
11839 063556  020427  000020          1$:     CMP     R4,#O.BKP+2     ;ALL DONE?
11840 063562  101240                          BHI     O.DCDA          ;JUMP IF YES
11841 063564  010064  062026                  MOV     R0,O.ADR1(R4)   ;RESET BKPT
11842 063570  012764  000003  062072          MOV     #TRT,O.UIN(R4)  ;RESET CONTENTS OF TABLE
11843 063576  005064  062050                  CLR     O.CT(R4)        ;CLEAR COUNT
11844 063602  005724                          TST     (R4)+           ;INCREMENT BY TWO
11845 063604  000764                          BR      1$
11846
```

```
11849
11850                                   ; PROCESS 0, COMPUTE OFFSET
11851
11852 063606  022737  000002  061706  0.OFST: CMP     #2,0.BW         ;CHECK WORD MODE
11853 063614  001036                          BNE     0.ERR1          ;ERROR IF NOT CORRECT MODE
11854 063616  012700  000040                  MOV     #' ,R0          ;TYPE ONE BLANK
11855 063622  004537  065050                  JSR     5,0.FTYP        ;  AS A SEPARATOR
11856 063626  005703                          TST     R3              ;WAS SEMI-COLON TYPED?
11857 063630  001430                          BEQ     0.ERR1          ;NO, CALL IT AN ERROR
11858 063632  163705  061700                  SUB     0.CAD,R5        ;COMPUTE
11859 063636  005305                          DEC     R5
11860 063640  005305                          DEC     R5              ;     16 BIT OFFSET
11861 063642  010500                          MOV     R5,R0
11862 063644  004537  064754                  JSR     5,0.CADV        ;NUMBER IN R0 - WORD MODE
11863 063650  010500                          MOV     R5,R0
11864 063652  006200                          ASR     R0              ;DIVIDE BY TWO
11865 063654  103414                          BCS     1$              ;ERROR IF ODD
11866 063656  022700  177600                  CMP     #-200,R0        ;COMPARE WITH -200
11867 063662  003011                          BGT     1$              ;DO NOT TYPE IF OUT OF RANGE
11868 063664  022700  000177                  CMP     #177,R0         ;COMPARE WITH +177
11869 063670  002406                          BLT     1$              ;DO NOT TYPE IF OUT OF RANGE
11870 063672  005337  061706                  DEC     0.BW            ;SET TEMPORARY BYTE MODE
11871 063676  004537  064754                  JSR     5,0.CADV        ;NUMBER IN R0 - BYTE MODE
11872 063702  005237  061706                  INC     0.BW            ;RESTORE WORD MODE
11873 063706  000137  062650          1$:     JMP     0.DCD3          ;ALL DONE
11874
11875 063712  000137  062630          0.ERR1: JMP     0.ERR           ;INTERMEDIATE HELP
```

```
11878                                   ; PROCESS G - GO
11879
11880 063716  105737  061721          O.GO:   TSTB    O.SMFD          ;WAS ":" TYPED?
11881 063722  001773                          BEQ     O.ERR1          ;BR IF NOT TYPED
11882 063724  005703                          TST     R3              ;WAS K; TYPED?
11883 063726  001771                          BEQ     O.ERR1          ; TYPE ?<CR,LF> IF NOT
11884 063730  112737  000021  061713          MOVB    #O.BKP+3,O.P    ;CLEAR PROCEED
11885 063736  006205                          ASR     R5              ;CHECK LOW ORDER BIT
11886 063740  103764                          BCS     O.ERR1          ;ERROR IF ODD NUMBER
11887 063742  006305                          ASL     R5              ;RESTORE WORD
11888 063744  010537  062020          O.GOGO: MOV     R5,O.UPC        ;SET UP NEW PC
11889 063750  012746  000340                  MOV     #O.STM,-(SP)    ;SET HIGH PRIORITY
11890 063754  004737  065322                  JSR     PC,O.WST
11891 063760  004537  064654                  JSR     5,O.RSTT        ;RESTORE TELETYPE
11892 063764  105037  061712          O.TBIT: CLRB    O.T             ;CLEAR
11893 063770  052737  000020  062022          BIS     #O.TBT,O.UST    ;    BOTH T-BIT FLAGS
11894 063776  105737  061711                  TSTB    O.S             ;SEE IF WE NEED A T BIT
11895 064002  001005                          BNE     O.GO2           ;IF NOT GO NOW
11896 064004  042737  000020  062022          BIC     #O.TBT,O.UST    ;CLEAR THE T BIT
11897 064012  004537  064570                  JSR     5,O.RSB         ;RESTORE BREAKPOINTS
11898 064016  004037  064536          O.GO2:  JSR     0,O.RSR         ;RESTORE REGISTERS
11899 064022  013746  062022                  MOV     O.UST,-(SP)     ;    AND STATUS
11900 064026  013746  062020                  MOV     O.UPC,-(SP)     ;    AND PC
11901 064032  000240                          NOP                     ; CHANGE TO HALT FOR DEBUGGING
11902 064034  013746  062022                  MOV     O.UST,-(SP)     ; MOV IN STATUS FIRST W/O T BIT
11903 064040  042716  177420                  BIC     #O.TBT!177400,(SP);  SO INTERRUPTS CAN HAPPEN BEFORE
11904 064044  004737  065322                  JSR     PC,O.WST        ;  RTT TURNS ON THE T BIT.
11905 064050  000006                          RTT
```

```
11908                                   ; PROCESS P - PROCEED
11909                                   ;   ONLY ALLOWED AFTER A BREAKPOINT
11910
11911 064052  105737  061721          O.PROC: TSTB    O.SMFD          ;WAS ";" TYPED?
11912 064056  001715                          BEQ     O.ERR1          ;BR IF NOT TYPED
11913 064060  113700  061713                  MOVB    O.P,R0
11914 064064  105700                          TSTB    R0              ;CHECK LEGALITY OF PROCEED
11915 064066  002711                          BLT     O.ERR1          ;NOT LEGAL
11916 064070  005702                          TST     R2              ;CHECK FOR ILLEGAL COUNT
11917 064072  001307                          BNE     O.ERR1          ;JUMP IF ILLEGAL
11918 064074  005703                          TST     R3              ;WAS COUNT SPECIFIED?
11919 064076  001402                          BEQ     O.PR1           ;NO
11920 064100  010560  062050                  MOV     R5,O.CT(R0)     ;YES, PUT AWAY COUNT
11921 064104  012746  000340          O.PR1:  MOV     #O.STM,-(SP)    ;FORCE HIGH PRIORITY
11922 064110  004737  065322                  JSR     PC,O.WST
11923 064114  004537  064654                  JSR     5,O.RSTT        ;RESTORE TTY
11924 064120  123727  061713  000016  O.C1:   CMPB    O.P,#O.BKP      ;SEE IF A REAL ONE OR A FAKE
11925 064126  003316                          BGT     O.TBIT          ;BRANCH IF FAKE
11926 064130  105737  061711                  TSTB    O.S             ;SEE IF SINGLE INSTRUCTION MODE
11927 064134  001313                          BNE     O.TBIT          ;IF SO EXIT NOW
11928 064136  012746  000340                  MOV     #O.STM,-(SP)    ;SET HIGH PRIORITY
11929 064142  004737  065322                  JSR     PC,O.WST
11930 064146  105237  061712                  INCB    O.T             ;SET T-BIT FLAG
11931 064152  052737  000020  062022          BIS     #O.TBT,O.UST    ;SET T-BIT
11932 064160  000716                          BR      O.GO2
```

```
11935                                   ; BREAKPOINT HANDLER
11936 064162  012637  062020          0.BRK:  MOV     (SP)+,0.UPC     ;PRIORITY IS 7 UPON ENTRY
11937 064166  012637  062022                  MOV     (SP)+,0.UST     ;SAVE STATUS AND PC
11938 064172  112737  000021  061713          MOVB    #0.BKP+3,0.P    ;TELL ;P THAT WE CAN CONTINUE
11939 064200  004037  064500          0.BK1:  JSR     0,0.SVR         ;SAVE VARIOUS REGISTERS
11940 064204  105737  061712                  TSTB    0.T             ;CHECK FOR T-BIT SET
11941 064210  001265                          BNE     0.TBIT          ;JUMP IF SET
11942 064212  004537  064722                  JSR     5,0.REM         ;REMOVE BREAKPOINTS
11943 064216  105737  062024                  TSTB    0.PRI           ;CHECK IF PRIORITY
11944 064222  100003                          BPL     2$              ; IS AS SAME AS USER PGM
11945 064224  113705  062022                  MOVB    0.UST,R5        ;PICK UP USER UST IF SO
11946 064230  000407                          BR      3$
11947 064232  113705  062024          2$:     MOVB    0.PRI,R5        ;OTHERWISE PICK UP ACTUAL PRIORITY
11948 064236  000257                          CCC                     ;CLEAR CARRY
11949 064240  106005                          RORB    R5              ;SHIFT LOW ORDER BITS
11950 064242  106005                          RORB    R5              ;  INTO
11951 064244  106005                          RORB    R5              ;    HIGH ORDER
11952 064246  106005                          RORB    R5              ;      POSITION
11953 064250  042705  177420          3$:     BIC     #0.TBT!177400,R5;CLEAR POSSIBLE T BIT (S/I MODE)
11954 064254  010546                          MOV     R5,-(SP)        ;PUT THE STATUS AWAY WHERE IT BELONGS
11955 064256  004737  065322                  JSR     PC,0.WST
11956 064262  013705  062020                  MOV     0.UPC,R5        ;GET PC, IT POINTS TO THE TRT
11957 064266  105737  061711                  TSTB    0.S             ;SEE IF IT WAS SINGLE INSTRUCTION FUN
11958 064272  100432                          BMI     14$             ;IF SO HANDLE THERE
11959 064274  005745                          TST     -(R5)
11960 064276  010537  062020                  MOV     R5,0.UPC
11961 064302  012704  000016                  MOV     #0.BKP,R4       ;GET A COUNTER
11962 064306  020564  062026          11$:    CMP     R5,0.ADR1(R4)   ;COMPARE WITH LIST
11963 064312  001426                          BEQ     12$             ;JUMP IF FOUND
11964 064314  005304                          DEC     R4
11965 064316  005304                          DEC     R4
11966 064320  002372                          BGE     11$             ;RE-LOOP UNTIL FOUND
11967 064322  004537  064620                  JSR     5,0.SVTT        ;SAVE TELETYPE STATUS
11968 064326  004537  065366                  JSR     5,0.CRLF
11969 064332  012704  065506                  MOV     #0.BD,R4        ;ERROR, NOTHING FOUND
11970 064336  012703  065507                  MOV     #0.BD+1,R3
11971 064342  004537  065246                  JSR     5,0.TYPE        ;OUTPUT 'BE'' FOR BAD ENTRY
11972 064346  010500                          MOV     R5,R0
11973 064350  062737  000002  062020          ADD     #2,0.UPC        ;POP OVER THE ADJUSTMENT ABOVE
11974 064356  000444                          BR      13$             ; OR CONTINUE
```

```
11977 064360  112704  000020          14$:    MOVB    #O.BKP+2,R4     ;SET BREAK POINT HIGH + 1
11978 064364  010564  062026                  MOV     R5,O.ADR1(R4)   ;STORE NEXT PC VALUE FOR TYPE OUT
11979 064370  110437  061713          12$:    MOVB    R4,O.P          ;ALLOW PROCEED
11980 064374  005364  062050                  DEC     O.CT(R4)
11981 064400  003247                          BGT     O.C1            ;JUMP IF REPEAT
11982 064402  012764  000001  062050          MOV     #1,O.CT(R4)     ;RESET COUNT TO 1
11983 064410  004537  064620                  JSR     5,O.SVTT        ;SAVE TELETYPE STATUS, R4 IS SAFE
11984 064414  012700  000102                  MOV     #'B,R0
11985 064420  004537  065050                  JSR     5,O.FTYP        ;TYPE 'B'
11986 064424  113700  061713                  MOVB    O.P,R0          ;CONVERT BREAKPOINT NUMBER TO ASCII
11987 064430  062700  000140                  ADD     #140,R0
11988 064434  006200                          ASR     R0
11989 064436  004537  065050                  JSR     5,O.FTYP
11990 064442  012700  000073                  MOV     #';,R0
11991 064446  004537  065050                  JSR     5,O.FTYP        ; TYPE
11992 064452  012737  000002  061706          MOV     #2,O.BW         ; SET WORD MODE
11993 064460  113704  061713                  MOVB    O.P,R4
11994 064464  016400  062026                  MOV     O.ADR1(R4),R0   ;GET ADDRESS OF BREAK
11995 064470  004537  065412          13$:    JSR     5,O.RORA        ; CHECK FORMAT
11996 064474  000137  062640                  JMP     O.DCD           ;GO TO DECODER
```

```
11999                                   ; SAVE REGISTERS R0-R6
12000                                   ;   INTERNAL STACK
12001
12002 064500  012637  061704      O.SVR: MOV     (SP)+,O.XXX     ;PICK REGISTER FROM STACK AND SAVE
12003 064504  010637  062016             MOV     SP,O.USP        ;SAVE USER STACK ADDRESS
12004 064510  012706  062016             MOV     #O.USP,SP       ;SET TO INTERNAL STACK
12005 064514  010546                     MOV     R5,-(SP)        ;SAVE
12006 064516  010446                     MOV     R4,-(SP)        ; REGISTERS
12007 064520  010346                     MOV     R3,-(SP)        ;1
12008 064522  010246                     MOV     R2,-(SP)        ; THRU
12009 064524  010146                     MOV     R1,-(SP)        ;     5
12010 064526  013746  061704             MOV     O.XXX,-(SP)     ;PUT SAVED REGISTER ON STACK
12011 064532  005746                     TST     -(SP)
12012 064534  000200                     RTS     R0
12013
12014                                   ; RESTORE REGISTERS R0-R6
12015
12016 064536  005726               O.RSR: TST     (SP)+           ;POP THE EXTRA CELL
12017 064540  012637  061704             MOV     (SP)+,O.XXX     ;GET R0 FROM STACK
12018 064544  012601                     MOV     (SP)+,R1        ;RESTORE
12019 064546  012602                     MOV     (SP)+,R2        ; REGISTERS
12020 064550  012603                     MOV     (SP)+,R3        ;   1
12021 064552  012604                     MOV     (SP)+,R4        ;   THRU
12022 064554  012605                     MOV     (SP)+,R5        ;     5
12023 064556  013706  062016             MOV     O.USP,SP        ;RESTORE USER STACK
12024 064562  013746  061704             MOV     O.XXX,-(SP)     ;PUT R0 ON USER STACK
12025 064566  000200                     RTS     R0
```

```
12028                                   ; RESTORE BREAKPOINTS 0-7
12029
12030 064570  012704  000016            O.RSB:  MOV     #O.BKP,R4       ;RESTORE ALL BREAKPOINTS
12031 064574  017464  062026  062072     1$:     MOV     @O.ADR1(R4),O.UIN(R4);SAVE CONTENTS
12032 064602  013774  065510  062026             MOV     O.TRTC,@O.ADR1(R4);REPLACE WITH TRAP
12033 064610  005304                             DEC     R4
12034 064612  005304                             DEC     R4
12035 064614  002367                             BGE     1$              ;RE-LOOP UNTIL DONE
12036 064616  000205                             RTS     R5              ;  THEN QUIT
12037
12038                                   ; SAVE TELETYPE STATUS
12039
12040 064620  113737  177560  061714     O.SVTT: MOVB    O.RCSR,O.CSR1   ;SAVE R C/SR
12041 064626  113737  177564  061715             MOVB    O.TCSR,O.CSR2   ;SAVE T C/SR
12042 064634  105037  177560                     CLRB    O.RCSR          ;CLEAR ENABLE AND MAINTENANCE
12043 064640  105037  177564                     CLRB    O.TCSR          ;  BITS IN BOTH C/SR
12044 064644  105737  177564             1$:     TSTB    O.TCSR          ;LOOP UNTIL READY BIT COMES ON
12045 064650  100375                             BPL     1$              ;BR IF BIT NOT ON
12046 064652  000205                             RTS     R5
12047                                   ; RESTORE TELETYPE STATUS
12048
12049 064654  004537  065366             O.RSTT: JSR     5,O.CRLF
12050 064660  105737  177564                     TSTB    O.TCSR          ;WAIT READY
12051 064664  100375                             BPL     .-4             ;  ON PRINTER
12052 064666  032737  004000  177560             BIT     #4000,O.RCSR    ;CHECK BUSY FLAG
12053 064674  001403                             BEQ     1$              ;SKIP READY LOOP IF NOT BUSY
12054 064676  105737  177560                     TSTB    O.RCSR          ;WAIT READY
12055 064702  100375                             BPL     .-4             ;    ON READER
12056 064704  113737  061714  177560     1$:     MOVB    O.CSR1,O.RCSR   ;RESTORE
12057 064712  113737  061715  177564             MOVB    O.CSR2,O.TCSR   ;  THE STATUS REGISTERS
12058 064720  000205                             RTS     R5
12059
12060                                   ; REMOVE BREAKPOINTS 0-7
12061                                   ;   IN THE OPPOSITE ORDER OF SETTING
12062
12063 064722  105737  061711             O.REM:  TSTB    O.S             ;SEE IF SINGLE INSTRUCTION IS GOING
12064 064726  001011                             BNE     2$              ;EXIT IF SO
12065 064730  005004                             CLR     R4              ;REMOVE ALL BREAKPOINTS
12066 064732  016474  062072  062026     1$:     MOV     O.UIN(R4),@O.ADR1(R4)   ;CLEAR BREAKPOINT
12067 064740  005204                             INC     R4
12068 064742  005204                             INC     R4
12069 064744  020427  000016                     CMP     R4,#O.BKP
12070 064750  003770                             BLE     1$              ;RE-LOOP UNTIL DONE
12071 064752  000205                     2$:     RTS     R5              ;THEN QUIT
```

```
12074                                   ; TYPE OUT CONTENTS OF WORD OR BYTE WITH ONE TRAILING SPACE
12075                                   ; WORD IS IN R0
12076
12077 064754  012703  000006            O.CADV: MOV     #6,R3           ;# OF DIGITS
12078 064760  012704  177776                    MOV     #-2,R4          ;# OF BITS FIRST-3
12079 064764  022737  000001  061706            CMP     #1,O.BW         ;SEE IF WORD MODE
12080 064772  001004                            BNE     3$              ;BRANCH IF SO
12081 064774  162703  000003                    SUB     #3,R3           ;ONLY DO 3 DIGITS
12082 065000  005204                            INC     R4              ;DO 2 BITS FIRST
12083 065002  000300                            SWAB    R0              ;AND TURN R0 AROUND
12084 065004  010046                    3$:     MOV     R0,-(SP)        ;SAVE R0
12085 065006  062704  000003            2$:     ADD     #3,R4           ;COMPUTE THE NUMBER OF BITS TO DO
12086 065012  005000                            CLR     R0
12087 065014  006116                    1$:     ROL     (SP)            ;GET A BIT
12088 065016  006100                            ROL     R0              ;STORE IT AWAY
12089 065020  005304                            DEC     R4              ;DECREMENT COUNTER
12090 065022  003374                            BGT     1$              ;LOOP IF MORE BITS NEEDED
12091 065024  062700  000060                    ADD     #'0,R0          ;CONVERT TO ASCII
12092 065030  004537  065050                    JSR     R5,O.FTYP       ;TYPE IT
12093 065034  005303                            DEC     R3              ;SEE IF MORE DIGITS TO DO
12094 065036  003363                            BGT     2$              ;LOOP IF SO
12095 065040  112700  000040                    MOVB    #' ,R0          ;SET UP FOR TRAILING SPACE
12096 065044  005726                            TST     (SP)+           ;GET RID OF JUNK
12097 065046  000400                            BR      O.FTYP
12098
12099
12100                                   ; TYPE ONLY ONE CHARACTER (CONTAINED IN R0)
12101
12102 065050  105737  177564            O.FTYP: TSTB    O.TCSR
12103 065054  100375                            BPL     O.FTYP
12104 065056  042700  177400                    BIC     #177400,R0      ;CLEAR HIGH BYTE,SHOULD NOT
12105                                                                   ;CONTAIN INPORTANT INFO.
12106 065062  001416                            BEQ     3$
12107 065064  022700  000176                    CMP     #176,R0         ;PRINT ? FOR 177;16-37
12108 065070  103411                            BLO     2$              ; 1-10 AND 200-377.
12109 065072  022700  000037                    CMP     #37,R0
12110 065076  103410                            BLO     3$
12111 065100  022700  000015                    CMP     #15,R0
12112 065104  103403                            BLO     2$
12113 065106  022700  000010                    CMP     #10,R0
12114 065112  103402                            BLO     3$
12115 065114  012700  000077            2$:     MOV     #'?,R0
12116 065120  110037  177566            3$:     MOVB    R0,O.TDB
12117 065124  000205                    O.TYP1: RTS     R5
```

```
12120                                   ; GENERAL CHARACTER INPUT ROUTINE -- ODT11X
12121                                   ; CHARACTER INPUT GOES TO R0
12122
12123 065126  105737  177560      O.GET:  TSTB    O.RCSR          ;WAIT FOR
12124 065132  100375                      BPL     O.GET           ;  INPUT FROM KBD
12125 065134  113700  177562              MOVB    O.RDB,R0        ;GET CHARACTER - STRIP OFF PARITY
12126 065140  042700  177600              BIC     #177600,R0      ;STRIP OFF PARITY FROM CHARACTER
12127 065144  122700  000003              CMPB    #3,R0           ;IS IT ^C?
12128 065150  001435                      BEQ     3$              ;IF SO, DO NOT ECHO
12129 065152  122700  000006              CMPB    #6,R0           ;IS IT ^F?
12130 065156  001432                      BEQ     3$              ;IF SO, DO NOT ECHO
12131 065160  122700  000005              CMPB    #5,R0           ;IS IT ^E?
12132 065164  001427                      BEQ     3$              ;IF SO, DO NOT ECHO
12133 065166  105737  061722              TSTB    O.SCRN          ;SHOULD WE ECHO <LF>?
12134 065172  001003                      BNE     2$              ;BR IF YES
12135 065174  120027  000012              CMPB    R0,#012         ;SEE IF A <LF>
12136 065200  001421                      BEQ     3$              ;IF SO SAVE THE PAPER
12137 065202  120027  000173      2$:     CMPB    R0,#173
12138 065206  002005                      BGE     1$
12139 065210  122700  000140              CMPB    #140,R0         ;TEST FOR LOWER CASE
12140 065214  002002                      BGE      1$
12141 065216  162700  000040              SUB     #40,R0          ;CHAR IS LC-CONVERT TO UPPER CASE
12142 065222  004537  065050      1$:     JSR     5,O.FTYP        ;ECHO CHARACTER
12143 065226  001737                      BEQ     O.GET           ;IGNORE NULLS
12144 065230  105737  061722              TSTB    O.SCRN          ;SHOULD WE PASS ON SPACES?
12145 065234  001003                      BNE     3$              ;BR IF YES
12146 065236  122700  000040              CMPB    #40,R0          ;CHECK FOR SPACES
12147 065242  001731                      BEQ     O.GET           ;IGNORE SPACES
12148 065244  000205              3$:     RTS     R5
```

```
                                        ; GENERAL CHARACTER OUTPUT ROUTINE - ODT11X
                                        ;  ADDRESS OF FIRST BYTE IN R4,
                                        ;  ADDRESS OF LAST BYTE IN R3, (R3)>(R4)
                                        ; EXPECTS LOCS 56,57 TO BE INITIALIZED BY MONITOR FOR FILL
                                        ; CHARACTERISTICS OF TERMINAL
                                        ;  56=CHAR TO BE FILLED AFTER
                                        ;  57=# OF NULLS TO FILL WITH

065246  020304                  O.TYPE: CMP     R3,R4           ;CHECK FOR COMPLETION
065250  103725                          BLO     O.TYP1          ; EXIT WHEN DONE
065252  112400                          MOVB    (R4)+,R0        ;GET A CHARACTER
065254  004537  065050                  JSR     5,O.FTYP        ;TYPE ONE CHARACTER
065260  120037  002612                  CMPB    R0,@#$FILLC     ;COMPARE CHAR AGAINST FILL REQUIREMENT
065264  001370                          BNE     O.TYPE          ;NO FILL NEEDED
065266  113737  002327  061716          MOVB    @#$FILLS,O.FIL  ;FILL COUNT INTO TEMP
065274  005000                          CLR     R0              ;FILL WITH NULLS
065276  004537  065050          2$:     JSR     R5,O.FTYP       ;TYPE NULLS
065302  105337  061716                  DECB    O.FIL           ;DECREASE COUNT
065306  003373                          BGT     2$              ;BRANCH IF NOT DONE
065310  000756                          BR      O.TYPE          ;LOOP UNTIL DONE
```

```
12173                                   ;SUBROUTINE TO READ THE PS INDEPENDENT OF MACHINE
12174
12175 065312  013737  177776  062022  O.RRST: MOV     ST,O.UST        ;STORE THE STATUS IN USER
12176                                                                 ;STATUS AREA.FOR AN LSI
12177                                                                 ;THIS IS CHANGED TO
12178                                                                 ;MFPS O.UST
12179 065320  000207                          RTS     PC
12180
12181                                   ;SUBROUTINE TO WRITE PS INDEPENDENT OF MACHINE
12182                                   ;CALL ROUTINE WITH PS VALUE
12183                                   ;ON THE STACK
12184
12185 065322  016637  000002  177776  O.WST:  MOV     2(SP),ST        ;STORE NEW PS VALUE
12186                                                                 ;THIS INSTRUCTION IS CHANGED FOR LSI
12187                                                                 ;TO MTPS 2(SP)
12188 065330  012616                          MOV     (SP)+,(SP)      ;PUT RETURN PC OVER SUB. ARGUMENT
12189 065332  000207                          RTS     PC              ;TO RETURN WITHOUT IT ON THE STACK
12190                                   ; CLOSE WORD OR BYTE AND EXIT,
12191                                   ; UPON ENTERING, R2 HAS NUMERIC FLAG, R4 HAS CONTENTS
12192
12193                                           .ENABL  LSB
12194 065334  005702                  O.CLSE: TST     R2              ;IF NO NUMBER WAS TYPED THERE IS
12195 065336  001412                          BEQ     1$              ;NO CHANGE TO THE OPEN CELL
12196 065340  022737  000001  061706          CMP     #1,O.BW
12197 065346  001404                          BEQ     2$              ;JUMP IF BYTE MODE
12198 065350  101005                          BHI     1$              ;JUMP IF ALREADY CLOSED
12199 065352  010477  174322                  MOV     R4,@O.CAD       ;STORE WORD
12200 065356  000402                          BR      1$
12201 065360  110477  174314          2$:     MOVB    R4,@O.CAD       ;STORE BYTE
12202 065364  000207                  1$:     RTS     PC
12203 065366  012703  065435          O.CRLF: MOV     #O.CR+1,R3      ;LWA <CR,LF>
12204 065372  000402                          BR      3$
12205 065374  012703  065441          O.CRLS: MOV     #O.CR+5,R3      ;LWA <CR,LF>ODT>
12206 065400  012704  065434          3$:     MOV     #O.CR,R4        ;FWA
12207 065404  004537  065246                  JSR     5,O.TYPE        ;TYPE SOMETHING
12208 065410  000205                          RTS     R5
12209                                           .DSABL  LSB
```

```
12212                                   ;SUBROUTINE O.RORA
12213                                   ;FUNCTION:                              THE
12214                                   ;ADDRESS WILL BE PRINTED IN ABSOLUTE FORM
12215                                   ;INPUT: THE ADDRESS TO BE PRINTED IS IN R0.
12216                                   ;DATA SAVED:  R0 CONTAINING THE ADDRESS TO BE
12217                                   ;PRINTED, AND LOCATION O.CAD CONTAINING
12218                                   ;THE CURRENT ADDRESS WERE SAVED AND RESTORED.
12219                                   ;CALLED:  JSR 5,O.RORA
12220
12221 065412  004537  064754           O.RORA: JSR     5,O.CADV        ;PRINT ABSOLUTE ADDRESS
12222 065416  000205                           RTS     R5
12223 065420     012                   O.ID:   .BYTE   012
12224 065421     015                           .BYTE   015
12225 065422     040                           .BYTE   40
12226 065423     103     132     115           .ASCII ''CZMSD ODT''
      065426     123     104     040
      065431     117     104     124
12227          065433                 O.IDND=.-1
12228 065434     015                   O.CR:   .BYTE   015     ;  <CR>
12229 065435     012                           .BYTE   012     ;  <LF>
12230 065436     117                           .BYTE   'O      ;  O
12231 065437     104                           .BYTE   'D      ;  D
12232 065440     124                           .BYTE   'T      ;  T
12233 065441     076                           .BYTE   '>      ;  >
12234
12235 065442     073                   O.LGCH: .BYTE   ';      ;
12236 065443     057                           .BYTE   '/      ;  /
12237 065444     134                           .BYTE   '\      ;  \     (BACK SLASH)
12238 065445     015                           .BYTE   015     ;  CARRIAGE RETURN
12239 065446     044                           .BYTE   '$      ;  $
12240 065447     107                           .BYTE   'G      ;  G
12241 065450     012                           .BYTE   012     ;  <LF>
12242 065451     137                           .BYTE   '_      ;       (BACK ARROW)
12243 065452     074                           .BYTE   '<      ;  <
12244 065453     136                           .BYTE   '^      ;  ^     (UP ARROW)
12245 065454     117                           .BYTE   'O      ;  O
12246 065455     102                           .BYTE   'B      ;  B
12247 065456     120                           .BYTE   'P      ;  P
12248 065457     100                           .BYTE   '@      ;  @
12249 065460     076                           .BYTE   '>      ;  >
12250 065461     123                           .BYTE   'S      ;  S
12251 065462     055                           .BYTE   '-      ;  -
12252 065463     003                           .BYTE   003     ;CTRL C
12253 065464     006                           .BYTE   006     ;CTRL F
12254 065465     005                           .BYTE   005     ;CTRL E
12255          000024                 O.CLGT  =       .-O.LGCH                ;TABLE LENGTH
12256
12257 065466     123                   O.TL:   .BYTE   'S      ;DO        1
12258 065467     120                           .BYTE   'P      ;NOT       2
12259 065470     115                           .BYTE   'M      ;CHANGE    3
12260 065471     000                           .BYTE   0       ;THE       4
12261 065472     000                           .BYTE   0       ;ORDER     5
12262 065473     103                           .BYTE   'C      ;          6
12263 065474     106                           .BYTE   'F      ;          7
12264 065475     122                           .BYTE   'R      ;         10
12265 065476     000                           .BYTE   0       ;         11
12266 065477     000                           .BYTE   0       ;         12
```

```
 12267 065500  000                        .BYTE   0       ;       13
 12268 065501  000                        .BYTE   0       ;       14
 12269 065502  000                        .BYTE   0       ;       15
 12270 065503  000                        .BYTE   0       ;       16
 12271 065504  000                        .BYTE   0       ;       17
 12272 065505  102                        .BYTE   'B      ;       20
 12273         000020          O.LG       =       .-O.TL
 12274                                    .EVEN
 12275 065506  042502          O.BD:     .WORD   'BE
 12276 065510  000003          O.TRTC:   TRT                     ;TRACE TRAP PROTOTYPE
```

```
                                        .SBTTL  TABLE   ERROR POINTER

                                ;*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


065512                          $ERRTB: ;ERROR  1
065512  070003                          EM24
065514  072130                          DH13
065516  066362                          DT13
065520  066720                          DF11
                                        ;ERROR  2
065522  066757                          EM2
065524  071437                          DH1
065526  066212                          DT1
065530  066576                          DF2
                                        ;ERROR  3
065532  067015                          EM3
065534  071517                          DH3
065536  066224                          DT3
065540  066713                          DF9
                                        ;ERROR  4
065542  067047                          EM4
065544  071517                          DH3
065546  066234                          DT4
065550  066713                          DF9
                                        ;ERROR  5
065552  067115                          EM5
065554  071553                          DH5
065556  066244                          DT5
065560  066576                          DF2
                                        ;ERROR  6
065562  067172                          EM6
065564  071553                          DH5
065566  066244                          DT5
065570  066576                          DF2
                                        ;ERROR  7
065572  067217                          EM7
065574  071553                          DH5
065576  066244                          DT5
065600  066576                          DF2
                                        ;ERROR  10
065602  071335                          EM53
065604  072573                          DH25
065606  066540                          DT25
065610  066576                          DF2
```

```
12335                                   ;ERROR  11
12336 065612  067257                    EM11
12337 065614  071677                    DH7
12338 065616  066276                    DT7
12339 065620  066622                    DF3
12340                                   ;ERROR  12
12341 065622  067257                    EM11
12342 065624  071677                    DH7
12343 065626  066276                    DT7
12344 065630  066635                    DF4
12345                                   ;ERROR  13
12346 065632  067301                    EM12
12347 065634  072007                    DH10
12348 065636  066326                    DT10
12349 065640  066576                    DF2
12350                                   ;ERROR  14
12351 065642  067257                    EM11
12352 065644  071677                    DH7
12353 065646  066276                    DT7
12354 065650  066650                    DF5
12355                                   ;ERROR  15
12356 065652  067257                    EM11
12357 065654  071677                    DH7
12358 065656  066276                    DT7
12359 065660  066663                    DF6
12360                                   ;ERROR  16
12361 065662  067325                    EM13
12362 065664  072130                    DH13
12363 065666  066362                    DT13
12364 065670  066720                    DF11
12365                                   ;ERROR  17
12366 065672  067357                    EM14
12367 065674  072130                    DH13
12368 065676  066362                    DT13
12369 065700  066720                    DF11
12370                                   ;ERROR  20
12371 065702  067423                    EM15
12372 065704  072130                    DH13
12373 065706  066362                    DT13
12374 065710  066720                    DF11
12375                                   ;ERROR  21
12376 065712  071364                    EM55
12377 065714  072617                    DH26
12378 065716  066550                    DT26
12379 065720  066576                    DF2
12380                                   ;ERROR  22
12381 065722  067471                    EM17
12382 065724  071677                    DH7
12383 065726  066276                    DT7
12384 065730  066650                    DF5
12385                                   ;ERROR  23
12386 065732  071172                    EM50
12387 065734  072445                    DH23
12388 065736  066476                    DT23
12389 065740  066731                    DF13
```

```
12392                                   ;ERROR  24
12393 065742  067531                    EM19
12394 065744  072130                    DH13
12395 065746  066362                    DT13
12396 065750  066720                    DF11
12397                                   ;ERROR  25
12398 065752  067606                    EM20
12399 065754  072130                    DH13
12400 065756  066362                    DT13
12401 065760  066720                    DF11
12402                                   ;ERROR  26
12403 065762  000000                    0               ;NO MESSAGE
12404 065764  072123                    DH12
12405 065766  066356                    DT12
12406 065770  066576                    DF2
12407                                   ;ERROR  27
12408 065772  067670                    EM21
12409 065774  072105                    DH11
12410 065776  066350                    DT11
12411 066000  066576                    DF2
12412                                   ;ERROR  30
12413 066002  067727                    EM22
12414 066004  072130                    DH13
12415 066006  066362                    DT13
12416 066010  066720                    DF11
12417                                   ;ERROR  31
12418 066012  000000                    0               ;NO MESSAGE
12419 066014  072225                    DH14
12420 066016  066404                    DT14
12421 066020  066576                    DF2
12422                                   ;ERROR  32
12423 066022  067754                    EM23
12424 066024  071553                    DH5
12425 066026  066244                    DT5
12426 066030  066576                    DF2
12434                                   ;ERROR  33
12435 066032  070062                    EM25
12436 066034  072304                    DH15
12437 066036  066422                    DT16
12438 066040  066676                    DF7
12439                                   ;ERROR  34
12440 066042  070107                    EM26
12441 066044  072423                    DH16
12442 066046  066452                    DT17
12443 066050  066622                    DF3
```

```
12453                                   ;ERROR  35
12454 066052  071310                    EM52
12455 066054  072573                    DH25
12456 066056  066540                    DT25
12457 066060  066576                    DF2
12458                                   ;ERROR  36
12459 066062  070160                    EM27
12460 066064  072423                    DH16
12461 066066  066452                    DT17
12462 066070  066711                    L
12470                                   ;ERROR  37
12471 066072  070764                    EM35
12472 066074  071677                    DH7
12473 066076  066276                    DT7
12474 066100  066622                    DF3
12475                                   ;ERROR  40
12476 066102  070250                    EM29
12477 066104  071677                    DH7
12478 066106  066276                    DT7
12479 066110  066622                    DF3
12480                                   ;ERROR  41
12481 066112  070332                    EM30
12482 066114  071677                    DH7
12483 066116  066276                    DT7
12484 066120  066650                    DF5
12485                                   ;ERROR  42
12486 066122  070451                    EM31
12487 066124  071677                    DH7
12488 066126  066276                    DT7
12489 066130  066622                    DF3
12490                                   ;ERROR  43
12491 066132  070551                    EM32
12492 066134  071677                    DH7
12493 066136  066276                    DT7
12494 066140  066622                    DF3
12495                                   ;ERROR  44
12496 066142  070656                    EM33
12497 066144  071677                    DH7
12498 066146  066276                    DT7
12499 066150  066622                    DF3
12500                                   ;ERROR  45
12501 066152  071226                    EM51
12502 066154  072524                    DH24
12503 066156  066520                    DT24
12504 066160  066741                    DF14
12505                                   ;ERROR  46
12506 066162  071051                    EM36
12507 066164  071632                    DH6
12508 066166  066262                    DT6
12509 066170  066576                    DF2
```

```
                                                    G 3
 12524                                              ;ERROR  47
 12525 066172  071120                               EM40
 12526 066174  071474                               DH2
 12527 066176  066460                               DT20
 12528 066200  066576                               DF2
 12566                                              ;ERROR  50
 12567 066202  071405                               EM56
 12568 066204  072635                               DH27
 12569 066206  066556                               DT27
 12570 066210  066750                               DF15
```

```
12580                                                   .SBTTL  ERROR DATA TAGS (DT)
12581 066212  002016  002032  002042  DT1:    .WORD   ERRPC,ADDRESS,GOOD,BAD,0
      066220  002050  000000
12585 066224  002016  002034  002070  DT3:    .WORD   ERRPC,PADDRESS,PARCNT,0
      066232  000000
12586 066234  002016  002032  002066  DT4:    .WORD   ERRPC,ADDRESS,NEMCNT,0
      066242  000000
12587 066244  002016  177572  177574  DT5:    .WORD   ERRPC,MMR0,MMR1,MMR2,MMR3,CPUERR,0
      066252  177576  172516  177766
      066260  000000
12588 066262  002016  002372  002350  DT6:    .WORD   ERRPC,APTPAR,LSIZE,APTECC,MSIZE,0
      066270  002374  002352  000000
12589 066276  002016  002170  002032  DT7:    .WORD   ERRPC,DUMMY,ADDRESS,DUMMY,GOOD,BAD,BADXOR
      066304  002170  002042  002050
      066312  002056
12590 066314  002170  002170  002170          .WORD   DUMMY,DUMMY,DUMMY,DUMMY,0
      066322  002170  000000
12591 066326  002172  002174  002176  DT10:   .WORD   DETR0,DETR1,DETR2,DETR3,DETR4,DETR5,DETSP,DETPSW,0
      066334  002200  002202  002204
      066342  002206  002210  000000
12592 066350  002016  002144  000000  DT11:   .WORD   ERRPC,CSR,0
12593 066356  002144  000000          DT12:   .WORD   CSR,0
12594 066362  002016  002170  002032  DT13:   .WORD   ERRPC,DUMMY,ADDRESS,DUMMY,TSTDAT,TSTDAT+2,CHECK,CSR,0
      066370  002170  002240  002242
      066376  002274  002144  000000
12595 066404  177746  177572  177574  DT14:   .WORD   CONTRL,MMR0,MMR1,MMR2,MMR3,CPUERR,0
      066412  177576  172516  177766
      066420  000000
12596 066422  002016  002170  002170  DT16:   .WORD   ERRPC,DUMMY,DUMMY,GOOD,GOOD2,GOOD3
      066430  002042  002044  002046
12597 066436  002050  002052  002054          .WORD   BAD,BAD2,BAD3,DUMMY,DUMMY,0
      066444  002170  002170  000000
12598 066452  002016  002170  000000  DT17:   .WORD   ERRPC,DUMMY,0
12603 066460  002016  002042  002050  DT20:   .WORD   ERRPC,GOOD,BAD,0
      066466  000000
12607 066470  002016  002170  000000  DT22:   .WORD   ERRPC,DUMMY,0
12608 066476  002016  002170  002042  DT23:   .WORD   ERRPC,DUMMY,GOOD,BAD,DUMMY,DUMMY,DUMMY,DUMMY,0
      066504  002050  002170  002170
      066512  002170  002170  000000
12609 066520  002016  002170  002144  DT24:   .WORD   ERRPC,DUMMY,CSR,DUMMY,DUMMY,DUMMY,DUMMY,0
      066526  002170  002170  002170
      066534  002170  000000
12610 066540  002016  002042  002144  DT25:   .WORD   ERRPC,GOOD,CSR,0
      066546  000000
12611 066550  002016  002050  000000  DT26:   .WORD   ERRPC,BAD,0
12612 066556  002016  002170  002032  DT27:   .WORD   ERRPC,DUMMY,ADDRESS,DUMMY,DUMMY,DUMMY,DUMMY,0
      066564  002170  002170  002170
      066572  002170  000000
```

```
12619                                                   .SBTTL  ERROR DATA FORMATS (DF)
12620 066576  000   000   000   DF2:    .BYTE   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      066601  000   000   000
      066604  000   000   000
      066607  000   000   000
      066612  000   000   000
      066615  000   000   000
      066620  000   000
12621 066622  000   005   000   DF3:    .BYTE   0,5,0,8.,0,0,0,3,6,2,4
      066625  010   000   000
      066630  000   003   006
      066633  002   004
12622 066635  000   005   000   DF4:    .BYTE   0,5,0,8.,0,8.,8.,3,6,2,4
      066640  010   000   010
      066643  010   003   006
      066646  002   004
12623 066650  000   005   000   DF5:    .BYTE   0,5,0,8.,9.,9.,9.,3,6,2,4
      066653  010   011   011
      066656  011   003   006
      066661  002   004
12624 066663  000   005   000   DF6:    .BYTE   0,5,0,8.,9.,8.,8.,3,6,2,4
      066666  010   011   010
      066671  010   003   006
      066674  002   004
12625 066676  000   005   010   DF7:    .BYTE   0,5,8.,0,0,9.,0,0,9.,2,4
      066701  000   000   011
      066704  000   000   011
      066707  002   004
12626 066711  000   005         DF8:    .BYTE   0,5
12627 066713  000   001   001   DF9:    .BYTE   0,1,1,1,1
      066716  001   001
12628 066720  000   005   000   DF11:   .BYTE   0,5,0,8.,0,0,0,0,0
      066723  010   000   000
      066726  000   000   000
12629 066731  000   005   000   DF13:   .BYTE   0,5,0,0,3,6,2,4
      066734  000   003   006
      066737  002   004
12630 066741  000   005   000   DF14:   .BYTE   0,5,0,3,6,2,4
      066744  003   006   002
      066747  004
12631 066750  000   005   000   DF15:   .BYTE   0,5,0,8.,3,6,4
      066753  010   003   006
      066756  004
```

```
12637                                           .SBTTL  ERROR MESSAGES (EM)
12646 066757  103  101  116  EM2:    .ASCIZ  /CAN'T SET 22 BIT MODE IN MMR3/
12647 067015  120  101  122  EM3:    .ASCIZ  /PARITY ERROR(S) IN BANK 0/
12648 067047  116  117  116  EM4:    .ASCIZ  /NON-EXISTANT MEMORY (HOLES) IN BANK 0/
12649 067115  111  114  114  EM5:    .ASCIZ  /ILLEGAL OR RESERVED INSTRUCTION (TRAP TO 10)/
12650 067172  125  116  105  EM6:    .ASCIZ  /UNEXPECTED TRAP TO 4/
12651 067217  115  105  115  EM7:    .ASCIZ  /MEMORY MANAGEMENT (TRAP TO 250)/
12655
12656 067257  115  105  115  EM11:   .ASCIZ  /MEMORY DATA ERROR/
12657 067301  104  105  124  EM12:   .ASCIZ  /DETAILED ERROR DUMP/
12658 067325  115  111  123  EM13:   .ASCIZ  /MISSING EXPECTED SBE FLAG/
12659 067357  127  122  111  EM14:   .ASCIZ  /WRITE BYTE FAILED TO CLEAR SBE FLAG/
12660 067423  106  101  111  EM15:   .ASCIZ  /FAILED TO GET INTERRUPT WITH DBE FLAG/
12661 067471  115  105  115  EM17:   .ASCIZ  /MEMORY DATA ERROR IN CHECK BITS/
12662 067531  123  102  105  EM19:   .ASCIZ  /SBE OR DBE CAUSED PARITY TRAP WHEN INHIBITED/
12663 067606  123  102  105  EM20:   .ASCIZ  /SBE OR DBE DID NOT CAUSE PARITY TRAP WHEN ENABLED/
12664 067670  123  102  105  EM21:   .ASCIZ  /SBE OR DBE ON MASTER TEST WORD/
12665 067727  115  111  123  EM22:   .ASCIZ  /MISSING EXPECTED DBE/
12666 067754  125  116  105  EM23:   .ASCIZ  /UNEXPECTED PARITY TRAP/
12667 070003  122  105  103  EM24:   .ASCIZ  /RECEIVED DBE FLAG WHEN EXPECTING ONLY SBE FLAG/
12668 070062  103  110  105  EM25:   .ASCIZ  /CHECK BIT DATA ERROR/
12669 070107  101  104  104  EM26:   .ASCIZ  /ADDRESS PARITY ERROR DID NOT CAUSE ABORT/
12670 070160  105  103  103  EM27:   .ASCIZ  /ECC INHIBIT MODE POINTER FAILURE - DID NOT PROTECT BANK/
12674 070250  103  117  122  EM29:   .ASCIZ  /CORRECTION FAILURE WITH ECC ENABLED ON FORCED SBE/
12675 070332  127  122  111  EM30:   .ASCII  /WRITE BYTE (MOVB) WITH ECC ENABLED FAILED TO CLEAR DATA AT/<CRLF>
12676 070425  106  117  122          .ASCIZ  /FORCED SBE LOCATION/
12677 070451  101  123  122  EM31:   .ASCIZ  /ASRB (R3)+ WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
12678 070551  115  117  126  EM32:   .ASCIZ  /MOVB #360,(R2)+ WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
12679 070656  115  117  126  EM33:   .ASCIZ  /MOV #177400,(R1) WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
12680 070764  125  116  105  EM35:   .ASCIZ  /UNEXPECTED CORRECTION WITH ECC DISABLE ON FORCED SBE/
12681 071051  101  120  124  EM36:   .ASCIZ  /APT SIZE DISAGREES WITH PROGRAM SIZING/
12687 071120  102  122  101  EM40:   .ASCIZ  /BRANCH GOBBLE FAILED CONDITION CODES TEST/
12696 071172  102  101  104  EM50:   .ASCIZ  /BAD ERROR ADDRESS GENERATED/
12697 071226  123  102  105  EM51:   .ASCIZ  /SBE & DBE FLAGS NOT SET ON FORCED UNCORRECTED SBE/
12698 071310  102  111  124  EM52:   .ASCIZ  /BIT SET ERROR IN CSR/
12699 071335  102  111  124  EM53:   .ASCIZ  /BIT CLEAR ERROR IN CSR/
12700 071364  111  114  114  EM55    .ASCIZ  /ILLEGAL CSR TYPE/
12701 071405  102  101  104  EM56:   .ASCIZ  /BAD PARITY TRAP GENERATED/
```

```
12707                                           .SBTTL  ERROR DATA HEADERS (DH)
12708 071437  040  040  120  DH1:    .ASCIZ  /  PC   DEV ADD   GOOD    BAD/
12709 071474  040  040  120  DH2:    .ASCIZ  /  PC   GD-CC BD-CC/
12710 071517  040  040  120  DH3:    .ASCIZ  /  PC   1ST ADD  # OF ERRORS/
12711 071553  040  040  120  DH5:    .ASCIZ  /  PC     MMR0    MMR1    MMR2    MMR3   CPUERR/
12712 071632  040  040  120  DH6:    .ASCIZ  /  PC   APTPAR   LSIZE  APTECC  MSIZE/
12713 071677  040  040  120  DH7:    .ASCII  /  PC    BANK  VADD      PADD      GOOD/
12714 071743  040  040  040          .ASCIZ  /      BAD      XOR  CSR  MTYP INT PAT/
12715 072007  040  040  122  DH10:   .ASCIZ  /  R0      R1      R2      R3      R4      R5      SP      PSW/
12716 072105  040  040  120  DH11:   .ASCIZ  /  PC      CSR/
12717 072123  040  103  123  DH12:   .ASCIZ  / CSR/
12718 072130  040  040  120  DH13:   .ASCII  /  PC    BANK  VADD      PADD    WROTE1  WROTE2/
12719 072205  040  103  110          .ASCIZ  / CHKBITS    CSR/
12720 072225  103  117  116  DH14:   .ASCIZ  /CONTRL   MMR0    MMR1    MMR2    MMR3   CPUERR/
12721 072304  040  040  120  DH15:   .ASCII  /  PC    BANK    PADD   GD-WD1  GD-WD2  GD-CHK/
12722 072361  040  102  101          .ASCIZ  / BAD-WD1 BAD-WD2  BAD-CHK INT PAT/
12723 072423  040  040  120  DH16:   .ASCIZ  /  PC    BANK/
12728 072440  040  040  120  DH19:   .ASCIZ  /  PC/
12734 072445  040  040  120  DH23:   .ASCIZ  /  PC    BANK GD-ERR BAD-ERR CSR  MTYP INT  PAT/
12735 072524  040  040  120  DH24:   .ASCIZ  /  PC    BANK  (CSR) CSR  MTYP INT  PAT/
12736 072573  040  040  120  DH25:   .ASCIZ  /  PC  GD-DAT  (CSR)/
12737 072617  040  040  120  DH26:   .ASCIZ  /  PC  BADCODE/
12738 072635  040  040  120  DH27:   .ASCIZ  /  PC    BANK  VADD     PADD   CSR  MTYP PAT/
```

```
12741                                           .SBTTL  MESSAGES
12742 072711  200  040  103  MSG000: .ASCIZ  <CRLF>'' CZMSDA ''
12743 072723  200  040  040  MSG001: .ASCIZ  <CRLF>/                              MEMORY CONFIGURATION MAP/
12744 073005  200  040  040  MSG002: .ASCIZ  <CRLF>/                                  16K WORD BANKS/
12745 073062  200  040  040  MSG003: .ASCII <CRLF>/                  1       2       3/
12746 073124  040  040  040          .ASCIZ  /       4       5       6       7  /
12747 073167  200  040  040  MSG004: .ASCII <CRLF>/        01234567012345670123456 7/
12748 073230  060  061  062          .ASCIZ  /01234567012345670123456701234567 0123/
12749 073275  200  105  122  MSG005: .ASCIZ  <CRLF>/ERRORS  /
12750 073307  200  125  116  MSG006: .ASCIZ  <CRLF>/UNDEFINED TRAP INSTRUCTION/<32>
12751 073344  200  111  116  MSG007: .ASCIZ  <CRLF>/INTRLV  /                    ;INTERLEAVED CSR #
12752 073356  200  103  120  MSG008: .ASCIZ  <CRLF>/CPU MAP /                    ;CPU ACCESSED BANK
12753 073370  200  115  105  MSG009: .ASCIZ  <CRLF>/MEMTYPE /                    ;MEMORY TYPE
12754 073402  200  120  122  MSG010: .ASCIZ  <CRLF>/PROTECT /                    ;MEMORY PROTECTED
12755 073414  040  040  040  MSG011: .ASCIZ  /    0       1       2       3       4       5       6/
12756 073502  064  065  066  MSG012: .ASCIZ  /4567012345670123456701234567012345670123456701234567012345670123456701234567/
12757 073577  130  000       MSG013: .ASCIZ  /X/
12758 073601  040  000       MSG014: .ASCIZ  / /             ;SPACE
12759 073603  000  000       MSG015: .BYTE   0,0             ;FOR SINGLE ASCII CHARACTERS & TERMINATOR
12760 073605  200  103  123  MSG016: .ASCIZ  <CRLF>/CSR     /
12761 073617  040  040  040  MSG017: .ASCIZ  /       /                      ;8 SPACES
12762 073630  040  040  000  MSG018: .ASCIZ  /  /                           ;2 SPACES
12763 073633  040  040  040  MSG019: .ASCIZ  /   /                          ;3 SPACES
12764 073637  200  106  111  MSG020: .ASCIZ  <CRLF>/FIELD SERVICE COMMAND MODE/
12765 073673  200  103  117  MSG021: .ASCII  <CRLF>/COMMANDS AVAILABLE ARE:/
12766 073723  200  060  040          .ASCII  <CRLF>/0 = EXIT FIELD SERVICE COMMANDS/
12767 073763  200  061  040          .ASCII  <CRLF>/1 = READ CSR/
12768 074000  200  062  040          .ASCII  <CRLF>/2 = LOAD CSR/
12769 074015  200  063  040          .ASCII  <CRLF>/3 = EXAMINE MEMORY/
12770 074040  200  064  040          .ASCII  <CRLF>/4 = MODIFY MEMORY/
12771 074062  200  065  040          .ASCII  <CRLF>/5 = SELECT BANK & PATTERN/
12772 074114  200  066  040          .ASCII  <CRLF>/6 = TYPE CONFIGURATION MAP/
12773 074147  200  067  040          .ASCII  <CRLF>'7 = BATTERY BACKUP - P/F TEST''
12774 074205  200  070  040          .ASCII  <CRLF>/8 = SOB-A-LONG TEST/
12775 074231  200  071  040          .ASCII  <CRLF>/9 = ERROR SUMMARY/
12776 074253  200  061  060          .ASCII  <CRLF>/10= REFRESH TEST/
12777 074274  200  061  061          .ASCII  <CRLF>/11= SET FILL COUNT/
12778 074317  200  061  062          .ASCII  <CRLF>/12= ENTER KAMIKAZE MODE/
12779 074347  200  061  063          .ASCII  <CRLF>/13= EXIT KAMIKAZE MODE/
12780 074376  200  061  064          .ASCII  <CRLF>/14= TURN CACHE OFF/
12781 074421  200  061  065          .ASCII  <CRLF>/15= TURN CACHE ON/
12786 074443  200  061  066          .ASCII  <CRLF>/16= TEST ONLY SELECTED BANKS/
12787 074500  200  061  067          .ASCII  <CRLF>/17= RESUME TESTING ALL BANKS/
12788 074535  015  012  000          .BYTE   15,12,0
12789 074540  200  127  110  MSG022: .ASCIZ  <CRLF>/WHICH CSR(0-F)? /
12790 074562  200  103  123  MSG023: .ASCIZ  <CRLF>/CSR WORD? /
12791 074576  200  124  110  MSG025: .ASCIZ  <CRLF>/THIS CSR DOES NOT EXIST/
12792 074627  200  103  117  MSG026: .ASCIZ  <CRLF>/COMMAND:/
12793 074641  200  117  114  MSG027: .ASCIZ  <CRLF>/OLD CSR WAS/
12794 074656  200  103  123  MSG028: .ASCIZ  <CRLF>/CSR IS NOW/
12795 074672  200  105  130  MSG029: .ASCIZ  <CRLF>/EXAMINE MEMORY/
12796 074712  200  102  101  MSG030: .ASCIZ  <CRLF>/BANK(0-177)? /
12797 074731  200  120  110  MSG031: .ASCIZ  <CRLF>/PHYSICAL ADDRESS(0-17757776)? /
12798 074771  200  120  101  MSG032: .ASCIZ  <CRLF>/PARITY ABORT/<32>
12799 075010  200  124  111  MSG033: .ASCIZ  <CRLF>/TIMEOUT TRAP/<32>
12800 075027  200  102  131  MSGA34: .ASCIZ  <CRLF>/BYPASSING ECC LOGIC TESTS ON BANK /
12801 075073  040  104  125  MSGB34: .ASCIZ  / DUE TO LACK OF SBE FREE LOCATIONS/
```

```
12802 075136  121  126  000  MSG035: .ASCIZ  /QV/
12803 075141  200  115  117  MSG036: .ASCIZ  <CRLF>/MODIFY MEMORY/
12804 075160  200  117  114  MSG037: .ASCIZ  <CRLF>/OLD DATA WAS /
12805 075177  200  104  101  MSG038: .ASCIZ  <CRLF>/DATA IS NOW /
12806 075215  200  111  116  MSG039: .ASCIZ  <CRLF>/INPUT NEW DATA? /
12807 075237  200  123  105  MSG040: .ASCIZ  <CRLF>/SELECT BANK & PATTERN TEST/
12808 075273  200  102  101  MSG041: .ASCIZ  <CRLF>/BANK NOT ACCESSABLE/
12809 075320  200  120  101  MSG042: .ASCIZ  <CRLF>/PATTERN(0-35)? /
12810 075341  200  120  101  MSG043: .ASCIZ  <CRLF>/PATTERN 0 DATA IS? /
12811 075366  200  124  117  MSG046: .ASCIZ  <CRLF>/TO ESCAPE TYPE ANY KEY/<CRLF><12><12>
12812 075421  200  124  105  MSG047: .ASCIZ  <CRLF>/TEST COMPLETE/
12813 075440  040  116  117  MSG048: .ASCIZ  / NOT AVAILABLE NOW - TRY LATER!/
12814 075500  200  102  101  MSG049: .ASCIZ  <CRLF>/BANK REQUIRES RELOCATION/
12815 075532  200  102  101  MSG050: .ASCII  <CRLF>/BATTERY BACKUP TEST/
12816 075556  200  127  122          .ASCIZ  <CRLF>/WRITING & CHECKING ADDRESS PATTERN AS BACKGROUND/
12817 075640  200  120  117  MSG051: .ASCIZ  <CRLF>/POWER RECOVERY/
12818 075660  200  122  105  MSG052: .ASCIZ  <CRLF>/REMOVE SYSTEM POWER FOR/
12819 075711  040  123  105  MSG053: .ASCIZ  / SECONDS MAX!/
12820 075727  200  116  117  MSG054: .ASCIZ  <CRLF>/NOW STARTING READ TEST OF MEMORY BANKS/
12821 075777  200  123  117  MSG055: .ASCIZ  <CRLF>/SOB-A-LONG TEST/
12822 076020  200  102  105  MSG056: .ASCIZ  <CRLF>/BELL = EACH PASS COMPLETE/
12823 076053  200  040  040  MSG058: .ASCIZ  <CRLF>/  CSR    CSR .../
12824 076075  077  077  077  MSG061: .ASCIZ  /??????/
12825 076104  111  116  120  MSG062: .ASCIZ  /INPUT MUST BE A/
12826 076124  116  040  117  MSG063: .ASCIZ  /N OCTAL /
12827 076135  116  125  115  MSG064: .ASCIZ  /NUMBER/<CRLF>
12828 076145  040  104  105  MSG065: .ASCIZ  / DECIMAL /
12829 076157  200  105  122  MSG066: .ASCIZ  <CRLF>/ERROR COUNT EXCEEDED 20 - ABORTING FOR XXDP CHAIN/
12830 076242  106  101  124  MSG067: .ASCIZ  /FATAL /
12831 076251  113  040  127  MSG070: .ASCIZ  /K WORDS OF MEMORY TOTAL/<CRLF>
12832 076302  113  040  117  MSG071: .ASCIZ  /K OF BIPOLAR/<CRLF>
12833 076320  113  040  117  MSG072: .ASCIZ  /K OF MF11S-K/<CRLF>
12834 076336  200  122  105  MSG073: .ASCIZ  <CRLF>/REFRESH TEST/
12835 076354  200  122  105  MSG075: .ASCIZ  <CRLF>/RELOCATION NOT POSSIBLE/<32>
12836 076406  200  040  040  MSG076: .ASCIZ  <CRLF>/  BANK  ERRORS/<CRLF>
12837 076427  200  105  116  MSG077: .ASCIZ  <CRLF>/END PASS #/
12838 076443  040  105  122  MSG079: .ASCIZ  / ERROR(S) DETECTED/<CRLF>
12845 076467  200  106  111  MSG085: .ASCIZ  <CRLF>/FILL COUNT(OCTAL)? /
12853 076514  200  113  105  MSG088: .ASCIZ  <CRLF>/KERNEL STACK/
12854 076532  200  123  125  MSG089: .ASCIZ  <CRLF>/SUPERVISOR STACK/
12855 076554  200  125  123  MSG090: .ASCIZ  <CRLF>/USER STACK/
12856 076570  040  111  123  MSG091: .ASCIZ  / IS EMPTY/
12857 076602  122  105  114  MSG092: .ASCIZ  /RELOCATED  /
12858 076616  102  101  116  MSG093: .ASCIZ  /BANK=/
12859 076624  040  040  120  MSG095: .ASCIZ  /  PAT=/
12867 076633  200  105  116  MSG101: .ASCIZ  <CRLF>/ENTERING KAMIKAZE MODE/
12868 076663  200  114  105  MSG102: .ASCIZ  <CRLF>/LEAVING KAMIKAZE MODE/
12869 076712  200  114  105  MSG103: .ASCIZ  <CRLF>/LEAVING FIELD SERVICE MODE/<CRLF>
12870 076747  032  000       MSG104: .BYTE   32,0                              ;CONTROL Z
12871 076751  200  105  116  MSG105: .ASCIZ  <CRLF>/ENTER BANKS IN OCTAL - USE NUMBER OUTSIDE RANGE TO TERMINATE (177)/
12872 077055  200  103  101  MSG106: .ASCIZ  <CRLF>/CACHE IS OFF/
12873 077073  200  103  101  MSG107: .ASCIZ  <CRLF>/CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)/
12878 077150  200  117  116  MSG110: .ASCIZ  <CRLF>/ONLY SELECTED BANKS WILL BE TESTED/
12879 077214  200  101  114  MSG111: .ASCIZ  <CRLF>/ALL BANKS WILL BE TESTED/
12880 077246  113  040  117  MSG112: .ASCIZ  /K OF MS11-L/<CRLF>
12881 077263  113  040  117  MSG113: .ASCIZ  /K OF MS11-M/<CRLF>
12882 077300  113  040  117  MSG114: .ASCIZ  /K OF UNIBUS PARITY/<CRLF>
```

```
 12883 077324      200      040      040  MSG115: .ASCIZ  <CRLF>/        01234567/
 12884 077346      200      040      040  MSG116: .ASCIZ  <CRLF>''   11/34''
 12885 077360      200      040      040  MSG117: .ASCIZ  <CRLF>''   11/44''
 12886 077372      200      040      040  MSG118: .ASCIZ  <CRLF>''   11/60''
 12887 077404      200      040      040  MSG119: .ASCIZ  <CRLF>/   NO/
 12888 077413      040      103      101  MSG120: .ASCIZ  / CACHE AVAILABLE/
 12889 077434      040      103      101  MSG121: .ASCIZ  / CACHE BYPASSED/
 12890 077454      200      103      123  MSG122: .ASCII  <CRLF>/CSR NUMBER /
 12891 077470      000                   MSGA122:.BYTE   0
 12892 077471      040      103      117          .ASCIZ  / CONTROLS TOO MANY BANKS/
 12893 077522      200      120      122  MSG123: .ASCIZ  <CRLF>/PROGRAM RELOCATED - ECC TESTS INHIBITED/
 12894 077573      200      116      125  MSG124: .ASCIZ  <CRLF>/NUMBER OF CSR'S IS WRONG IN BANK /
 12895 077636      040      120      101  MSG125: .ASCIZ  / PASSES COMPLETED/
 12896                                           .EVEN
 12902 077660                               $$END
 12903 077660                          END:
 12904 077660  005304                             .PRINT  60000-SUPLIMIT  ;SUPERVISOR ADDRESSES LEFT
 12916 077660  000120                             .PRINT  100000-END      ;ADDRESSES LEFT IN 16K
 12920         000200                             .END START3
```

```
ABORTF  002140        B10     013456        CACHVE= 000114        DETAIL  055734        DT22    066470
ACFLAG  002114        B11     015074        CBCSR = 104474        DETFLA  002212        DT23    066476
ACTFLA  002320        B12     015342        CB1CSR= 104475        DETPSW  002210        DT24    066520
ADDRES  002032        B13     015350        CHECK   002274        DETR0   002172        DT25    066540
ANA2    006046        B14     015366        CHKDIS= 104504        DETR1   002174        DT26    066550
APTDOW  043650        B15     015430        CHKGEN  040360        DETR2   002176        DT27    066556
APTECC  002374        B16     015502        CHKTAB  040466        DETR3   002200        DT3     066224
APTFLA  002322        B17     020662        CHK1DI= 104505        DETR4   002202        DT4     066234
APTHAN  013412        B2      011512        CKEND   057476        DETR5   002204        DT5     066244
APTHLT  043716        B20     021070        CKSWR = 104410        DETSP   002206        DT6     066262
APTPAR  002372        B21     023024        CLRCSR= 104502        DET1    056422        DT7     066276
APTSIZ  002414        B22     023030        CLR1CS= 104503        DF11    066720        DUMMY   002170
BACKGN  035156        B23     023224        CMD10B  047350        DF13    066731        DUMPCS= 000062
BAD     002050        B24     023232        CMD10C  047424        DF14    066741        ECCDIS= 104470
BADPC   002020        B25     023522        CMD16L= 000053        DF15    066750        ECCINI= 104472
BADPSW  002030        B26     033076        CMD17A  047734        DF2     066576        ECC1DI= 104471
BADSP   002024        B27     033134        CMD5B   045622        DF3     066622        ECC1IN= 104473
BADSTA  036626        B3      011554        CMD5C   046102        DF4     066635        EMTVEC= 000030
BADXOR  002056        B30     033150        CMD7A   046352        DF5     066650        EM11    067257
BAD2    002052        B31     033270        CMD8B   046652        DF6     066663        EM12    067301
BAD3    002054        B32     033450        CMD8C   046726        DF7     066676        EM13    067325
BAFPAF  013546        B33     033472        CONFGE  002420        DF8     066711        EM14    067357
BAFPAR  013654        B34     034666        CONFIE  003630        DF9     066713        EM15    067423
BAKPAT  002572        B35     037444        CONFIG  002624        DH1     071437        EM17    067471
BANK    002100        B36     037450        CONTFL  002214        DH10    072007        EM19    067531
BANKIN  002102        B37     037610        CONTP   057252        DH11    072105        EM2     066757
BANKMO  042350        B4      012202        CONTRL= 177746        DH12    072123        EM20    067606
BANKOK  043350        B40     037614        CONTS   057646        DH13    072130        EM21    067670
BAWPAF  013762        B41     040020        CONTS1  057212        DH14    072225        EM22    067727
BAWPAR  014112        B42     040024        CONTS2  057650        DH15    072304        EM23    067754
BGTEST  034534        B43     041216        CONTT   057572        DH16    072423        EM24    070003
BIT0  = 000001        B44     041224        COUNT   002340        DH19    072440        EM25    070062
BIT1  = 000002        B45     041354        CPUBIT  002104        DH2     071474        EM26    070107
BIT10 = 002000        B46     041370        CPUERR= 177766        DH23    072445        EM27    070160
BIT11 = 004000        B47     046410        CR    = 000015        DH24    072524        EM29    070250
BIT12 = 010000        B5      012344        CRLF  = 000200        DH25    072573        EM3     067015
BIT13 = 020000        B50     046656        CSR     002144        DH26    072617        EM30    070332
BIT14 = 040000        B51     047100        CSRADD= 172100        DH27    072635        EM31    070451
BIT15 = 100000        B52     047354        CSRCAS  016254        DH3     071517        EM32    070551
BIT2  = 000004        B53     047650        CSRFBA  002224        DH5     071553        EM33    070656
BIT3  = 000010        B54     047650        CSRFIR  002220        DH6     071632        EM35    070764
BIT4  = 000020        B55     047746        CSRHOL  002476        DH7     071677        EM36    071051
BIT5  = 000040        B56     050660        CSRINC  002300        DIAGFL  002002        EM4     067047
BIT6  = 000100        B57     050664        CSRINF  002432        DISPLA  002600        EM40    071120
BIT7  = 000200        B6      012406        CSRINT  002230        DISPRE= 000174        EM5     067115
BIT8  = 000400        B60     051156        CSRLAS  002222        DISPTB  013062        EM50    071172
BIT9  = 001000        B61     051164        CSRLBA  002226        DOBACK  013446        EM51    071226
BLOCK1  043720        B62     056166        CSRLOO  002302        DSWR  = 177570        EM52    071310
BLOCK2  043740        B63     056172        CSRNO   002146        DT1     066212        EM53    071335
BLOCK3  043754        B64     056270        CSROUT  040012        DT10    066326        EM55    071364
BMFLAG  002126        B65     056362        CSRSTU= 000014        DT11    066350        EM56    071405
BOOT    043426        B66     056446        CTLKVE  002142        DT12    066356        EM6     067172
BOOT1   043472        B7      012672        DATARG= 177754        DT13    066362        EM7     067217
BRGOBB  034536        CACHKF  002520        DATBUF  002234        DT14    066404        ENASBE= 104506
BSIZE   002344        CACHKN  002514        DBEMSK  002250        DT16    066422        ENA1SB= 104507
B0      004466        CACHOF- 104424        DDISP = 177570        DT17    066452        END     077660
B1      010732        CACHON= 104423        DEENER- 104421        DT20    066460        ENERGI= 104420
```

```
ENEXBK  043340          E53     047710          HT    = 000011          LOWMAP  042304          L155    020400
ERRADD  002430          E54     047710          I       002422          LSIZE   002350          L156    020410
ERRGEN= 104512          E55     047764          IIII  = 177777          LWDBE = 000037          L157    020410
ERRMAX  002524          E56     050720          ILLCSR  012102          LWSBE = 000035          L16     007734
ERROR = 104000          E57     050720          IMPTES  011222          L0      004476          L160    020420
ERRPC   002016          E6      012434          INCBNK  043412          L1      004540          L161    020456
ERRPSW  002026          E60     051220          INCPAT  043366          L10     005034          L162    020466
ERRSP   002022          E61     051220          INCRPT  043366          L100    013034          L163    020466
ERRVEC= 000004          E62     056234          INHBAN  002510          L101    013120          L164    020524
EUFLAG  002130          E63     056234          INHECC  002506          L102    013162          L165    020534
EVEN    002334          E64     056320          INTFLA  002134          L103    013262          L166    020534
EXBANK  042700          E65     056414          INT64K  002136          L104    013272          L167    020614
EXCMD3  044756          E66     056500          INVALI= 104511          L105    013442          L17     007736
EXCMD4  045274          E7      013012          IOTVEC= 000020          L106    013442          L170    020624
EXIT    043534          FASTCI= 177640          JMPRL1  041724          L107    013530          L171    020624
EXIT2   043540          FATAL$  002062          KAMIKA  002004          L11     005042          L172    020734
E0      004514          FCMD10  047216          KAMITE  025110          L110    013642          L173    020714
E1      011150          FCMD11  047510          KDIAG = 000010          L111    013750          L174    020714
E10     013544          FCMD12  047536          KDPAR0= 172360          L112    014100          L175    020772
E11     015114          FCMD13  047560          KDPAR6= 172374          L113    014230          L176    021026
E12     015666          FCMD14  047600          KDPAR7= 172376          L114    014360          L177    021030
E13     015652          FCMD15  047622          KERNEL= 104417          L115    014532          L2      004644
E14     015606          FCMD16  047640          KERSTK= 002000          L116    014662          L20     010540
E15     015606          FCMD17  047724          KFLAG   002500          L117    015034          L200    021056
E16     015556          FIELDS  044004          KIPAR0= 172340          L12     005072          L201    021060
E17     020726          FINDBA= 000045          KIPAR4= 172350          L120    015114          L202    021144
E2      011714          FIRST = 060000          KIPAR5= 172352          L121    015152          L203    021262
E20     021170          FLIPDJ  002556          KIPAR6= 172354          L122    015162          L204    021264
E21     023202          FLIPWA  035026          KIPDR0= 172300          L123    015606          L205    022062
E22     023166          FLUSH   013166          KMAP  = 104422          L124    015570          L206    022114
E23     023404          FSCMD0  044200          KPFLAG  002112          L125    015634          L207    022160
E24     023370          FSCMD1  044302          KSIZE   002346          L126    016076          L21     010554
E25     023600          FSCMD2  044406          KSTACK  002534          L127    016206          L210    022416
E26     033442          FSCMD3  044550          LAST  = 157776          L13     005324          L211    022426
E27     033426          FSCMD4  045022          LASTBA  002526          L130    016344          L212    022426
E3      011700          FSCMD5  045340          LASTBL  002530          L131    016372          L213    023152
E30     033242          FSCMD6  046172          LASTER  002014          L132    016376          L214    023122
E31     033412          FSCMD7  046200          LBLS0 = 000456          L133    016400          L215    023152
E32     033466          FSCMD8  046520          LBLS1 = 000066          L134    016444          L216    023216
E33     033512          FSCMD9  047012          LBLS2 = 000450          L135    016500          L217    023354
E34     035010          FSINFL  002412          LBLS3 = 000443          L136    016504          L22     010700
E35     037516          FSPAT   046006          LBLS4 = 000315          L137    016506          L220    023324
E36     037516          FSSTAC  002266          LBLS5 = 000317          L14     005324          L221    023354
E37     037662          FS1     044070          LBLS6 = 000016          L140    016654          L222    023564
E4      012336          FS7FLA  002416          LCSROU= 000041          L141    017322          L223    023564
E40     037662          FULLRE  002512          LCSRRE= 000060          L142    020162          L224    023672
E41     040054          GBLENG= 000076          LCSRSA= 000056          L143    020172          L225    023644
E42     040054          GETDAT  050350          LEGALC= 000003          L144    020172          L226    023722
E43     041354          GETDA1  050446          LF    = 000012          L145    020230          L227    024112
E44     041334          GETDIS  054374          LINK1   002472          L146    020240          L23     010656
E45     041574          GOOD    002042          LINK2   002474          L147    020240          L230    024440
E46     041502          GOOD2   002044          LKS   = 177546          L15     007730          L231    024724
E47     046472          GOOD3   002046          LOADBA  002402          L150    020270          L232    024760
E5      012404          GTSWR = 104407          LOADCS= 104425          L151    020274          L233    025060
E50     046720          HEADER  002552          LOADER= 000043          L152    020324          L234    025132
E51     047202          HIPAT   043402          LOADHO  002536          L153    020334          L235    025140
E52     047416          HOLDLO= 000011          LOOP    013034          L154    020334          L236    025144
```

```
L237    026736  L320    041540  L402    053260  L54     012302  MSG010  073402
L24     010670  L321    041540  L403    053276  L55     012256  MSG011  073414
L240    026734  L322    041570  L404    053300  L56     012250  MSG012  073502
L241    026736  L323    041706  L405    053320  L57     012254  MSG013  073577
L242    030356  L324    043042  L406    053332  L6      005042  MSG014  073601
L243    033120  L325    043266  L407    053350  L60     012300  MSG015  073603
L244    033130  L326    043450  L41     011416  L61     012274  MSG016  073605
L245    033242  L327    043554  L410    053352  L62     012300  MSG017  073617
L246    033172  L33     011134  L411    053366  L63     012326  MSG018  073630
L247    033204  L330    043560  L412    053570  L64     012326  MSG019  073633
L25     010704  L331    043566  L413    053576  L65     012326  MSG020  073637
L250    033222  L332    043602  L414    053624  L66     012422  MSG021  073673
L251    033230  L333    043614  L415    053642  L67     012472  MSG022  074540
L252    033254  L334    044026  L416    053654  L7      005042  MSG023  074562
L253    033412  L335    044036  L417    053666  L70     012644  MSG025  074576
L254    033312  L336    044170  L42     011416  L71     013034  MSG026  074627
L255    033324  L337    044230  L420    053700  L72     013002  MSG027  074641
L256    033356  L34     011066  L421    053722  L73     012720  MSG028  074656
L257    033342  L340    044234  L422    053770  L74     012722  MSG029  074672
L26     011134  L341    044264  L423    054050  L75     012762  MSG030  074712
L260    033354  L342    044276  L424    054064  L76     012776  MSG031  074731
L261    033400  L343    045450  L425    054066  L77     013032  MSG032  074771
L262    033366  L344    045510  L426    054172  MAINT = 177750  MSG033  075010
L263    033400  L345    045546  L427    054222  MAPH0 = 170202  MSG035  075136
L264    033512  L346    045720  L43     011416  MAPL0 = 170200  MSG036  075141
L265    035372  L347    045734  L430    054232  MAPL1 = 170204  MSG037  075160
L266    035412  L35     011134  L431    054532  MAPPER  040526  MSG038  075177
L267    036030  L350    045746  L432    054532  MEMDON  013102  MSG039  075215
L27     011134  L351    046246  L433    054674  MJPAT   016700  MSG040  075237
L270    036216  L352    046250  L434    054634  MJTEST  016574  MSG041  075273
L271    036320  L353    046456  L435    054654  MKCONT  015142  MSG042  075320
L272    037230  L354    046570  L436    054674  MKCSRT  016264  MSG043  075341
L273    037236  L355    046572  L437    055070  MKFLAG  002116  MSG046  075366
L274    037474  L356    046704  L44     011422  MKLOOP  015324  MSG047  075421
L275    037474  L357    047202  L440    054742  MKPAT   016514  MSG048  075440
L276    037536  L36     011206  L441    055066  MKTEST  016354  MSG049  075500
L277    037544  L360    047166  L442    055016  MMR0  = 177572  MSG050  075532
L3      004666  L361    047136  L443    055030  MMR1  = 177574  MSG051  075640
L30     011134  L362    047266  L444    055066  MMR2  = 177576  MSG052  075660
L300    037640  L363    047270  L445    055076  MMR3  = 172516  MSG053  075711
L301    037640  L364    047402  L446    055376  MMTRAP  036602  MSG054  075727
L302    037702  L365    047672  L447    056212  MMVEC = 000250  MSG055  075777
L303    037770  L366    047674  L45     011700  MSEEDH  002546  MSG056  076020
L304    040036  L367    050112  L450    056416  MSEEDL  002550  MSG058  076053
L305    041170  L37     011212  L451    056422  MSGA12  077470  MSG061  076075
L306    041204  L370    050126  L452    056502  MSGA34  075027  MSG062  076104
L307    041216  L371    050132  L453    056506  MSGB34  075073  MSG063  076124
L31     011017  L372    050150  L454    057612  MSG000  072711  MSG064  076135
L310    041216  L373    050276  L455    057702  MSG001  072723  MSG065  076145
L311    041320  L374    050300  L456    057704  MSG002  073005  MSG066  076157
L312    041350  L375    050344  L46     011700  MSG003  073062  MSG067  076242
L313    041472  L376    050570  L47     011654  MSG004  073167  MSG070  076251
L314    041472  L377    050676  L5      004762  MSG005  073275  MSG071  076302
L315    041472  L4      004704  L50     011626  MSG006  073307  MSG072  076320
L316    041452  L40     011214  L51     011636  MSG007  073344  MSG073  076336
L317    041472  L400    051176  L52     011656  MSG008  073356  MSG075  076354
L32     011016  L401    053246  L53     012326  MSG009  073370  MSG076  076406
```

```
MSG077  076427    MTPC20  032614    MT0024  022150    O.CTLF  062330    O.TBIT  063764
MSG079  076443    MTPC21  033004    MT0025  022402    O.C1    064120    O.TBT = 000020
MSG085  076467    MTPC24  033620    MT0026  022450    O.DCD   062640    O.TCL2  062562
MSG088  076514    MTPC25  034236    MT0027  022740    O.DCDA  063264    O.TCSR= 177564
MSG089  076532    MTPC26  034354    MT0030  023412    O.DCD1  062670    O.TDB = 177566
MSG090  076554    MTPD03  025742    MT0031  023712    O.DCD3  062650    O.TL    065466
MSG091  076570    MTPD20  032644    MT0032  024102    O.DOT   061702    O.TRTC  065510
MSG092  076602    MTPD21  033040    MT0033  024430    O.ERR   062630    O.TVEC= 000014
MSG093  076616    MTPD25  034102    MT0034  024616    O.ERR1  063712    O.TYPE  065246
MSG095  076624    MTPD26  034374    MT0035  024762    O.ERR2  063304    O.TYP1  065124
MSG101  076633    MTPE20  032674    MT020Y  021404    O.ERR3  063270    O.UIN   062072
MSG102  076663    MTPE25  034124    MT020Z  021220    O.FIL   061716    O.UPC   062020
MSG103  076712    MTP000  025514    MT0999  025074    O.FTYP  065050    O.URO   062002
MSG104  076747    MTP001  025540    MT1     015122    O.GET   065126    O.USP   062016
MSG105  076751    MTP002  025572    MT2     015126    O.GO    063716    O.UST   062022
MSG106  077055    MTP005  026036    MUT     002106    O.GOGO  063744    O.WRD   063100
MSG107  077073    MTP006  026072    NEMCNT  002066    O.GO2   064016    O.WRD1  063156
MSG110  077150    MTP007  026272    NEWBAN  002270    O.ID    065420    O.WST   065322
MSG111  077214    MTP010  026372    NEWKER  042600    O.IDND= 065433    O.XXX   061704
MSG112  077246    MTP011  026500    NEWLOA  042646    O.LG  = 000020    PADDRE  002034
MSG113  077263    MTP012  027244    NOERRO  002400    O.LGCH  065442    PAFBAF  014242
MSG114  077300    MTP013  027632    NOFSMO  002376    O.MIN   062622    PAFBAW  014372
MSG115  077324    MTP014  030346    NONEM   002076    O.MINS  061723    PARBAF  014544
MSG116  077346    MTP015  031130    NONEXI  036524    O.ODT   062114    PARBAW  014674
MSG117  077360    MTP016  031674    NOPAR   002074    O.OFST  063606    PARCNT  002070
MSG118  077372    MTP017  032456    NOSCOP  002410    O.OLD   063274    PARITY  036420
MSG119  077404    MTP020  032534    NOSUPE  002426    O.OP1   063300    PARTHE  002264
MSG120  077413    MTP022  033070    NOTAB   002342    O.OP2   063340    PARVEC= 000114
MSG121  077434    MTP025  033636    NO22BI  002424    O.OP2A  063346    PASFLG  002256
MSG122  077454    MTP030  034412    NULLFL  002314    O.ORAB  062504    PATERR  002072
MSG123  077522    MTP031  034422    OLDCAC  002262    O.ORPC  062462    PATPLU  004460
MSG124  077573    MTP032  034500    OLDCSR  002150    O.ORRB  062514    PATTER  002110
MSG125  077636    MTP033  034532    ONES    002554    O.P     061713    PCBUMP  002276
MSIZE   002352    MTP034  034630    O.ADR1  062026    O.PRI   062024    PCONFI  035254
MTA030  023424    MTP035  034654    O.BACK  063422    O.PROC  064052    PCONFS  035570
MTB020= 000017    MTST3   010560    O.BD    065506    O.PR1   064104    PCONF1  035500
MTEST   015046    MTV020  021216    O.BIAS  061724    O.RALL  063550    PCONF2  035536
MTLA11  026502    MT0000  016760    O.BKP = 000016    O.RCSR= 177560    PDP110  036614
MTLB11  026514    MT0001  017036    O.BKPT  063444    O.RDB = 177562    PD1     050570
MTLC11  026526    MT0002  017154    O.BK1   064200    O.REGT  062410    PERA05  052632
MTLD11  026622    MT0003  017312    O.BRK   064162    O.REM   064722    PERBNK  053464
MTL020  020654    MT0004  017542    O.BW    061706    O.RORA  065412    PERECC  053544
MTPA03  025624    MT0005  017662    O.BYT   063112    O.RRST  065312    PERRAB  053302
MTPA04  025762    MT0006  020014    O.CAD   061700    O.RSB   064570    PERRAW  053230
MTPA20  032534    MT0007  020050    O.CADV  064754    O.RSR   064536    PERRA3  050136
MTPA21  032720    MT0010  020112    O.CLGT= 000024    O.RSTT  064654    PERRA7  053354
MTPA24  033544    MT0011  020146    O.CLSE  065334    O.S     061711    PERR01= 104427
MTPA25  034154    MT0012  020214    O.CMFD  061720    O.SCAN  062674    PERR02= 104430
MTPA26  034304    MT0013  020310    O.CR    065434    O.SCRN  061722    PERR03= 104431
MTPB03  025664    MT0014  020364    O.CRET  063260    O.SEMI  063062    PERR04= 104432
MTPB04  026016    MT0015  020442    O.CRLF  065366    O.SEQ   061710    PERR05  052626
MTPB20  032564    MT0016  020510    O.CRLS  065374    O.SMFD  061721    PERR06  052654
MTPB21  032750    MT0017  020556    O.CSR1  061714    O.SNGL  062566    PERR07= 104433
MTPB24  033604    MT0020  020600    O.CSR2  061715    O.STM = 000340    PERR10= 104434
MTPB25  034176    MT0021  021630    O.CT    062050    O.SVR   064500    PERR11= 104435
MTPB26  034320    MT0022  022052    O.CTLC  062302    O.SVTT  064620    PERR12= 104436
MTPC03  025724    MT0023  022104    O.CTLE  062362    O.T     061712    PERR13= 104437
```

SYMBOL TABLE

```
PERR14= 104440   SBEMSK  002244   SUPDR6  002166   TST1    005344   $AUTO   002060
PERR15= 104441   SBENT   016226   SUPLIM  052474   TST2    007554   $BANK   002011
PERR16= 104442   SBETES  015750   SUPSTK= 000740   TST3    007736   $BASE   061332
PERR17= 104443   SCOPE  = 000004   SWAPAT  002574   TST4    010714   $BELL   002613
PERR20= 104444   SDPAR0= 172260   SWR     002576   TST5    013034   $CACHF  036732
PERR21= 104445   SDPAR5= 172272   SWREG  = 000176   TST6    013120   $CACHN  036706
PERR22= 104446   SDPAR6= 172274   SW0    = 000001   TYPDS  = 104405   $CBCSR  037370
PERR23= 104447   SDPAR7= 172276   SW1    = 000002   TYPEIT= 104401   $CB1CS  037412
PERR24= 104450   SEEDHI  002542   SW10   = 002000   TYPOC  = 104402   $CDW1   061336
PERR25= 104451   SEEDLO  002544   SW11   = 004000   TYPOS  = 104403   $CDW2   061340
PERR26= 104452   SELONL  002000   SW12   = 010000   TYPS0  = 000000   $CHARC  052452
PERR27= 104453   SETPAT  043402   SW13   = 020000   TYPS1  = 000002   $CHKDI  037764
PERR30= 104454   SHADL1  010610   SW14   = 040000   TYPS2  = 000000   $CHK1D  040000
PERR31= 104455   SHUTUP  043566   SW15   = 100000   TYPS3  = 000000   $CKSWR  057174
PERR32= 104456   SIPAR0= 172240   SW2    = 000004   TYPS4  = 000000   $CLRCS  037742
PERR33= 104457   SIPAR3= 172246   SW3    = 000010   TYPS5  = 000000   $CLR1C  037754
PERR34= 104460   SIPAR5= 172252   SW4    = 000020   TYPS6  = 000002   $CMTAG  002000
PERR35= 104461   SIPAR6= 172254   SW5    = 000040   T12A    031674   $CMTGE  002514
PERR36= 104462   SIPDR0= 172200   SW6    = 000100   T12B    031716   $CNTLC  060346
PERR37= 104463   SIZE   = 040000   SW7    = 000200   UDPAR0= 177660   $CNTLG  060360
PERR40= 104464   SKIPKA  002006   SW8    = 000400   UDPAR7= 177676   $CNTLK  057564
PERR41= 104465   SKIPMK  002312   SW9    = 001000   UIPAR0= 177640   $CNTLU  060353
PERR42= 104466   SKPERR  002064   TAG2$    010204   UIPAR1= 177642   $CPUOP  061304
PERR43= 104467   SKUB    041570   TAG3$    010236   UIPAR2= 177644   $CRLF   002620
PERXOR  053440   SOBK    002532   TAG4$    025240   UIPAR3= 177646   $DBLK   057164
PFLAG   002120   SOBLEN= 000056   TAG70$   055404   UIPAR4= 177650   $DB20   061136
PGMCSR  002502   SOFTPA  002560   TAG71$   055414   UIPAR5= 177652   $DDW0   061342
PHYADD  002036   SOURCE  002272   TAG72$   055424   UIPAR6= 177654   $DDW1   061344
PSIZE   002354   SPLTCS  002232   TAG73$   055474   UIPDR0= 177600   $DDW2   061346
PSW    = 177776   SSP    =%000006   TAG74$   055534   UNITOP  002366   $DDW3   061350
PWRVEC= 000024   ST     = 177776   TAG75$   055546   UNRELO  042042   $DDW4   061352
QUICK   002406   STACK  = 002000   TAG76$   055560   UPPFLG  002257   $DDW5   061354
QVFLAG  002316   START   003630   TAG77$   055624   USERMA  042516   $DEENE  036676
RANODD  034334   START1  000300   TAG78$   055632   USESTK= 000700   $DEVCT  061266
RDCHR  = 104411   START2  000310   TAG79$   055712   USP    =%000006   $DEVM   061334
RDDEC  = 104414   START3  000200   TAG9$    010034   WARN1   010122   $DIDDO= 000000
RDLIN  = 104412   STAR27  023016   TBG4$    025416   WARN2   025636   $DOAGA  013442
RDOCT  = 104413   STOPOK  002370   TCFIG1  035714   WARN3   025652   $DOAGN  013336
READCS= 104426   STRIPE  002336   TCFIG2  036046   WARN4   025676   $DOWN   051016
READON  002360   SUBAAA  004516   TCFIG3  036234   WARN5   025712   $DTBL   057154
REALPA  002260   SUBAAB  004644   TCONFI  035572   WARN6   035236   $ECCDI  037266
REFRES  033444   SUBAAI  010604   TEMP    002404   WARN6A  035176   $ECCIN  037314
REFSUB  033514   SUBAAP  012144   TEST    005752   WARN6B  035230   $ECC1D  037302
REGCOP  035016   SUBAAR  011464   TESTAD  002362   WARN7   022774   $ECC1I  037330
RELENT  041600   SUBAAS  007550   TESTMO  002522   WASDBE= 104500   $ENASB  037342
RELOCA  041154   SUCCES  002304   TIME    002310   WASSBE= 104476   $ENA1S  037356
RELOC1  041614   SUPDOA  002254   TIMEOU  036570   WAS1DB= 104501   $ENDAD  013326
RESREG= 104416   SUPDO1  025144   TKVEC  = 000060   WAS1SB= 104477   $ENERG  036666
RESTAR  002566   SUPDO2  025160   TMFLAG  002132   WHICHC  047766   $ENV    061276
RESVEC= 000010   SUPDO3  025322   TOOMAN  002356   WOOPEN  051772   $ENVM   061277
RES0    044276   SUPDO4  025336   TOTCSR  002216   WOOPS   051424   $EOP    013172
RES1    044354   SUPDR0  002152   TRAPVE= 000034   WOOPSA  052022   $ERFLG  002012
RES2    044516   SUPDR1  002154   TRT    = 000003   WOOPUP  051610   $ERRGE  040110
RLFLAG  002124   SUPDR2  002156   TSTBAN  010446   WORST   002540   $ERROR  054446
RRFLAG  002122   SUPDR3  002160   TSTDAT  002240   XXDPCH  002324   $ERRTB  065512
RTNVAL=%000000   SUPDR4  002162   TSTRD1  037240   ZEROS   002306   $ERRTY  055110
SAVREG- 104415   SUPDR5  002164   TSTREA= 104510   $APTHD  061356   $ERTTL  002570
```

```
$ESCAP  002332          $MADR3  061320          $PER01  052474          $PWRUP  051022          $TSTM   061362
$ETABL  061276          $MADR4  061324          $PER02  052522          $QUES   002617          $TSTRD  037064
$ETEND  061356          $MAIL   061256          $PER03  052550          $R    = 177777          $TTYIN  060322
$EXHAL  043560          $MAMS1  061306          $PER04  052600          $RAND   061042          $TYPDS  056750
$E$   = 000001          $MAMS2  061312          $PER07  052662          $RDCHR  057704          $TYPE   052240
$FATAL  061260          $MAMS3  061316          $PER10  052704          $RDDEC  060560          $TYPEC  052364
$FILLC  002612          $MAMS4  061322          $PER11  052734          $RDLIN  060024          $TYPEX  052454
$FILLS  002327          $MBADR  061360          $PER12  052754          $RDOCT  060410          $TYPOC  056546
$F$   = 000000          $MNEW   060376          $PER13  052776          $READC  037044          $TYPON  056562
$GTSWR  057340          $MSGAD  061272          $PER14  053016          $RESRE  061004          $TYPOS  056522
$HALT   054714          $MSGLG  061274          $PER15  053040          $SAVRE  060746          $T1   = 000000
$HALT2  061432          $MSGTY  061256          $PER16  053062          $SAVR6  051422          $T2   = 000456
$HIBTS  061356          $MSWR   060365          $PER17  053102          $SCOPE  054154          $UNIT   061270
$HIOCT  060556          $MTYP1  061307          $PER20  053120          $STN  = 000001          $UNITM  061366
$ILLUP  051416          $MTYP2  061313          $PER21  053136          $SVLAD  054346          $USWR   061302
$INVAL  040060          $MTYP3  061317          $PER22  053156          $SV$  = 000000          $VECT1  061326
$ITEMB  002013          $MTYP4  061323          $PER23  053174          $SWR  = 163000          $VECT2  061330
$I$   = 000001          $NOTRA  061426          $PER24  053212          $SWREG  061300          $WASDB  037576
$KERNE  036656          $NULL   002326          $PER25  050054          $T    = 000457          $WASSB  037432
$KMAP   041062          $NWTST= 000001          $PER26  053402          $TESTN  061262          $WAS1D  037712
$K$   = 000062          $OCNT   056744          $PER27  053422          $TKB    002604          $WAS1S  037546
$L    = 000067          $OCTVL  061240          $PER30  050302          $TKS    002602          $XTSTR  054242
$LF     002621          $OCT8 = 061244          $PER31  053612          $TN   = 000007          $Y$   = 000000
$LL   = 000065          $OMODE  056746          $PER32  053710          $TPB    002610          $ZAP42  013306
$LOADC  036750          $OVER   054362          $PER33  053756          $TPFLG  002330          $Z$   = 000000
$LPADR  002562          $O$   = 000000          $PER34  054036          $TPS    002606          $$S   = 000000
$LPERR  002564          $PASS   061264          $PER35  054070          $TRAP   061372          $$T   = 000442
$L$   = 000000          $PASTM  061364          $PER36  054124          $TRAP2  061414          $$TT  = 000450
$MADR1  061310          $PATMA  002010          $PWRDN  050450          $TRPAD  061434          $OFILL  056745
$MADR2  061314

. ABS.  077660      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  26672 WORDS  ( 105 PAGES)
DYNAMIC MEMORY:  20724 WORDS  ( 79 PAGES)
ELAPSED TIME:  01:38:37
MB9C,MB9C/-SP=CZMSDA.SML,MB9C.P11
```